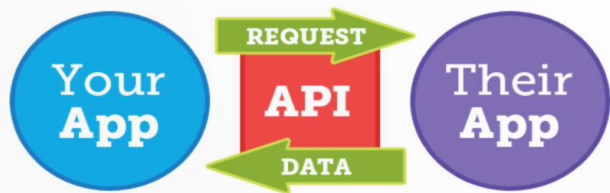


DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

Módulo 9 REST API

Delano Medeiros Beder
delano@dc.ufscar.br

Porque APIs são tão importantes?

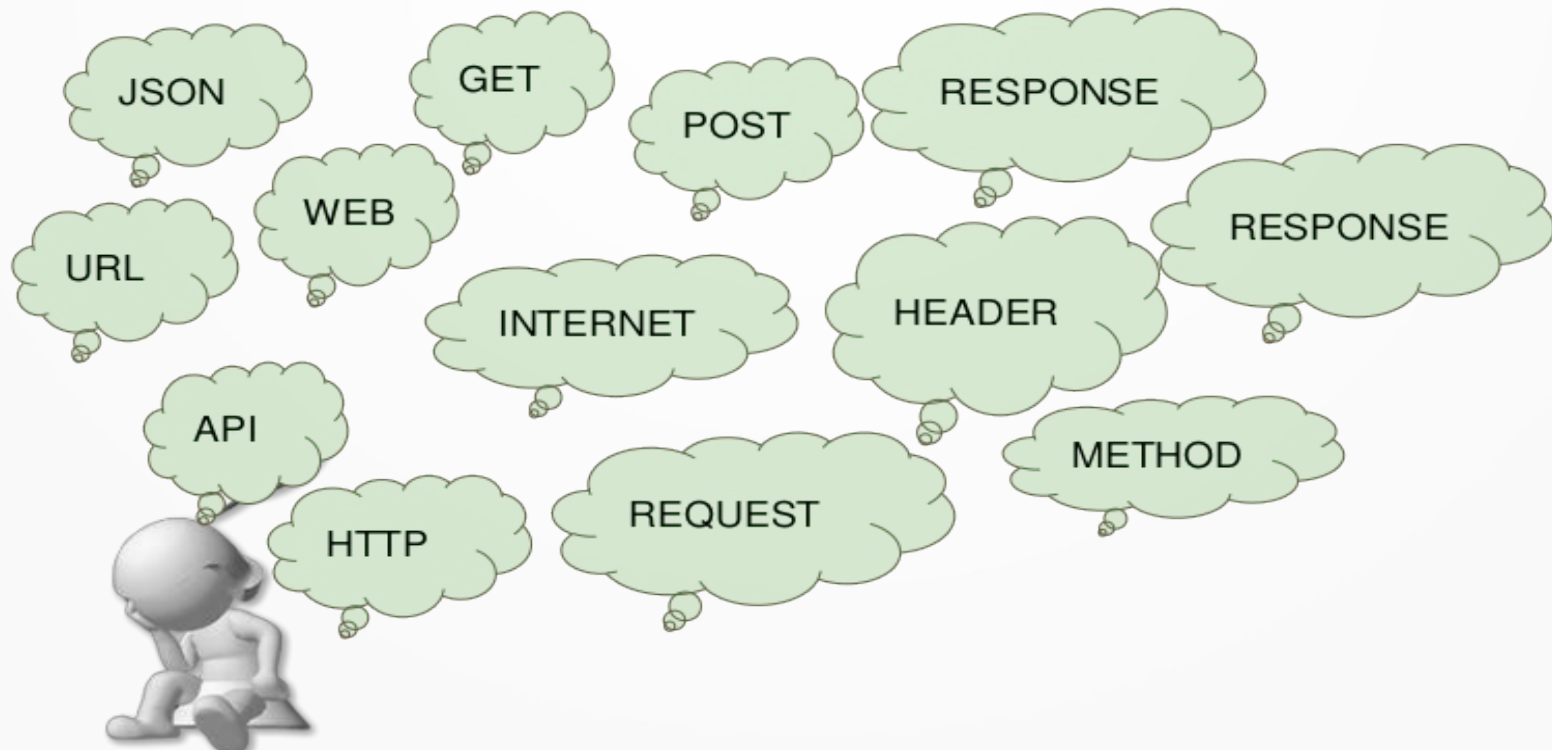


Uma API é um contrato fornecido por um software para outro.

É uma maneira de dois softwares (sistemas) trocarem informações.



REST API ? O que é isso ?



REST ? O que é isso ?

- **REST** é um acrônimo para

REpresentational

State

Transfer

Objetivo: definição de características fundamentais para a construção de aplicações Web seguindo boas práticas.

Representational?? State?? Transfer??

- **REST** é um estilo de arquitetura baseado em padrões da web e no protocolo HTTP.
- Em uma arquitetura baseada em REST, tudo é um **Recurso**.
- Um **recurso** é acessado por meio de uma interface comum baseada nos métodos padrão HTTP.
- Normalmente, você tem um servidor REST que fornece acesso aos **recursos** e um cliente REST que acessa e modifica os **recursos** REST.

Representational?? State?? Transfer??

- Um **recurso** é um elemento abstrato e que nos permite mapear qualquer coisa do mundo real como um elemento para acesso via Web.
- Cada **recurso** deve suportar as operações comuns do protocolo HTTP.
- Os **recursos** são identificados por IDs globais (que normalmente são URIs ou URLs).
- REST permite que recursos tenham diferentes representações, por exemplo, HTML, XML, JSON etc.

Métodos HTTP

- Os métodos POST, GET, PUT e DELETE são normalmente usados em arquiteturas REST.

Método HTTP	Operação CRUD	Descrição
POST	CREATE	Adiciona um novo recurso
GET	READ	Acessa 1 ou vários recurso(s).
PUT	UPDATE	Atualiza um recurso
DELETE	DELETE	Remove um recurso

Métodos HTTP

List items

GET /items

Create an item

POST /items

Get an item

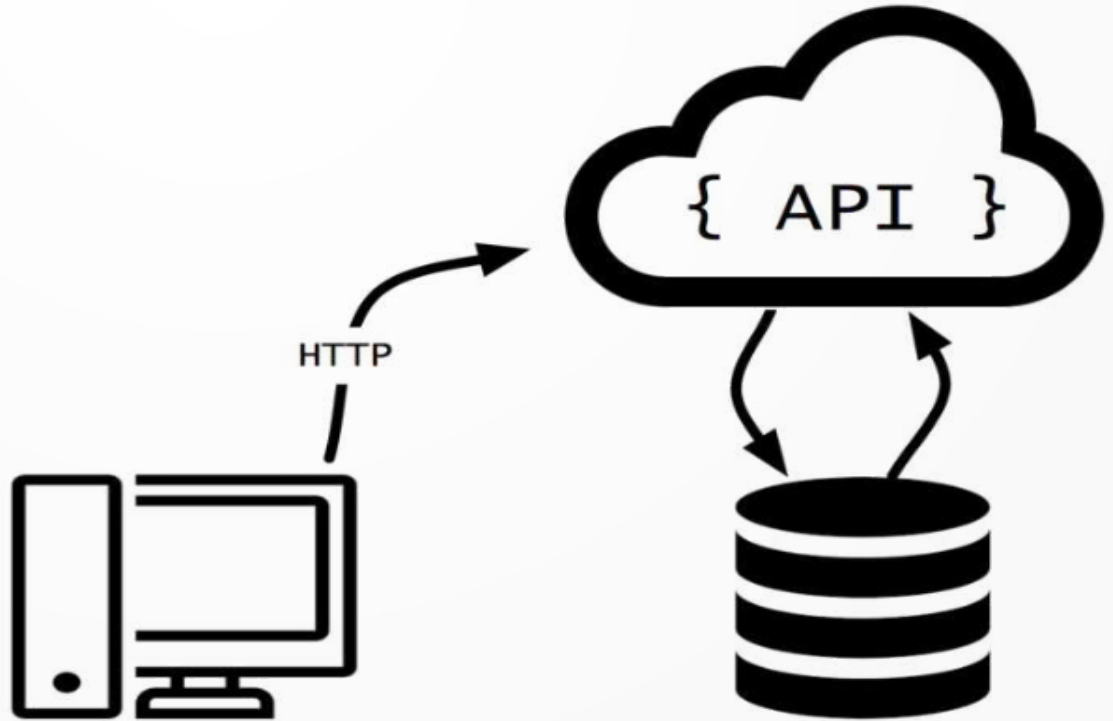
GET /items/{id}

Update an item

PUT /items/{id}

Delete an item

DELETE /items/{id}



Métodos HTTP

- Os métodos POST, GET, PUT e DELETE são normalmente usados em arquiteturas REST.

Método HTTP	URI	Descrição
POST	/cidades	Adiciona uma nova cidade
GET	/cidades	Retorna todas as cidades
GET	/cidades/{id}	Retorna a cidade de id = {id}
GET	/cidades/estados/{id}	Retorna as cidades do estado de id = {id}
PUT	/cidades/{id}	Atualiza a cidade de id = {id}
DELETE	/cidades/{id}	Remove a cidade de id = {id}

Negociação do conteúdo/representação

- Processo pelo qual o cliente determina a representação do recurso
 - Através do HTTP header
 - Accept: application/json
 - Através da extensão na URL (opcionalmente)
 - Não recomendado atualmente pela comunidade de desenvolvedores
 - `http://localhost:8080/restService/persons.json`

Código de status de resposta HTTP



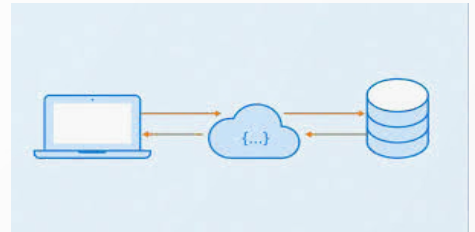
Lista: https://pt.wikipedia.org/wiki/Lista_de_c%C3%B3digos_de_estado_HTTP

Obs: Use substantivos e não verbos

Purpose	Method	Incorrect	Correct
Retrieves a list of users	GET	/getAllCars	/users
Create a new user	POST	/createUser	/users
Delete a user	DELETE	/deleteUser	/users/10
Get balance of user	GET	/getUserBalance	/users/11/balance

Não misture plural e singular. Mantenha simples e use apenas o plural para todos os recursos.

Spring MVC REST API



Spring MVC REST API

- Spring MVC é adequado para criar serviços web (REST API) porque é baseado no mapeamento de URLs a solicitações (controladores) e por poder atender com flexibilidade a diferentes tipos de conteúdo
- **Recursos** são **modelos (entidades)** associados a controladores REST (**@RestController**)
- Integra-se bem com uma série de serializadores, tanto para JSON (Jackson) quanto para XML (JAXB)

Aplicação completa

Demonstração 1

REST API (CRUD): Cidades & Estados

Link YouTube: <https://youtu.be/BsBwd5lh-8c>

CRUD: **C**reate, **R**ead, **U**ppdate & **D**elele

Métodos HTTP

- Os métodos POST, GET, PUT e DELETE são normalmente usados em arquiteturas REST.

Método HTTP	URI	Descrição
POST	/cidades	Adiciona uma nova cidade
GET	/cidades	Retorna todas as cidades
GET	/cidades/{id}	Retorna a cidade de id = {id}
GET	/cidades/estados/{id}	Retorna as cidades do estado de id = {id}
PUT	/cidades/{id}	Atualiza a cidade de id = {id}
DELETE	/cidades/{id}	Remove a cidade de id = {id}

Spring MVC: REST API

Método HTTP	URI	Descrição
POST	/cidades	Adiciona uma nova cidade

```
@RestController
public class CidadeRestController {

    @Autowired
    private ICidadeService service;

    private boolean isValid(String jsonInString) {}
    private void parse(Estado estado, JSONObject json) {}
    private void parse(Cidade cidade, JSONObject json) {}

    @PostMapping(path = "/cidades")
    @ResponseBody
    public ResponseEntity<Cidade> cria(@RequestBody JSONObject json) {
        try {
            if (isValid(json.toString())) {
                Cidade cidade = new Cidade();
                parse(cidade, json);
                service.save(cidade);
                return ResponseEntity.ok(cidade);
            } else {
                return ResponseEntity.badRequest().body(null);
            }
        } catch (Exception e) {
            e.printStackTrace();
            return ResponseEntity.status(HttpStatus.UNPROCESSABLE_ENTITY).body(null);
        }
    }

    public ResponseEntity<List<Cidade>> lista() {}
    public ResponseEntity<Cidade> lista(@PathVariable("id") long id) {}
    public ResponseEntity<List<Cidade>> listaPorEstado(@PathVariable("id") long id) {}
    public ResponseEntity<List<String>> listaPorNome(@RequestParam(name = "term") String nome) {}
    public ResponseEntity<Cidade> atualiza(@PathVariable("id") long id, @RequestBody JSONObject json) {}
    public ResponseEntity<Boolean> remove(@PathVariable("id") long id) {}
}
```

Spring MVC: REST API

Método HTTP	URI	Descrição
GET	/cidades	Retorna todas as cidades

```
@RestController
public class CidadeRestController {

    @Autowired
    private ICidadeService service;

    private boolean isValid(String jsonInString) {}
    private void parse(Estado estado, JSONObject json) {}
    private void parse(Cidade cidade, JSONObject json) {}

    public ResponseEntity<Cidade> cria(@RequestBody JSONObject json) {}

    @GetMapping(path = "/cidades")
    public ResponseEntity<List<Cidade>> lista() {
        List<Cidade> lista = service.findAll();
        if (lista.isEmpty()) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(lista);
    }

    public ResponseEntity<Cidade> lista(@PathVariable("id") long id) {}
    public ResponseEntity<List<Cidade>> listaPorEstado(@PathVariable("id") long id) {}
    public ResponseEntity<List<String>> listaPorNome(@RequestParam(name = "term") String nome) {}
    public ResponseEntity<Cidade> atualiza(@PathVariable("id") long id, @RequestBody JSONObject json) {}
    public ResponseEntity<Boolean> remove(@PathVariable("id") long id) {}
}
```

Spring MVC: REST API

Método HTTP	URI	Descrição
GET	/cidades/{id}	Retorna a cidade de id = {id}

```
@RestController
public class CidadeRestController {

    @Autowired
    private ICidadeService service;

    private boolean isJSONValid(String jsonInString) {}
    private void parse(Estado estado, JSONObject json) {}
    private void parse(Cidade cidade, JSONObject json) {}

    public ResponseEntity<Cidade> cria(@RequestBody JSONObject json) {}
    public ResponseEntity<List<Cidade>> lista() {}

    @GetMapping(path = "/cidades/{id}")
    public ResponseEntity<Cidade> lista(@PathVariable("id") long id) {
        Cidade cidade = service.findById(id);
        if (cidade == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(cidade);
    }

    public ResponseEntity<List<Cidade>> listaPorEstado(@PathVariable("id") long id) {}
    public ResponseEntity<List<String>> listaPorNome(@RequestParam(name = "term") String nome) {}
    public ResponseEntity<Cidade> atualiza(@PathVariable("id") long id, @RequestBody JSONObject json) {}
    public ResponseEntity<Boolean> remove(@PathVariable("id") long id) {}
}
```

Spring MVC: REST API

Método HTTP	URI	Descrição
GET	/cidades/estados/{id}	Retorna as cidades do estado de id = {id}

```
@RestController
public class CidadeRestController {

    @Autowired
    private ICidadeService service;

    private boolean isValid(String jsonString) {}
    private void parse(Estado estado, JSONObject json) {}
    private void parse(Cidade cidade, JSONObject json) {}

    public ResponseEntity<Cidade> cria(@RequestBody JSONObject json) {}
    public ResponseEntity<List<Cidade>> lista() {}
    public ResponseEntity<Cidade> lista(@PathVariable("id") long id) {}

    @GetMapping(path = "/cidades/estados/{id}")
    public ResponseEntity<List<Cidade>> listaPorEstado(@PathVariable("id") long id) {
        List<Cidade> lista = service.findByEstado(id);
        if (lista.isEmpty()) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(lista);
    }

    public ResponseEntity<List<String>> listaPorNome(@RequestParam(name = "term") String nome) {}
    public ResponseEntity<Cidade> atualiza(@PathVariable("id") long id, @RequestBody JSONObject json) {}
    public ResponseEntity<Boolean> remove(@PathVariable("id") long id) {}
}
```

Spring MVC: REST API

Método HTTP	URI	Descrição
PUT	/cidades/{id}	Atualiza a cidade de id = {id}

```
@RestController
public class CidadeRestController {

    @Autowired
    private ICidadeService service;

    private boolean isValid(String jsonInString) {}
    private void parse(Estado estado, JSONObject json) {}
    private void parse(Cidade cidade, JSONObject json) {}
    public ResponseEntity<Cidade> cria(@RequestBody JSONObject json) {}
    public ResponseEntity<List<Cidade>> lista() {}
    public ResponseEntity<Cidade> lista(@PathVariable("id") long id) {}
    public ResponseEntity<List<Cidade>> listaPorEstado(@PathVariable("id") long id) {}
    public ResponseEntity<List<String>> listaPorNome(@RequestParam(name = "term") String nome) {}

    @PutMapping(path = "/cidades/{id}")
    public ResponseEntity<Cidade> atualiza(@PathVariable("id") long id, @RequestBody JSONObject json) {
        try {
            if (isValid(json.toString())) {
                Cidade cidade = service.findById(id);
                if (cidade == null) {
                    return ResponseEntity.notFound().build();
                } else {
                    parse(cidade, json);
                    service.save(cidade);
                    return ResponseEntity.ok(cidade);
                }
            } else {
                return ResponseEntity.badRequest().body(null);
            }
        } catch (Exception e) { return ResponseEntity.status(HttpStatus.UNPROCESSABLE_ENTITY).body(null); }
    }

    public ResponseEntity<Boolean> remove(@PathVariable("id") long id) {}
}
```

Spring MVC: REST API

Método HTTP	URI	Descrição
DELETE	/cidades/{id}	Remove a cidade de id = {id}

```
@RestController
public class CidadeRestController {

    @Autowired
    private ICidadeService service;

    private boolean isValid(String jsonInString) {}
    private void parse(Estado estado, JSONObject json) {}
    private void parse(Cidade cidade, JSONObject json) {}
    public ResponseEntity<Cidade> cria(@RequestBody JSONObject json) {}
    public ResponseEntity<List<Cidade>> lista() {}
    public ResponseEntity<Cidade> lista(@PathVariable("id") long id) {}
    public ResponseEntity<List<Cidade>> listaPorEstado(@PathVariable("id") long id) {}
    public ResponseEntity<List<String>> listaPorNome(@RequestParam(name = "term") String nome) {}
    public ResponseEntity<Cidade> atualiza(@PathVariable("id") long id, @RequestBody JSONObject json) {}

    @DeleteMapping(path = "/cidades/{id}")
    public ResponseEntity<Boolean> remove(@PathVariable("id") long id) {

        Cidade cidade = service.findById(id);
        if (cidade == null) {
            return ResponseEntity.notFound().build();
        } else {
            service.delete(id);
            return ResponseEntity.noContent().build();
        }
    }
}
```


Aplicação completa

Demonstração 2

REST API (CRUD): Cidades & Estados

Cliente REST API – AJAX (Asynchronous Javascript and XML)

Link YouTube: <https://youtu.be/1hQcF2euuOI>

CRUD: **C**reate, **R**ead, **U**ppdate & **D**elele

Spring: Cliente API REST

- O acesso às REST APIs dentro de uma aplicação Spring envolve a utilização da classe RestTemplate

<https://docs.spring.io/spring-framework/docs/3.0.x/javadoc-api/org/springframework/web/client/RestTemplate.html>

[Overview](#) [Package](#) [Class](#) [Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

org.springframework.web.client
Class RestTemplate

java.lang.Object
└─ org.springframework.http.client.support.HttpAccessor
 └─ org.springframework.web.client.RestTemplate

All Implemented Interfaces:
[RestOperations](#)

public class **RestTemplate**
extends [HttpAccessor](#)
implements [RestOperations](#)

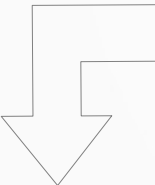
The central class for client-side HTTP access. It simplifies communication with HTTP servers, and enforces RESTful principles. It handles HTTP connections, leaving application code to provide URLs (with possible template variables) and extract results.

The main entry points of this template are the methods named after the six main HTTP methods:

HTTP method	RestTemplate methods
DELETE	<code>delete(java.lang.String, java.lang.Object...)</code>
GET	<code>getForObject(java.lang.String, java.lang.Class, java.lang.Object...)</code> <code>getForEntity(java.lang.String, java.lang.Class, java.lang.Object...)</code>
HEAD	<code>headForHeaders(java.lang.String, java.lang.Object...)</code>
OPTIONS	<code>optionsForAllow(java.lang.String, java.lang.Object...)</code>
POST	<code>postForLocation(java.lang.String, java.lang.Object, java.lang.Object...)</code> <code>postForObject(java.lang.String, java.lang.Object, java.lang.Class, java.lang.Object...)</code>
PUT	<code>put(java.lang.String, java.lang.Object, java.lang.Object...)</code>
any	<code>exchange(java.lang.String, org.springframework.http.HttpMethod, org.springframework.http.HttpEntity, java.lang.Class, java.lang.Object...)</code> <code>execute(java.lang.String, org.springframework.http.HttpMethod, org.springframework.web.client.RequestCallback, org.springframework.web.client.ResponseExtractor, java.lang.Object...)</code>

For each of these HTTP methods, there are three corresponding Java methods in the RestTemplate. Two variants take a String URI as first argument (eg. `getForObject(String, Class, Object[])`, `getForObject(String, Class, Map)`), and are capable of substituting any URI templates in that URL using either a String variable arguments array, or a Map<String, String>. The String varargs variant expands the given template variables in order, so that

Spring: Cliente REST API



```
{  
  "id": 4601,  
  "nome": "São Carlos",  
  "estado": { "id": 26, "sigla": "SP", "nome": "São Paulo" }  
}
```

```
public class Cidade {  
  
    private Long id;  
    private String nome;  
    private Estado estado;  
  
    public Cidade() {}  
    public Cidade(String nome, Estado estado) {}  
    public Long getId() {}  
    public void setId(Long id) {}  
    public String getNome() {}  
    public void setNome(String nome) {}  
    public Estado getEstado() {}  
    public void setEstado(Estado estado) {}  
  
    @Override  
    public String toString() {  
        return nome + "/" + estado.getSigla();  
    }  
}
```



```
public class Estado {  
  
    private Long id;  
    private String sigla;  
    private String nome;  
  
    public Estado() {}  
    public Estado(Long id, String sigla, String nome) {}  
    public Long getId() {}  
    public void setId(Long id) {}  
    public String getSigla() {}  
    public void setSigla(String sigla) {}  
    public String getNome() {}  
    public void setNome(String nome) {}  
  
    @Override  
    public String toString() {  
        return nome + " (" + sigla + ")";  
    }  
}
```

Spring: Cliente REST API

Método HTTP	URI	Descrição
POST	/cidades	Adiciona uma nova cidade

```
@Service
public class RestClientService implements IRestClientService {

    @Autowired
    private RestTemplate restTemplate;

    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        return builder.build();
    }

    @Override
    public Long create(Cidade cidade) {
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
        HttpEntity<Cidade> entity = new HttpEntity<Cidade>(cidade, headers);
        String url = "http://localhost:8080/cidades";
        ResponseEntity<Cidade> res = restTemplate.postForEntity(url, entity, Cidade.class);
        Cidade c = res.getBody();

        return c.getId();
    }

    public List<Cidade> get() {}
    public List<Cidade> get(Estado estado) {}
    public Cidade get(Long id) {}
    public boolean update(Cidade cidade) {}
    public boolean delete(Long id) {}
}
```

Spring: Cliente REST API

Método HTTP	URI	Descrição
GET	/cidades	Retorna todas as cidades

```
@Service
public class RestClientService implements IRestClientService {

    @Autowired
    private RestTemplate restTemplate;

    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        return builder.build();
    }

    public Long create(Cidade cidade) {

    }

    @Override
    public List<Cidade> get() {
        String url = "http://localhost:8080/cidades";
        Cidade[] cidades = restTemplate.getForObject(url, Cidade[].class);
        return Arrays.asList(cidades);
    }

    public List<Cidade> get(Estado estado) {
    }
    public Cidade get(Long id) {
    }
    public boolean update(Cidade cidade) {
    }
    public boolean delete(Long id) {
    }
}
```

Spring: Cliente REST API

Método HTTP	URI	Descrição
GET	/cidades/{id}	Retorna a cidade de id = {id}

```
@Service
public class RestClientService implements IRestClientService {

    @Autowired
    private RestTemplate restTemplate;

    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        return builder.build();
    }

    public Long create(Cidade cidade) { }
    public List<Cidade> get() { }
    public List<Cidade> get(Estado estado) { }

    public Cidade get(Long id) {
        String url = "http://localhost:8080/cidades/" + id;
        return restTemplate.getForObject(url, Cidade.class);
    }

    public boolean update(Cidade cidade) { }
    public boolean delete(Long id) { }
}
```

Spring: Cliente REST API

Método HTTP	URI	Descrição
GET	/cidades/estados/{id}	Retorna as cidades do estado de id = {id}

```
@Service
public class RestClientService implements IRestClientService {

    @Autowired
    private RestTemplate restTemplate;

    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        return builder.build();
    }

    public Long create(Cidade cidade) {}
    public List<Cidade> get() {}

    @Override
    public List<Cidade> get(Estado estado) {
        String url = "http://localhost:8080/cidades/estados/" + estado.getId();
        Cidade[] cidades = restTemplate.getForObject(url, Cidade[].class);
        return Arrays.asList(cidades);
    }

    public Cidade get(Long id) {}
    public boolean update(Cidade cidade) {}
    public boolean delete(Long id) {}
}
```

Spring: Cliente REST API

Método HTTP	URI	Descrição
PUT	/cidades/{id}	Atualiza a cidade de id = {id}

```
@Service
public class RestClientService implements IRestClientService {

    @Autowired
    private RestTemplate restTemplate;

    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        return builder.build();
    }

    public Long create(Cidade cidade) {}
    public List<Cidade> get() {}
    public List<Cidade> get(Estado estado) {}
    public Cidade get(Long id) {}

    public boolean update(Cidade cidade) {
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
        HttpEntity<Cidade> entity = new HttpEntity<Cidade>(cidade, headers);
        String url = "http://localhost:8080/cidades/" + cidade.getId();
        ResponseEntity<Cidade> res = restTemplate.exchange(url, HttpMethod.PUT,
                                                            entity, Cidade.class);

        return res.getStatusCode() == HttpStatus.OK;
    }

    public boolean delete(Long id) {}
}
```

Spring: Cliente REST API

Método HTTP	URI	Descrição
DELETE	/cidades/{id}	Remove a cidade de id = {id}

```
@Service
public class RestClientService implements IRestClientService {

    @Autowired
    private RestTemplate restTemplate;

    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        return builder.build();
    }

    public Long create(Cidade cidade) {}
    public List<Cidade> get() {}
    public List<Cidade> get(Estado estado) {}
    public Cidade get(Long id) {}
    public boolean update(Cidade cidade) {}

    public boolean delete(Long id) {
        try {
            String url = "http://localhost:8080/cidades/" + id;
            restTemplate.delete(url);
            return true;
        } catch (RestClientException e) {
            return false;
        }
    }
}
```

Aplicação completa

Demonstração 3

REST API (CRUD): Transações (Cartão de Crédito)

Link YouTube: <https://youtu.be/YSB4IBw-uAc>

CRUD: **C**reate, **R**ead, **U**ppdate & **D**elele

Aplicação completa

Demonstração 4

Livraria Virtual - Cliente do REST API (Transações)

Link YouTube: <https://youtu.be/7oXUe7pqty8>

CRUD: **C**reate, **R**ead, **U**ppdate & **D**elele

FIM