REST e RESTful em Java

Guia Completo para a Comunidade EducaCiência FastCode

A construção de APIs modernas é um pilar da integração entre sistemas. Com o crescimento de apps mobile, microsserviços e sistemas web, entender REST e RESTful é fundamental para qualquer desenvolvedor.

Neste guia, exploraremos a fundo esses conceitos aplicados com **Java + Spring Boot**, além de traçarmos uma **linha do tempo com a evolução do Java LTS** focada no desenvolvimento de APIs.

1. O que é REST?

REST (Representational State Transfer) é um estilo arquitetural criado por **Roy Fielding** em 2000.

Princípios do REST:

Princípio	Explicação
Cliente-Servidor	Separação entre cliente (frontend) e servidor (backend).
Stateless	Nenhum estado é armazenado entre requisições.
Cacheável	Respostas podem ser armazenadas em cache para melhorar performance.
Interface uniforme	Recursos acessados por URIs e manipulados por métodos HTTP.
Camadas	Sistema pode ter múltiplas camadas (ex: gateways, proxies).

Métodos HTTP e suas funções:

Método	Função
GET	Recuperar um recurso ou lista
POST	Criar um novo recurso
PUT	Atualizar um recurso existente
DELETE	Remover um recurso

2. O que é RESTful?

Uma API é chamada de **RESTful** quando **segue os princípios REST** corretamente.

Exemplo prático de URLs RESTful:

Recurso	URL RESTful	Ação
Usuário	GET /usuarios	Listar
Usuário	POST /usuarios	Criar
Usuário	GET /usuarios/1	Buscar
Usuário	PUT /usuarios/1	Atualizar
Usuário	DELETE /usuarios/1	Remover

3. Criando uma API RESTful com Java (Spring Boot)

3.1. Setup com Spring Boot (pom.xml)

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

3.2. Modelo de dados (entidade):

```
public class Usuario {
    private Long id;
    private String nome;
    private String email;
    // Getters e setters
}
```

3.3. Controlador REST:

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioController {

    private List<Usuario> usuarios = new ArrayList<>();

    @GetMapping
    public List<Usuario> listar() { return usuarios; }

    @PostMapping
    public Usuario criar(@RequestBody Usuario u) {
        usuarios.add(u); return u;
    }
```

```
    @GetMapping("/{id}")
    public Usuario buscar(@PathVariable Long id) {
        return usuarios.stream().filter(u -> u.getId().equals(id)).findFirst().orElse(null);
    }

    @PutMapping("/{id}")
    public Usuario atualizar(@PathVariable Long id, @RequestBody Usuario novo) {
        for (Usuario u : usuarios) {
            if (u.getId().equals(id)) {
                u.setNome(novo.getNome());
                u.setEmail(novo.getEmail());
                return u;
            }
        }
        return null;
    }

    @DeleteMapping("/{id}")
    public void remover(@PathVariable Long id) {
        usuarios.removeIf(u -> u.getId().equals(id));
    }
}
```

4. RESTful: Padrão de Rotas

Método HTTP	Rota	Ação da API
GET	/usuarios	Lista todos os usuários
GET	/usuarios/5	Busca usuário com ID 5
POST	/usuarios	Cria um novo usuário
PUT	/usuarios/5	Atualiza usuário com ID 5
DELETE	/usuarios/5	Remove usuário com ID 5

5. Benefícios das APIs RESTful com Java

- **Simplicidade**: HTTP puro, sem SOAP ou RPC.
- **Escalabilidade**: excelente para microsserviços e nuvem.
- **Desempenho**: uso de JSON e cache.
- **Flexibilidade**: clientes diversos (web, mobile, IoT).
- **Integração facilitada**: ferramentas como Swagger, Postman e OpenAPI.

6. Linha do Tempo: Evolução das versões LTS do Java e seu impacto no desenvolvimento de APIs

Ano	Versão LTS	Novidades importantes para APIs RESTful
2014	Java 8	Lambdas, Streams e Optional – facilitam lógica de negócio
2018	Java 11	API HTTP Client nativa (java.net.http) e melhorias de performance
2021	Java 17	Record classes, melhorias de garbage collector, maior segurança
2024	Java 21	Padrões de correspondência (Pattern Matching), threads virtuais (Projeto Loom) – excelente para APIs de alta concorrência
2027	Java 25 (previsto)	Promete avanços em segurança, paralelismo e IA nativa

Entender REST e RESTful é um **fundamento obrigatório** no desenvolvimento de APIs.

Com o **Java e o Spring Boot**, é possível criar **serviços robustos, escaláveis e fáceis de manter**.

A cada nova versão do Java, o desenvolvimento se torna mais produtivo, performático e moderno — e é por isso que a comunidade **EducaCiência FastCode** deve estar sempre atualizada.

Dominar esses padrões é o primeiro passo para criar aplicações de impacto e qualidade profissional.

EducaCiência FastCode para a comunidade