

19.1. Introdução à Discoverability e HATEOAS

quarta-feira, 12 de abril de 2023

14:05



Discoverability e HATEOAS

- Discoverability
 - Capacidade que a api dá aos consumidores de navegar entre os recursos sem conhecer previamente os endpoints/api
 - Diferente de HETEOAS, pois é somente um conceito de que sua api precisa ser descoberta
- HATEOAS
 - componente do REST
 - Permite o Discoverability por meio de inserção de hypermedia nas respostas (links de recursos dentro de recursos)
 - Consumir a api por meio de próximos links disponibilizados nos recursos para se guiar pela api
- [4.39. Conhecendo o nível 3 do RMM](#)

Nota: APIS sem HOTEAS não é REST de fato, porém, o mercado e a comunidade chamam apis de rest mesmo sem HOTEAS

19.2. Adicionando a URI do recurso criado no header da resposta

quarta-feira, 12 de abril de 2023 14:17

- Descoberta de serviço rest

Mesmo sem implementar HATEOAS, é uma boa prática durante o POST de um recurso, adicionar no header da response um Location com a uri do recurso criado, pois é custoso para o consumidor da api montar uma nova uri com o novo recurso criado

```
@Override
@PostMapping
public ResponseEntity<CidadeDTO> adicionar(@RequestBody @Valid CidadeInputDTO cidadeInputDTO) {
    try {
        final Cidade cidade = cidadeInputDisassembler.DTotoDomainModel(cidadeInputDTO);
        final CidadeDTO cidadeSalva = cAssembler.toDTO(cidadeService.salvar(cidade));

        /*Monta uma uri vinda da request. URI do recurso criado*/
        final URI uri = ServletUriComponentsBuilder.fromCurrentRequestUri()
            .path("/{cidadeId}")
            .buildAndExpand(cidadeSalva.getId()).toUri();

        /*Classe Holder para expor a solicitação da Web na forma de um objeto
        RequestAttributes associado a thread*/
        final ServletRequestAttributes requestAttributes =
            ((ServletRequestAttributes) RequestContextHolder.getRequestAttributes());

        final HttpServletResponse response = requestAttributes.getResponse();

        /*Header enviado na resposta com a uri*/
        response.addHeader(HttpHeaders.LOCATION, uri.toString());

        return ResponseEntity.status(HttpStatus.CREATED).body(cidadeSalva);
    } catch (EstadoNaoEncontradoException e) {
        throw new NegocioException(e.getMessage(), e);
    }
}
```

- Abstrair o trecho do código para uma classe utilitária

```
package com.algaworks.algafood.api;

import ...

2 usages
@UtilityClass
public class ResourceUriHelper {

    1 usage
    public static void addUriInResponseHeader(Object resourceId){

        /*Monta uma uri vinda da request. URI do recurso criado*/
        final URI uri = ServletUriComponentsBuilder.fromCurrentRequestUri()
            .path("/{cidadeId}")
            .buildAndExpand(resourceId).toUri();

        /*Classe Holder para expor a solicitação da Web na forma de um objeto
        RequestAttributes associado a thread*/
        final ServletRequestAttributes requestAttributes =
            ((ServletRequestAttributes) RequestContextHolder.getRequestAttributes());

        final HttpServletResponse response = requestAttributes.getResponse();

        /*Header enviado na resposta com a uri*/
        response.addHeader(HttpHeaders.LOCATION, uri.toString());
    }
}
```

Vary	①	Access-Control-Request-Method
Vary	①	Access-Control-Request-Headers
Location	①	http://localhost:8080/cidades/6
Content-Type	①	application/json

19.3. Adicionando o Spring HATEOAS no projeto

quarta-feira, 12 de abril de 2023 15:15

```
<!-- HATEOAS -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-hateoas</artifactId>
    </dependency>
</dependency>
```

19.4. Atualizando o projeto para Spring Boot 2.2 (Spring HATEOAS 1.0)

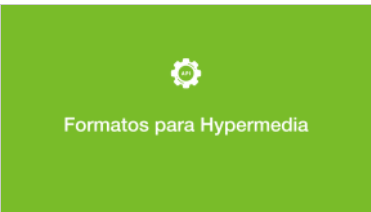
quarta-feira, 12 de abril de 2023

15:20

- Versão atual do Spring Boot 2.7.8
- Versão do HATEOAS 2.0.3

19.5. Resolvendo conflito de dependências com Spring HATEOAS e SpringFox

quarta-feira, 12 de abril de 2023 15:27



Hypermedia

```
<html>
  <body>
    <a href="/produtos/macbook.html">Macbook Pro 13</a>
    <a href="/carrinho.html">Carrinho</a>
  </body>
</html>
```

HAL

The Hypertext Application Lan

x

+

←

→

↺

🔒 Not Secure

stateless.co/hal_specification.html

☆

🔍

🌐

⋮

HAL - Hypertext Application Language

A lean hypermedia type

- Author: Mike Kelly <mike@stateless.co>
- Created: 2011-06-13
- Updated: 2013-09-18 (Updated)

Summary

HAL is a simple format that gives a consistent and easy way to hyperlink between resources in your API.

Adopting HAL will make your API explorable, and its documentation easily discoverable from within the API itself. In short, it will make your API easier to work with and therefore more attractive to client developers.

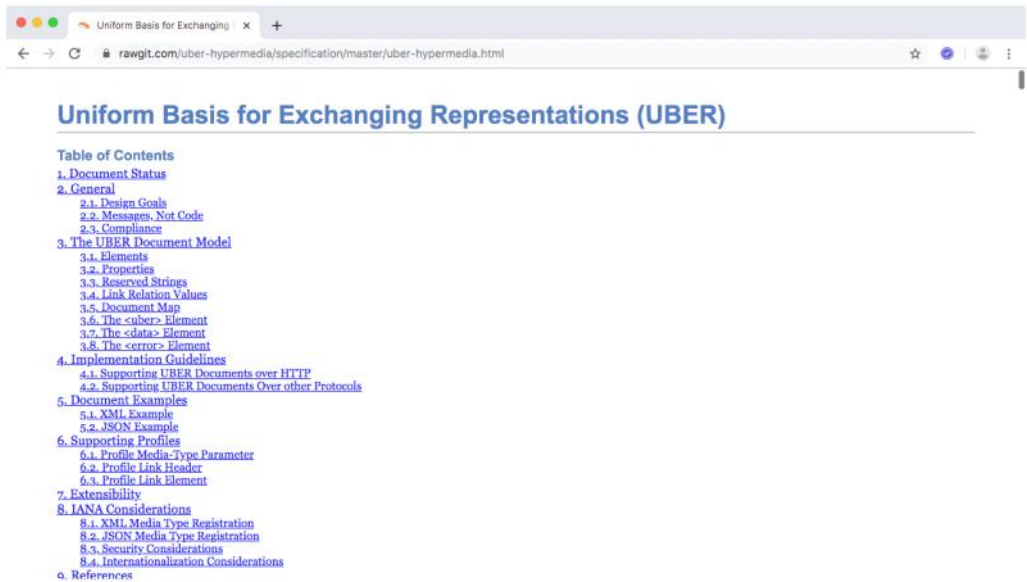
APIs that adopt HAL can be easily served and consumed using open source libraries available for most major programming languages. It's also simple enough that you can just deal with it as you would any other JSON.

About The Author

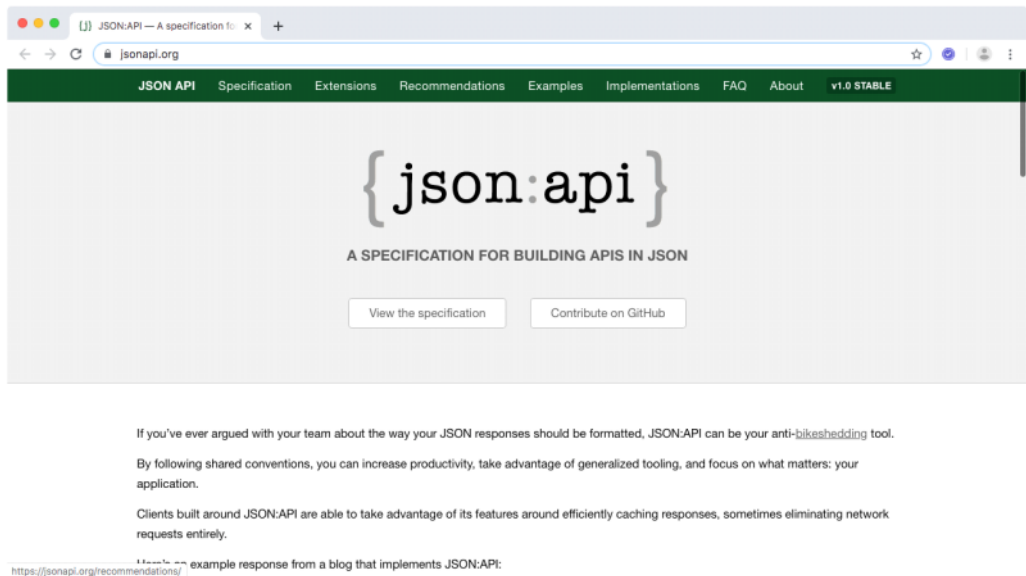
Mike Kelly is a software engineer from the UK. He runs an [API consultancy](#) helping companies design and build beautiful APIs that developers love.

```
{
  "id": 2,
  "nome": "Uberlândia",
  "estado": {
    "id": 3,
    "nome": "Minas Gerais",
    "_links": {
      "self": "https://api.algafood.com.br/estados/3",
      "estados": "https://api.algafood.com.br/estados"
    }
  },
  "_links": {
    "self": "https://api.algafood.com.br/cidades/2",
    "cidades": "https://api.algafood.com.br/cidades"
  }
}
```

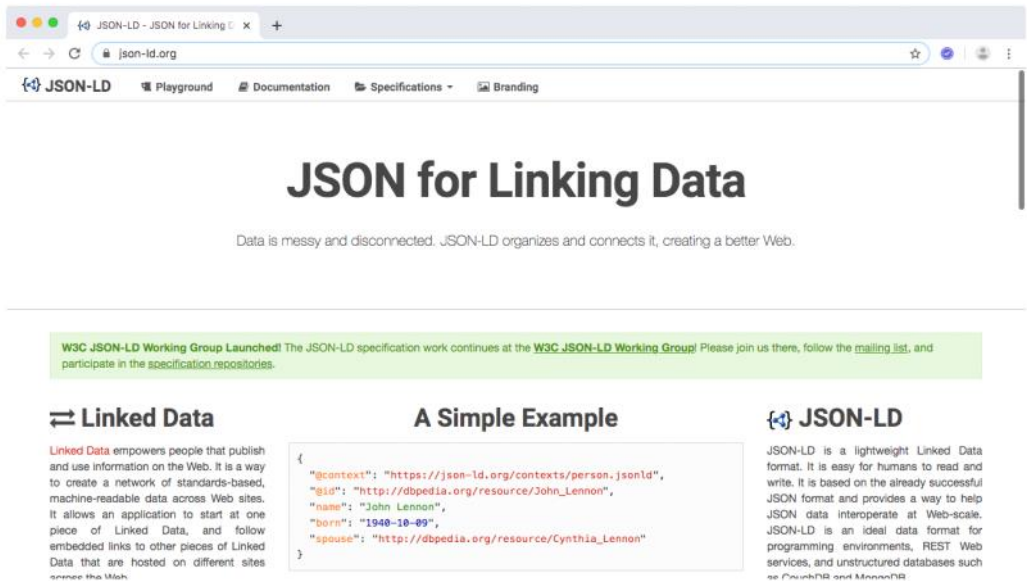
UBER



JSON:API



JSON-LD




Collection+JSON

Collection+JSON - Document

Collection+JSON - Document

amundsen.com/media-types/collection/format/

There's an alien on your Mac



Delete with CleanMyMac X

Media Types

[Home](#) » [Collection+JSON](#) » [Format](#)

Collection+JSON - Document Format

Description
[Collection+JSON](#) is a JSON-based read/write hypermedia-type designed to support management and querying of simple collections.

Author:
Mike Amundsen (mamund@yahoo.com)

Dates:
2011-05-04 (Created)
2013-02-24 (Updated)

Status:
Approved

Contents

- [1. Profiles](#)
- [2. General Concepts](#)
- [3. Objects](#)
- [4. Arrays](#)
- [5. Properties](#)
- [6. Link Relations](#)
- [7. Data Types](#)
- [8. Extensibility](#)

Siren

GitHub - kevinswiber/siren: Siren

github.com/kevinswiber/siren

Siren: a hypermedia specification for representing entities

DOI: 10.5281/zenodo.336763

Your input is appreciated. Feel free to file a GitHub Issue, a Pull Request, or contact us. Thank you!

- [Official Siren Google Group](#)
- Kevin on Twitter [@kevinswiber](#)

Example

Below is a JSON Siren example of an order, including sub-entities. The first sub-entity, a collection of items associated with the order, is an embedded link. Clients may choose to automatically resolve linked sub-entities. The second sub-entity is an embedded representation of customer information associated with the order. The example also includes an action to add items to the order and a set of links to navigate through a list of orders.

The media type for JSON Siren is `application/vnd.siren+json`.

```
{
  "class": [ "order" ],
  "properties": {
    "orderNumber": 42,
    "itemCount": 3,
    "status": "pending"
  },
  "entities": [
    {
```

GitHub - kevinswiber/siren: Siren

github.com/kevinswiber/siren

19.7. Adicionando hypermedia na representação de recurso único com HAL

quarta-feira, 12 de abril de 2023

15:48

Adicionar hypermedia na resposta, no formato HAL, na representação única de cidade

AlgaFood / Cidade / Cidade - adicionar

POST ▼ http://localhost:8080/cidades

Params Authorization Headers (9) **Body** ● Pre-req

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw

```
1 {
2   "nome": "Bragança",
3   "estado": {
4     "id": 1
5   }
6 }
```

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1 {
2   "id": 6,
3   "nome": "Bragança",
4   "estado": {
5     "id": 1,
6     "nome": "Minas Gerais"
7   }
8 }
```

- RepresentationModel
 - Container para adicionar links para outros recursos
 - Disponibiliza métodos para adicionar hypermedia do HATEOAS como links

```
package com.algaworks.algafood.api.model.dto;

import ...

15 usages
@Getter
@Setter
public class CidadeDTO extends RepresentationModel {
    RepresentationModel<T> org.springframework.hateoas
```

```

@Getter
@Setter
public class CidadeDTO extends RepresentationModel<CidadeDTO> {

```

```

@Override
@GetMapping("/{cidadeId}")
public CidadeDTO buscar(@PathVariable Long cidadeId) {

    final CidadeDTO cidade = cAssembler.toDTO(cidadeService.buscar(cidadeId));

    cidade.add(Link.of( href: "http://localhost:8080/cidades/1"));
    cidade.add(Link.of("http://localhost:8080/cidades/1", IanaLinkRelations.SELF));

    cidade.add(Link.of("http://localhost:8080/cidades", IanaLinkRelations.COLLECTION));
    cidade.add(Link.of( href: "http://localhost:8080/cidades", relation: "cidades"));

    cidade.getEstado().add(Link.of( href: "http://localhost:8080/estados/1"));

    return cidade;
}

```

Foi adicionado um representation model no dto de estado

- Link Relations
 - Uma relação de link é um atributo descritivo anexado a um hiperlink para definir o tipo de link ou o relacionamento entre os recursos de origem e destino
 - <https://www.iana.org/assignments/link-relations/link-relations.xhtml>

GET ⌵ http://localhost:8080/cidades/4

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (9) Test Results 🌐 Status: 200 OK Time

Pretty Raw Preview Visualize JSON ⌵ ≡

```

1  {
2    "id": 4,
3    "nome": "Campinas",
4    "estado": {
5      "id": 2,
6      "nome": "São Paulo",
7      "_links": {
8        "self": {
9          "href": "http://localhost:8080/estados/1"
10       }
11     }
12   },
13   "_links": {
14     "self": {
15       "href": "http://localhost:8080/cidades/1"
16     },
17     "cidades": {
18       "href": "http://localhost:8080/cidades"

```

o nome do link relations pode ser atribuído no segundo argumento do método Link.ok para o método add

19.8. Construindo links dinâmicos com WebMvcLinkBuilder

quarta-feira, 12 de abril de 2023 17:11

- Alterar hypermedia hardcoded, como na aula anterior

19.7. Adicionando hypermedia na representação de recurso único com HAL

e tornar o HATEOAS dinâmico, conseguindo capturar host, domínio, porta e etc, incluindo quando a aplicação estiver na nuvem.

- Utilizando WebMvcLinkBuilder

```
import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.linkTo;

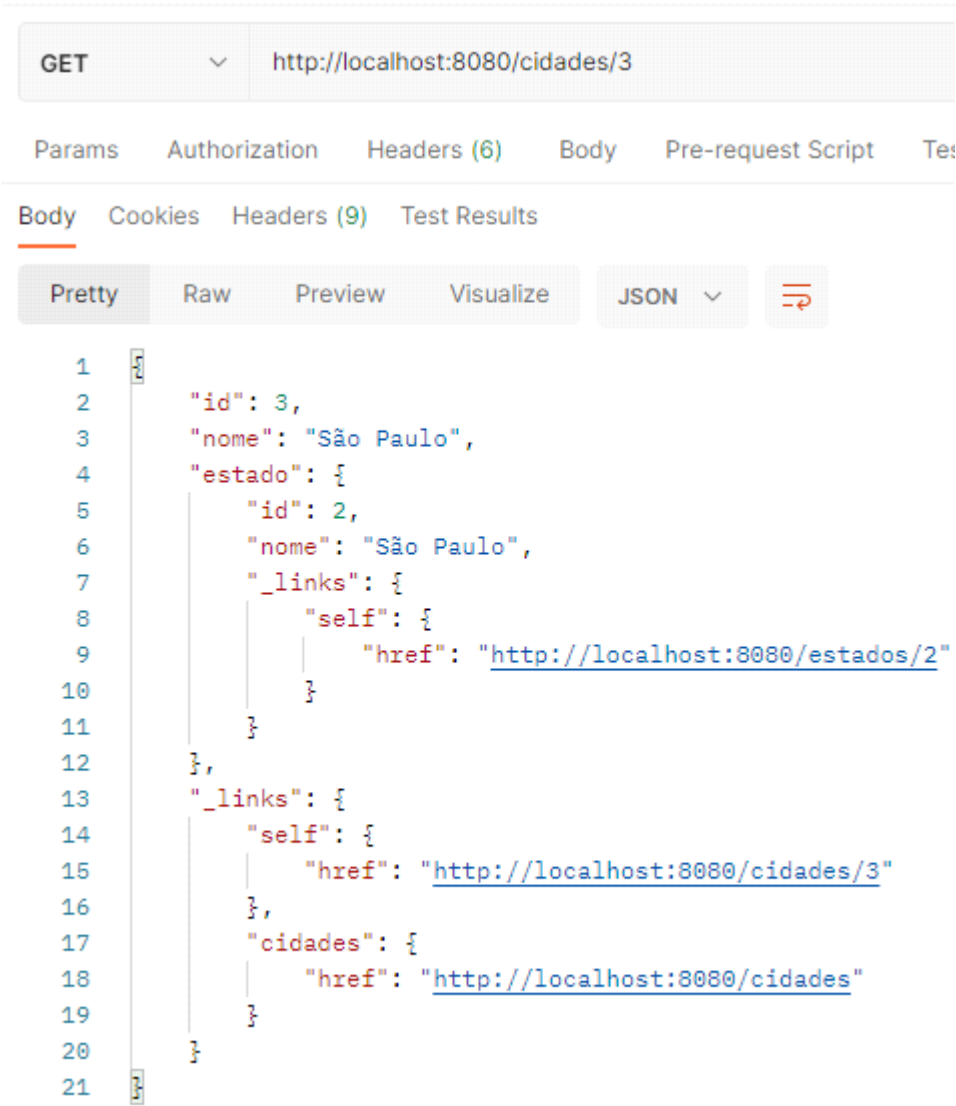
@Override
@GetMapping("/{cidadeId}")
public CidadeDTO buscar(@PathVariable Long cidadeId) {
    final CidadeDTO cidade = cAssembler.toDTO(cidadeService.buscar(cidadeId));

    /*_link.self para o próprio recurso*/
    cidade.add(linkTo(CidadeController.class)
        .slash(cidade.getId()).withSelfRel());

    /*_link.collection para coleção de cidades */
    cidade.add(linkTo(CidadeController.class)
        .withRel("cidades"));

    /*_link.self para o recurso estado/id dentro de cidade*/
    cidade.getEstado().add(linkTo(EstadoController.class)
        .slash(cidade.getEstado().getId()).withSelfRel());

    return cidade;
}
```



19.9. Construindo links que apontam para métodos

quarta-feira, 12 de abril de 2023 19:35

```
@Override
@GetMapping("/{cidadeId}")
public CidadeDTO buscar(@PathVariable Long cidadeId) {
    final CidadeDTO cidade = cAssembler.toDTO(cidadeService.buscar(cidadeId));

    /*_link.self para o próprio recurso
    cidade.add(LinkTo(CidadeController.class)
        .slash(cidade.getId()).withSelfRel());*/

    /*_link.self para o próprio recurso
    linkTo recebendo um Object method como argumento
    chamada de método para controller específicos*/
    final Link linkCidade = linkTo( methodOn(CidadeController.class).buscar(cidade.getId()) ).withSelfRel();
    cidade.add(linkCidade);

    /*_link.collection para coleção de cidades
    cidade.add(LinkTo(CidadeController.class)
        .withRel("cidades"));*/

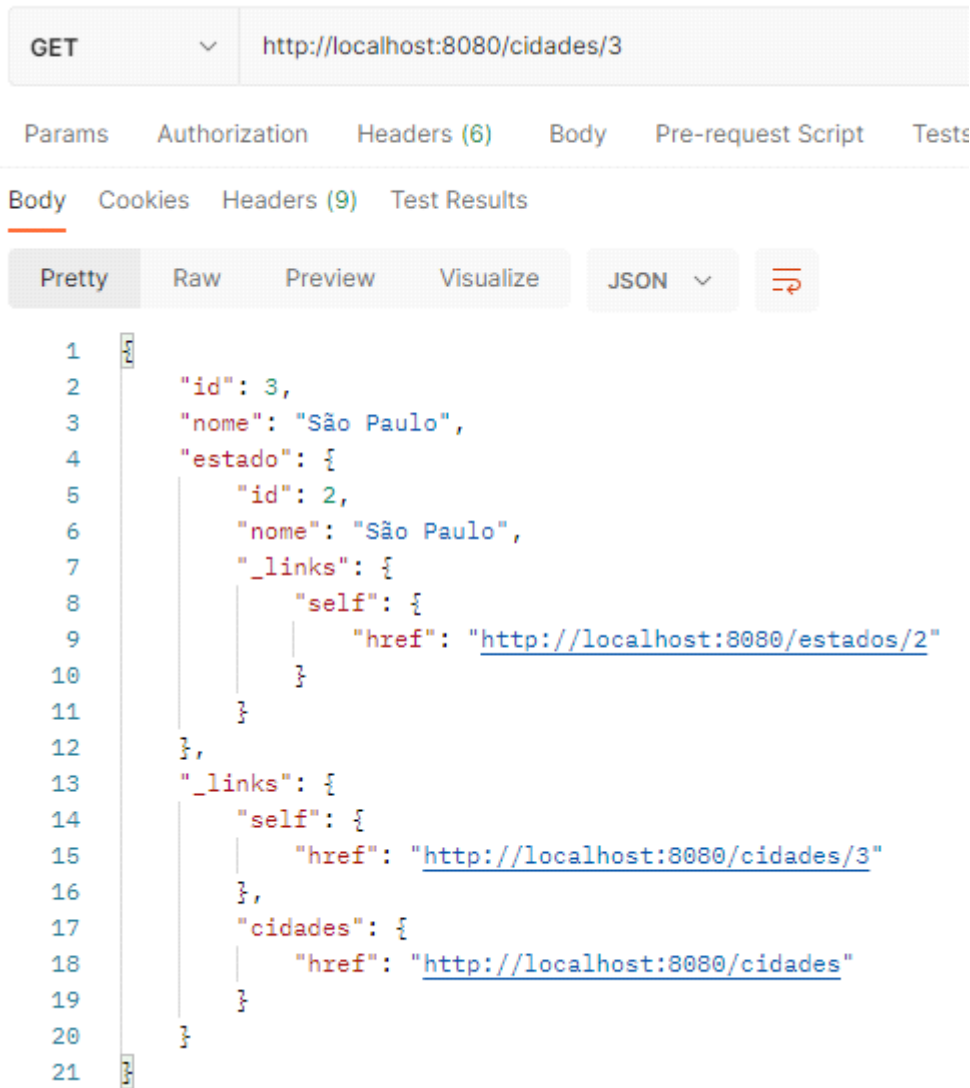
    final Link linkCidades = linkTo(methodOn(CidadeController.class).listar()).withRel("Cidades");
    cidade.add(linkCidades);

    /*_link.self para o recurso estado/id dentro de cidade
    cidade.getEstado().add(LinkTo(EstadoController.class)
        .slash(cidade.getEstado().getId()).withSelfRel());*/

    final Link linkEstados = linkTo(methodOn(EstadoController.class).
        buscar(cidade.getEstado().getId())).withSelfRel();
    cidade.getEstado().add(linkEstados);

    return cidade;
}
```

linkTo recebe um tipo Object auxiliado na chamada do método MethodOn, onde chamamos o método do controlador que queremos para ser o hypermedia



19.10. Adicionando hypermedia na representação de recursos de coleção

quarta-feira, 12 de abril de 2023

20:24

- Tratar arrays vazios na listagem de coleção de recursos

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/cidades`. The 'Query Params' section is empty. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response is an array with one object:

```
1 [
2   {
3     "id": 1,
4     "nome": "Uberlândia",
5     "estado": {
6       "id": 1,
7       "nome": "Minas Gerais",
8       "links": []
9     },
10    "links": []
11  },
12  ...
```

Estamos retornando um DTO de um recurso que estende (é um) `RepresentationModel` [19.7. Adicionando hypermedia na representação de recurso único com HAL](#) e não podemos retornar uma lista de `RepresentationModel`. Podemos contornar isso retornando um tipo `CollectionModel` tipado com o modelo de representação, ou seja, o `CollectionModel` é um wrapper de `RepresentationModel`

```

@Override
@GetMapping
public CollectionModel<CidadeDTO> listar() {
    final List<Cidade> cidades = cidadeService.listar();
    final List<CidadeDTO> cidadesDTOS = cAssembler.toListDTO(cidades);

    return CollectionModel.of(cidadesDTOS);
}

```

GET
http://localhost:8080/cidades

Params
Authorization
Headers (6)
Body
Pre-req

Query Params

Key	Value
Key	Value

Body
Cookies
Headers (9)
Test Results

Pretty
Raw
Preview
Visualize
JSON

```

1  {
2    "_embedded": {
3      "cidadeDTOList": [
4        {
5          "id": 1,
6          "nome": "Uberlândia",
7          "estado": {
8            "id": 1,
9            "nome": "Minas Gerais"
10         }
11       },
12       {
13         "id": 2,
14         "nome": "Belo Horizonte",
15         "estado": {
16           "id": 1,
17           "nome": "Minas Gerais"
18         }
19       }
20     ]
21   }
22 }

```

Podemos adicionar links na coleção toda ou em cada objeto json

```

@Override
@GetMapping
public CollectionModel<CidadeDTO> listar() {
    final List<Cidade> cidades = cidadeService.listar();
    final List<CidadeDTO> cidadesDTOS = cAssembler.toListDTO(cidades);

    /*Para criar links para cada objeto json*/
    cidadesDTOS.forEach(cidade -> {
        final Link linkEstados = linkTo(methodOn(EstadoController.class).
            buscar(cidade.getEstado().getId())).withSelfRel();
        cidade.getEstado().add(linkEstados);

        final Link linkCidades = linkTo(methodOn(CidadeController.class).listar()).withRel("Cidades");
        cidade.add(linkCidades);

        final Link linkCidade = linkTo( methodOn(CidadeController.class).buscar(cidade.getId()) ).withSelfRel();
        cidade.add(linkCidade);
    });

    final CollectionModel<CidadeDTO> cidadesCollectionModel = CollectionModel.of(cidadesDTOS);

    /*link para listagem de cidades*/
    cidadesCollectionModel.add(linkTo(CidadeController.class).withSelfRel());

    return cidadesCollectionModel;
}

```

GET
▼
http://localhost:8080/cidades

Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings

Body
Cookies
Headers (9)
Test Results
Status: 200

Pretty
Raw
Preview
Visualize
JSON ▼

```

1  {
2    "_embedded": {
3      "cidadeDTOList": [
4        {
5          "id": 1,
6          "nome": "Uberlândia",
7          "estado": {
8            "id": 1,
9            "nome": "Minas Gerais",
10           "_links": {
11             "self": {
12               "href": "http://localhost:8080/estados/1"
13             }
14           }
15         },
16         "_links": {
17           "Cidades": {
18             "href": "http://localhost:8080/cidades"
19           },
20           "self": {
21             "href": "http://localhost:8080/cidades/1"
22           }
23         }
24       ]
25     }
26   }

```

Para alterar o nome "cidadeDTOList" usar a anotação @Relations contendo

collectionRelation = "cidades"

```
package com.algaworks.algafood.api.model.dto;

import ...

19 //usages
@Relation(collectionRelation = "cidades")
@Getter
@Setter
public class CidadeDTO extends RepresentationModel<CidadeDTO> {
```

19.11. Montando modelo de representação com RepresentationModelAssembler

quarta-feira, 12 de abril de 2023

22:06

- O Spring HATEOAS possui suporte para conversão de entidades de domínio para modelos de representações usando `RepresentationModelAssemblerSupport<Entidade, Dto>`
- O Spring HATEOAS também utiliza o termo "Model" em vez de "DTO"

```
@Component
public class CidadeAssembler extends RepresentationModelAssemblerSupport<Cidade, CidadeDTO> {

    1 usage
    @Autowired
    IMapper mapper;

    public CidadeAssembler() { super(CidadeController.class, CidadeDTO.class); }

    public CidadeAssembler(Class<?> controllerClass, Class<CidadeDTO> resourceType) {
        super(controllerClass, resourceType);
    }

    @Override
    public CidadeDTO toModel(Cidade cidade) { return mapper.map(cidade, CidadeDTO.class); }

    public List<CidadeDTO> toListDTO(List<Cidade> cidades) { return cidades.stream().map(this::toModel).toList(); }
}
```

Para os tipos parametrizados, utilizamos classe de origem e destino

- Na versão da aula, era obrigado a utilizar o construtor, porém, na versão atual (12/04) não é necessário, mas vamos colocar por garantia
- Os argumentos para o construtor da superclasse é a classe controladora da origem, e a classe de destino (dto)
- Não precisa do construtor da classe com argumentos

```
public CidadeAssembler(){
    super(CidadeController.class, CidadeDTO.class);
}
```

- Sobrescrever o método toModel

```
@Override
public CidadeDTO toModel(Cidade cidade){
    return mapper.map(cidade, CidadeDTO.class);
}
```

- Utilizar Dto para a adição de links para hypermedia

```

@Override
public CidadeDTO toModel(Cidade cidade){
    final CidadeDTO cidadeDTO = modelMapper.map(cidade, CidadeDTO.class);

    /*_link.self para o próprio recurso
    cidadeDTO.add(LinkTo(CidadeController.class)
        .slash(cidadeDTO.getId()).withSelfRel());*/

    /*_link.self para o próprio recurso
    linkTo recebendo um Object method como argumento
    chamada de método para controller específicos*/
    final Link linkCidade = linkTo( methodOn(CidadeController.class).buscar(cidadeDTO.getId()) ).withSelfRel();
    cidadeDTO.add(linkCidade);

    /*_link.collection para coleção de cidades
    cidadeDTO.add(LinkTo(CidadeController.class)
        .withRel("cidades"));*/

    final Link linkCidades = linkTo(methodOn(CidadeController.class).listar()).withRel("Cidades");
    cidadeDTO.add(linkCidades);

    /*_link.self para o recurso estado/id dentro de cidadeDTO
    cidadeDTO.getEstado().add(LinkTo(EstadoController.class)
        .slash(cidadeDTO.getEstado().getId()).withSelfRel());*/

    final Link linkEstados = linkTo(methodOn(EstadoController.class).
        buscar(cidadeDTO.getEstado().getId())).withSelfRel();
    cidadeDTO.getEstado().add(linkEstados);

    return cidadeDTO;
}

```

e podemos limpar o controlador. Agora cada dto de cidade possui links hipermidia. Dentro de cada Cidade temos um Estado que também possui link hipermidia

```

@Override
@GetMapping
public CollectionModel<CidadeDTO> listar() {
    final List<Cidade> cidades = cidadeService.listar();
    final List<CidadeDTO> cidadesDTOS = cAssembler.toListDTO(cidades);

    return CollectionModel.of(cidadesDTOS);
}

@Override
@GetMapping("/{cidadeId}")
public CidadeDTO buscar(@PathVariable Long cidadeId) {
    return cAssembler.toModel(cidadeService.buscar(cidadeId));
}

```

A classe RepresentationModelAssemblerSupport também oferece a conversão de coleção de entidades para coleção de dtos usando toCollectionModel. Podemos excluir o método do nosso assembler e usar o método da classe (implementa interface)

```

@Override
@GetMapping
public CollectionModel<CidadeDTO> listar() {
    final List<Cidade> cidades = cidadeService.listar();
    final List<CidadeDTO> cidadesDTOS = cAssembler.toListDTO(cidades);

    return CollectionModel.of(cidadesDTOS);
}

```

```

@Override
@GetMapping
public CollectionModel<CidadeDTO> listar() {
    final List<Cidade> cidades = cidadeService.listar();
    final CollectionModel<CidadeDTO> cidadeDTOS = cAssembler.toCollectionModel(cidades);

    return cidadeDTOS;
}

```

Representação:

GET ⌵ http://localhost:8080/cidades

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
-----	-------

Body Cookies Headers (9) Test Results ⌕ Status: 200

Pretty Raw Preview Visualize JSON ⌵ ↻

```
2  _embedded: {
3    "cidades": [
4      {
5        "id": 1,
6        "nome": "Uberlândia",
7        "estado": {
8          "id": 1,
9          "nome": "Minas Gerais",
10         "_links": {
11           "self": {
12             "href": "http://localhost:8080/estados/1"
13           }
14         }
15       },
16       "_links": {
17         "self": {
18           "href": "http://localhost:8080/cidades/1"
19         },
20         "Cidades": {
21           "href": "http://localhost:8080/cidades"
22         }
23       }
24     ]
25   }
26 }
```

O hypermedia para o próprio objeto não existe ainda

GET <http://localhost:8080/cidades>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
-----	-------

Body Cookies Headers (9) Test Results ⌚ Status: 200

Pretty Raw Preview Visualize JSON ↕

```

92      "id": 3,
93      "nome": "Ceará",
94      "_links": {
95        "self": {
96          "href": "http://localhost:8080/estados/3"
97        }
98      },
99    },
100    "_links": {
101      "self": {
102        "href": "http://localhost:8080/cidades/5"
103      },
104      "Cidades": {
105        "href": "http://localhost:8080/cidades"
106      }
107    }
108  }
109 ]
110 }
111 }
```

pois não implementamos o link para a coleção em si, apenas para os objetos dentro. Podemos adicionar um link na coleção:

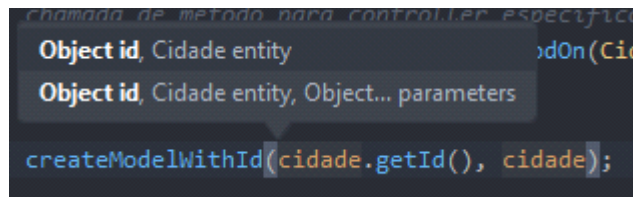
Deletamos o método para conversão de lista de entidade para lista de dto, mas vamos sobrescrever na classe para adicionar na Collection da resposta um hypermedia

```

2 usages
@Override
public CollectionModel<CidadeDTO> toCollectionModel(Iterable<? extends Cidade> entities) {
    return super.toCollectionModel(entities).add(linkTo(CidadeController.class).withSelfRel());
}

```

Podemos também utilizar o método `createModelWithId` para adicionar o `linkself` automaticamente em uma conversão de entidade para dto, logo não precisamos adicionar manualmente o `linkself`



passando no argumento o id e o objeto de origem e transforma em um dto desse objeto

```
@Override
public CidadeDTO toModel(Cidade cidade){

    /*_link.self para o próprio recurso
    cidadeDTO.add(linkTo(CidadeController.class)
        .slash(cidadeDTO.getId()).withSelfRel());*/

    /*_link.self para o próprio recurso
    linkTo recebendo um Object method como argumento
    chamada de método para controller específicos
    final Link linkCidade = linkTo( methodOn(CidadeController.class).buscar(cidade
    cidadeDTO.add(linkCidade);*/

    final CidadeDTO cidadeDTO = createModelWithId(cidade.getId(), cidade);

    modelMapper.map(cidade, cidadeDTO);
}
```

e agora o modelMapper copia as informações do objeto para o dto (com linkself) já criado

GET

▼

http://localhost:8080/cidades/3

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

Key	Value
-----	-------

Body

Cookies

Headers (9)

Test Results

⌐ Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON ▼

⌐

```
2      "id": 3,
3      "nome": "São Paulo",
4      "estado": {
5        "id": 2,
6        "nome": "São Paulo",
7        "_links": {
8          "self": {
9            "href": "http://localhost:8080/estados/2"
10         }
11      }
12    },
13    "_links": {
14      "self": {
15        "href": "http://localhost:8080/cidades/3"
16      },
17      "Cidades": {
18        "href": "http://localhost:8080/cidades"
19      }
20    }
21  }
```


19.12. Desafio: adicionando hypermedia nos recursos de usuários

quinta-feira, 13 de abril de 2023 10:42

- No DTO estender RepresentationModel [19.10. Adicionando hypermedia na representação de recursos de coleção](#)
- No Assembler do DTO estender RepresentationModelAssemblerSupport [19.11. Montando modelo de representação com RepresentationModelAssembler](#)
- Alterar nome da representação do DTO usando @Relational no DTO
- Adicionar hypermedia no recurso único de usuário (linkself e linkRelations)
- Adicionar hyermedia na coleção do recurso
- alterar retorno da coleção para CollectionModel
- Adicionar construtor para a superclasse
- Sobrescrever toModel e toCollectionModel

```
@Component
public class UsuarioAssembler extends RepresentationModelAssemblerSupport<Usuario, UsuarioDTO> {

    3 usages
    @Autowired
    private Mapper mapper;

    public UsuarioAssembler() { super(UsuarioController.class, UsuarioDTO.class); }

    @Override
    public UsuarioDTO toModel (Usuario usuario){

        final UsuarioDTO usuarioDTO = createModelWithId(usuario.getId(), usuario);

        mapper.map(usuario, usuarioDTO);

        usuarioDTO.add(linkTo(methodOn(UsuarioController.class).buscar(usuarioDTO.getId())).withSelfRel());

        usuarioDTO.add(linkTo(UsuarioController.class).withRel("usuarios"));

        usuarioDTO.add(linkTo(methodOn(UsuarioGrupoController.class).listar(usuarioDTO.getId())).withRel("grupos-usuario"));

        return usuarioDTO;
    }

    @Override
    public CollectionModel<UsuarioDTO> toCollectionModel(Iterable<? extends Usuario> entities) {
        return super.toCollectionModel(entities).add(linkTo(UsuarioController.class).withSelfRel());
    }
}
```

```
@Override
@GetMapping
public CollectionModel<UsuarioDTO> listar() {
    final CollectionModel<UsuarioDTO> usuariosDTO = aAssembler.toCollectionModel(usuarioService.listar());

    return usuariosDTO;
}

@Override
@GetMapping("/{usuarioId}")
public ResponseEntity<UsuarioDTO> buscar(@PathVariable Long usuarioId) {
    final UsuarioDTO usuarioDTO = aAssembler.toModel(usuarioService.buscar(usuarioId));

    return ResponseEntity.ok(usuarioDTO);
}
```

```
26 usages
@Relation(collectionRelation = "usuarios")
@Getter
@Setter
public class UsuarioDTO extends RepresentationModel<UsuarioDTO> {

    @ApiModelProperty(example = "1")
    private Long id;

    @ApiModelProperty(example = "João da Silva")
    private String nome;

    @ApiModelProperty(example = "joao.ger@algafood.com.br")
    private String email;
}
```

19.13. Corrigindo link de coleção de recurso de responsáveis por restaurante

quinta-feira, 13 de abril de 2023

10:59

```

    @Override
    @GetMapping
    public CollectionModel<UsuarioDTO> listarResponsaveis(@PathVariable Long restauranteId){
        final Restaurante restaurante = restauranteService.buscar(restauranteId);

        return uAssembler.toCollectionModel(restaurante.getResponsaveis()).removeLinks()
            .add(linkTo(methodOn(RestauranteUsuarioResponsavelController.class)
                .listarResponsaveis(restaurante.getId())).withSelfRel());
    }
}
```

o método removeLinks só remove os links do objeto em si, dos objetos alinhados não.

Depois adicionar o método de linkself do endpoint

Foi necessário remover o link pois estava chamando o linkself de usuários

```
usuarioDTO.add(linkTo(UsuarioController.class).withRel("usuarios"));
```

em usuarioAssembler

19.14. Desafio: adicionando hypermedia nos recursos de estados

quinta-feira, 13 de abril de 2023 12:36

19.14. Desafio: adicionando hypermedia nos recursos de estados

quinta-feira, 13 de abril de 2023 14:43

19.15. Adicionando hypermedia em recursos com paginação

quinta-feira, 13 de abril de 2023 14:43

Na listagem de cozinhas, adicionamos o recurso de paginação do recurso cozinha no endpoint listar.

```
@Override
@GetMapping
public Page<CozinhaDTO> listar(@PageableDefault(size = 5) Pageable pageable) {
    final List<CozinhaDTO> cozinhas = cAssembler.toListDTO(cozinhaService.listar(pageable).getContent());

    return new PageImpl<>(cozinhas, pageable, cozinhas.size());
}
```

e Page não é um RepresentationModel (como anotado no DTO). Podemos ajustar esse trecho de código usando PagedResourcesAssembler tipado com Cozinha

```
1 usage
@Autowired
private PagedResourcesAssembler<Cozinha> pagedResourcesAssembler;
```

```
@Override
@GetMapping
public PagedModel<CozinhaDTO> listar(@PageableDefault(size = 5) Pageable pageable) {

    final Page<Cozinha> cozinhasPage = cozinhaService.listar(pageable);

    final PagedModel<CozinhaDTO> cozinhaDTOPagedModel = pagedResourcesAssembler.toModel(cozinhasPage, cAssembler);

    final CollectionModel<CozinhaDTO> cozinhas = cAssembler.toCollectionModel(cozinhasPage.getContent());

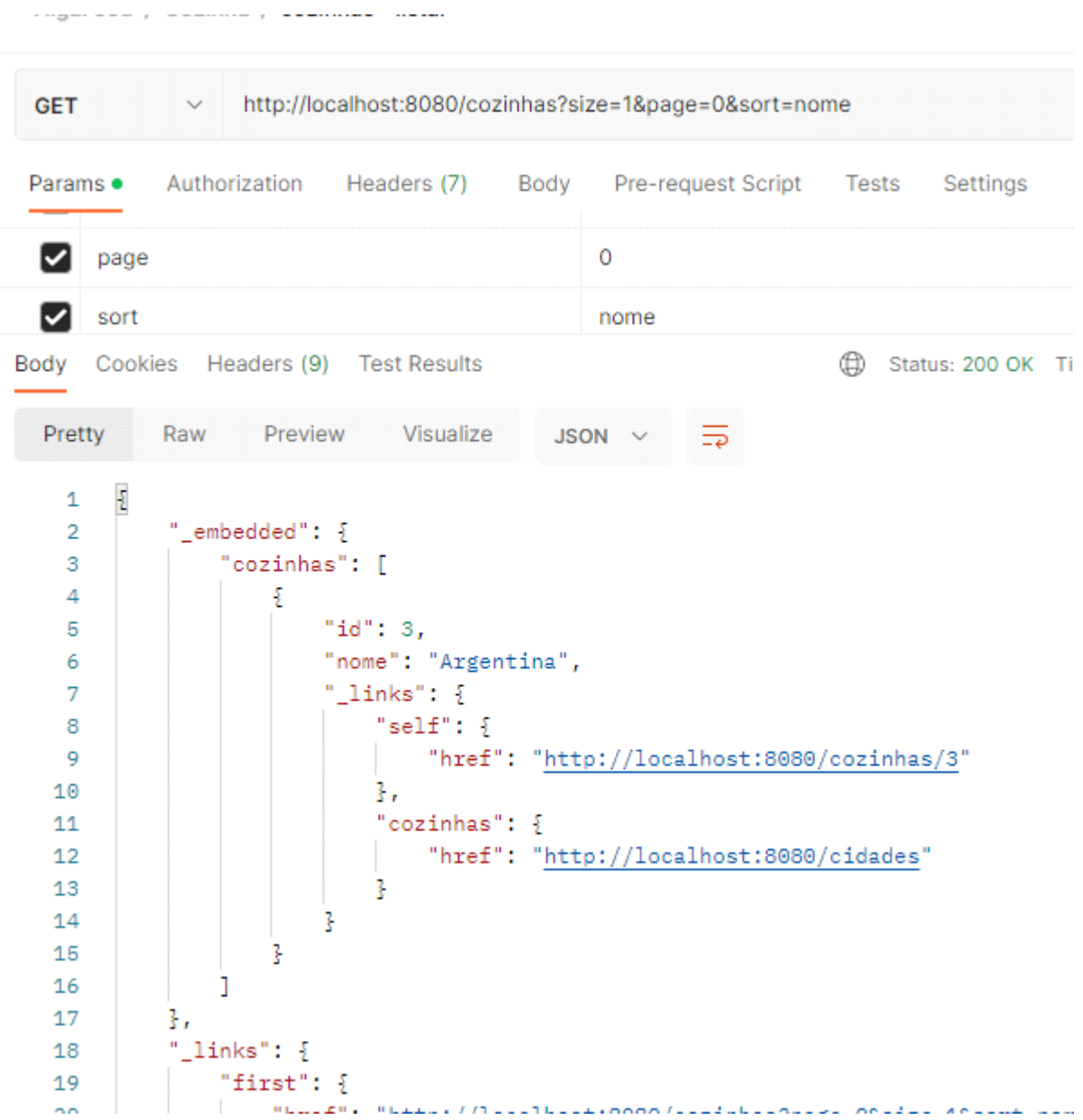
    final PageImpl<CozinhaDTO> cozinhasPageModel = new PageImpl<>(cozinhas, pageable, cozinhas.size());

    return cozinhasPageModel;
    return cozinhaDTOPagedModel;
}
```

antes usávamos um cozinhaAssembler para converter um page de cozinhas (lista) em uma lista de page de cozinhas

mas agora podemos usar PagedResourcesAssembler para transformar o page de cozinhas e um PagedModel de cozinha.

```
{
  "_links": {
    "first": {
      "href": "http://localhost:8080/cozinhas?page=0&size=1&sort=nome,asc"
    },
    "self": {
      "href": "http://localhost:8080/cozinhas?page=0&size=1&sort=nome,asc"
    },
    "next": {
      "href": "http://localhost:8080/cozinhas?page=1&size=1&sort=nome,asc"
    },
    "last": {
      "href": "http://localhost:8080/cozinhas?page=3&size=1&sort=nome,asc"
    }
  },
  "page": {
    "size": 1,
    "totalElements": 4,
    "totalPages": 4,
  }
}
```



19.16. Desafio: adicionando hypermedia em recursos de pedidos (paginação)

quinta-feira, 13 de abril de 2023

16:23

1 - Alterando os Models

RestauranteResumoModel

@Relation(collectionRelation = "restaurantes")

@Setter

@Getter

```
public class RestauranteResumoModel extends  
RepresentationModel<RestauranteResumoModel> {  
    //...
```

}

PedidoResumoModel

@Relation(collectionRelation = "pedidos")

@Setter

@Getter

```
public class PedidoResumoModel extends  
RepresentationModel<PedidoResumoModel> {  
    //...
```

}

ItemPedidoModel

@Getter

@Setter

```
public class ItemPedidoModel extends RepresentationModel<ItemPedidoModel> {  
    //...
```

}

PedidoModel

@Relation(collectionRelation = "pedidos")

@Setter

@Getter

```
public class PedidoModel extends RepresentationModel<PedidoModel> {  
    //...
```

}

CidadeResumoModel

@Relation(collectionRelation = "cidades")

@Setter

@Getter

```
public class CidadeResumoModel extends  
RepresentationModel<CidadeResumoModel> {  
    //...
```

}

FormaPagamentoModel

@Relation(collectionRelation = "formasPagamento")

@Setter

@Getter

```
public class FormaPagamentoModel extends  
RepresentationModel<FormaPagamentoModel> {
```

```

    //...
}
2 - Alterando os Assemblers
PedidoResumoModelAssembler
@Component
public class PedidoResumoModelAssembler
    extends RepresentationModelAssemblerSupport<Pedido, PedidoResumoModel> {

    @Autowired
    private ModelMapper modelMapper;

    public PedidoResumoModelAssembler() {
        super(PedidoController.class, PedidoResumoModel.class);
    }

    @Override
    public PedidoResumoModel toModel(Pedido pedido) {
        PedidoResumoModel pedidoModel = createModelWithId(pedido.getCodigo(),
pedido);
        modelMapper.map(pedido, pedidoModel);

        pedidoModel.add(linkTo(PedidoController.class).withRel("pedidos"));

        pedidoModel.getRestaurante().add(linkTo(methodOn(RestauranteController.class)
        .buscar(pedido.getRestaurante().getId())).withSelfRel());

        pedidoModel.getCliente().add(linkTo(methodOn(UsuarioController.class)
        .buscar(pedido.getCliente().getId())).withSelfRel());

        return pedidoModel;
    }
}
PedidoModelAssembler
@Component
public class PedidoModelAssembler
    extends RepresentationModelAssemblerSupport<Pedido, PedidoModel> {

    @Autowired
    private ModelMapper modelMapper;

    public PedidoModelAssembler() {
        super(PedidoController.class, PedidoModel.class);
    }

    @Override
    public PedidoModel toModel(Pedido pedido) {
        PedidoModel pedidoModel = createModelWithId(pedido.getCodigo(), pedido);
        modelMapper.map(pedido, pedidoModel);
    }
}

```

```

pedidoModel.add(linkTo(PedidoController.class).withRel("pedidos"));

pedidoModel.getRestaurante().add(linkTo(methodOn(RestauranteController.class)
    .buscar(pedido.getRestaurante().getId())).withSelfRel());

pedidoModel.getCliente().add(linkTo(methodOn(UsuarioController.class)
    .buscar(pedido.getCliente().getId())).withSelfRel());

// Passamos null no segundo argumento, porque é indiferente para a
// construção da URL do recurso de forma de pagamento

pedidoModel.getFormaPagamento().add(linkTo(methodOn(FormaPagamentoContro
ller.class)
    .buscar(pedido.getFormaPagamento().getId(), null)).withSelfRel());

pedidoModel.getEnderecoEntrega().getCidade().add(linkTo(methodOn(CidadeContr
oller.class)
    .buscar(pedido.getEnderecoEntrega().getCidade().getId())).withSelfRel());

pedidoModel.getItems().forEach(item -> {
    item.add(linkTo(methodOn(RestauranteProdutoController.class)
        .buscar(pedidoModel.getRestaurante().getId(), item.getProdutoid()))
        .withRel("produto"));
});

return pedidoModel;
}
}

```

3 - Alterando PedidoControllerOpenApi

Vamos alterar o tipo de retorno do método pesquisar para PagedModel

```

@ApiOperation("Pesquisa os pedidos")
@ApiImplicitParams({
    @ApiImplicitParam(value = "Nomes das propriedades para filtrar na resposta,
separados por vírgula",
        name = "campos", paramType = "query", type = "string")
})
PagedModel<PedidoResumoModel> pesquisar(PedidoFilter filtro, Pageable
pageable);

```

4 - Alterando PedidoController

Vamos injetar uma propriedade do tipo PagedResourcesAssembler primeiro

```

@Autowired
private PagedResourcesAssembler<Pedido> pagedResourcesAssembler;
Agora, como alteramos na interface, precisamos mudar a implementação também

```



```
@Override
@GetMapping
public PagedModel<PedidoResumoModel> pesquisar(PedidoFilter filtro,
    @PageableDefault(size = 10) Pageable pageable) {
    pageable = traduzirPageable(pageable);

    Page<Pedido> pedidosPage = pedidoRepository.findAll(
        PedidoSpecs.usandoFiltro(filtro), pageable);

    return pagedResourcesAssembler.toModel(pedidosPage,
        pedidoResumoModelAssembler);
}
```

19.17. Corrigindo links de paginação com ordenação

quinta-feira, 13 de abril de 2023 18:21

GET

http://localhost:8080/pedidos?size=2&sort=restauranteNome,asc

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

<input checked="" type="checkbox"/>	size	2	Quantidade de itens por página
<input checked="" type="checkbox"/>	sort	restauranteNome,asc	Ordena pelo atributo 'dataCriacao' crescent
<input type="checkbox"/>	sort	nomeCliente,desc	Ordena pelo atributo 'valorTotal' decrescent
<input type="checkbox"/>	campos	-cliente	
	Key	Value	Description

Body

Cookies

Headers (9)

Test Results

Status: 200 OK

Time: 318 ms

Size: 1.74 KB

Save as Example

Pretty

Raw

Preview

Visualize

JSON

```
83 |         "next": {
84 |           "href": "http://localhost:8080/pedidos?page=1&size=2&sort=restaurante.nome,asc"
85 |         },
86 |         "last": {
87 |           "href": "http://localhost:8080/pedidos?page=3&size=2&sort=restaurante.nome,asc"
88 |         }
89 |       },
```

o nome da key de ordenação tem que ser repassada para o hypermedia em href

19.18. Adicionando links com template variables

19.18. Adicionando links com template variables

quinta-feira, 13 de abril de 2023

19:22

- Especificar para o consumidor da api que o hypermedia disponibilizado possui query params da requisição, ou seja, se é paginado ou não
- URI modelada / URI com variáveis de template

```
@Override
public PedidoDTO toModel(Pedido pedido){
    final PedidoDTO pedidoDTO = createModelWithId(pedido.getId(), pedido);
    modelMapper.map(pedido, pedidoDTO);

    /*template variables para links com paginação*/
    TemplateVariables pageVariables = new TemplateVariables(
        new TemplateVariable( name: "page", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "size", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "sort", VariableType.REQUEST_PARAM)
    );

    TemplateVariables filtroVariables = new TemplateVariables(
        new TemplateVariable( name: "clienteId", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "restauranteId", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "dataCriacaoInicio", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "dataCriacaoFim", VariableType.REQUEST_PARAM));

    String pedidosUrl = linkTo(PedidoController.class).toUri().toString();

    pedidoDTO.add(Link.of(UriTemplate.of(pedidosUrl, pageVariables.concat(filtroVariables)), relation: "pedidos"));
```

19.19. Desafio: adicionando template variables do filtro de pedidos

sexta-feira, 14 de abril de 2023 09:16

```
@Override
public PedidoDTO toModel(Pedido pedido){
    final PedidoDTO pedidoDTO = createModelWithId(pedido.getId(), pedido);
    modelMapper.map(pedido, pedidoDTO);

    /*template variables para links com paginação*/
    TemplateVariables pageVariables = new TemplateVariables(
        new TemplateVariable( name: "page", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "size", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "sort", VariableType.REQUEST_PARAM)
    );

    TemplateVariables filtroVariables = new TemplateVariables(
        new TemplateVariable( name: "clienteId", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "restauranteId", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "dataCriacaoInicio", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "dataCriacaoFim", VariableType.REQUEST_PARAM));

    String pedidosUrl = linkTo(PedidoController.class).toUri().toString();

    pedidoDTO.add(Link.of(UriTemplate.of(pedidosUrl, pageVariables.concat(filtroVariables)), relation: "pedidos"));
```

AlgaFood / Pedidos / Pedido - Buscar

Save Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (9) Test Results Status: 200 OK Time: 114 ms Size: 1.52 KB Save as Example

Pretty Raw Preview Visualize JSON

```
68     }
69   },
70   "_links": {
71     "self": {
72       "href": "http://localhost:8080/pedidos/6"
73     },
74     "pedidos": {
75       "href": "http://localhost:8080/pedidos?page, size, sort, clienteId, restauranteId, dataCriacaoInicio, dataCriacaoFim",
76       "templated": true
77     }
78   }
79 }
80 }
```

19.20. Refatorando construção e inclusão de links em representation model

sexta-feira, 14 de abril de 2023 09:16

- Refatorar a classe PedidoAssembler no método toModel, pois estamos com responsabilidades demais

```
@Override
public PedidoDTO toModel(Pedido pedido){
    final PedidoDTO pedidoDTO = createModelWithId(pedido.getId(), pedido);
    modelMapper.map(pedido, pedidoDTO);

    /*template variables para links com paginação*/
    TemplateVariables pageVariables = new TemplateVariables(
        new TemplateVariable( name: "page", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "size", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "sort", VariableType.REQUEST_PARAM)
    );

    TemplateVariables filtroVariables = new TemplateVariables(
        new TemplateVariable( name: "clienteId", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "restauranteId", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "dataCriacaoInicio", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "dataCriacaoFim", VariableType.REQUEST_PARAM));

    String pedidosUrl = linkTo(PedidoController.class).toUri().toString();

    pedidoDTO.add(Link.of(UriTemplate.of(pedidosUrl, pageVariables.concat(filtroVariables)), relation: "pedidos"));

    /*pedidoDTO.add(LinkTo(PedidoController.class).withRel("pedidos"))*/

    pedidoDTO.getRestaurante().add(linkTo(methodOn(RestauranteController.class)
        .buscar(pedido.getRestaurante().getId()))).withSelfRel());

    pedidoDTO.getCliente().add(linkTo(methodOn(UsuarioController.class)
        .buscar(pedido.getCliente().getId()))).withSelfRel());

    // Passamos null no segundo argumento, porque é indiferente para a
    // construção da URL do recurso de forma de pagamento
    pedidoDTO.getFormaPagamento().add(linkTo(methodOn(FormaPagamentoController.class)
        .buscar(pedido.getFormaPagamento().getId(), request: null)).withSelfRel());
}
```

e criaremos uma classe para composição dos links

```
package com.algaworks.algafood.api;

import ...

@Component
public class AlgaLinks {

    1 usage
    public Link linkToPedidos (){
        /*template variables para links com paginação*/
        TemplateVariables pageVariables = new TemplateVariables(
            new TemplateVariable( name: "page", TemplateVariable.VariableType.REQUEST_PARAM),
            new TemplateVariable( name: "size", TemplateVariable.VariableType.REQUEST_PARAM),
            new TemplateVariable( name: "sort", TemplateVariable.VariableType.REQUEST_PARAM)
        );

        TemplateVariables filtroVariables = new TemplateVariables(
            new TemplateVariable( name: "clienteId", TemplateVariable.VariableType.REQUEST_PARAM),
            new TemplateVariable( name: "restauranteId", TemplateVariable.VariableType.REQUEST_PARAM),
            new TemplateVariable( name: "dataCriacaoInicio", TemplateVariable.VariableType.REQUEST_PARAM),
            new TemplateVariable( name: "dataCriacaoFim", TemplateVariable.VariableType.REQUEST_PARAM));

        String pedidosUrl = linkTo(PedidoController.class).toUri().toString();

        return Link.of(UriTemplate.of(pedidosUrl, pageVariables.concat(filtroVariables)), relation: "pedidos");
    }
}
```

```
@Override
public PedidoDTO toModel(Pedido pedido){
    final PedidoDTO pedidoDTO = createModelWithId(pedido.getId(), pedido);
    modelMapper.map(pedido, pedidoDTO);

    pedidoDTO.add(algaLinks.linkToPedidos());
    /*pedidoDTO.add(linkTo(PedidoController.class).withRel("pedidos"))*/

    pedidoDTO.getRestaurante().add(linkTo(methodOn(RestauranteController.class)
        .buscar(pedido.getRestaurante().getId()))).withSelfRel());
}
```

injetamos a classe AlgaLinks

Reaproveitando a instância de TemplatesVariables

```
@Component
public class AlgaLinks {

    /*template variables para links com paginação*/
    1 usage
    public static final TemplateVariables PAGINACAO_VARIABLES = new TemplateVariables(
        new TemplateVariable( name: "page", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "size", VariableType.REQUEST_PARAM),
        new TemplateVariable( name: "sort", VariableType.REQUEST_PARAM));

    1 usage
    public Link linkToPedidos (){

        TemplateVariables filtroVariables = new TemplateVariables(
            new TemplateVariable( name: "clienteId", VariableType.REQUEST_PARAM),
            new TemplateVariable( name: "restauranteId", VariableType.REQUEST_PARAM),
            new TemplateVariable( name: "dataCriacaoInicio", VariableType.REQUEST_PARAM),
            new TemplateVariable( name: "dataCriacaoFim", VariableType.REQUEST_PARAM));

        String pedidosUrl = linkTo(PedidoController.class).toUri().toString();

        return Link.of(UriTemplate.of(pedidosUrl, PAGINACAO_VARIABLES.concat(filtroVariables)), relation: "pedidos");
    }
}
```

19.21. Desafio: refatorando construção e inclusão de links

sexta-feira, 14 de abril de 2023 09:18

19.22. Adicionando links de transições de status de pedidos

sexta-feira, 14 de abril de 2023 10:55

```
public PedidoAssembler() { super(PedidoController.class, PedidoDTO.class); }

@Override
public PedidoDTO toModel(Pedido pedido){
    final PedidoDTO pedidoDTO = createModelWithId(pedido.getId(), pedido);
    modelMapper.map(pedido, pedidoDTO);

    pedidoDTO.add(algaLinks.linkToConfirmacaoPedido(pedido.getCodigo(), rel: "confirmar"));
    pedidoDTO.add(algaLinks.linkToCancelamentoPedido(pedido.getCodigo(), rel: "cancelar"));
    pedidoDTO.add(algaLinks.linkToEntregaPedido(pedido.getCodigo(), rel: "entregar"));
}

1 usage
public Link linkToConfirmacaoPedido(String codigoPedido, String rel){
    return linkTo(methodOn(FluxoPedidoController.class).confirmar(codigoPedido)).withRel(rel);
}

1 usage
public Link linkToEntregaPedido(String codigoPedido, String rel){
    return linkTo(methodOn(FluxoPedidoController.class).confirmar(codigoPedido)).withRel(rel);
}

1 usage
public Link linkToCancelamentoPedido(String codigoPedido, String rel){
    return linkTo(methodOn(FluxoPedidoController.class).confirmar(codigoPedido)).withRel(rel);
}
```

```
@Override
@PutMapping("/{confirmacao}")
@ResponseStatus(HttpStatus.NO_CONTENT)
public ResponseEntity<Void> confirmar(@PathVariable String codigoPedido){

    fluxoPedidoService.confirmar(codigoPedido);

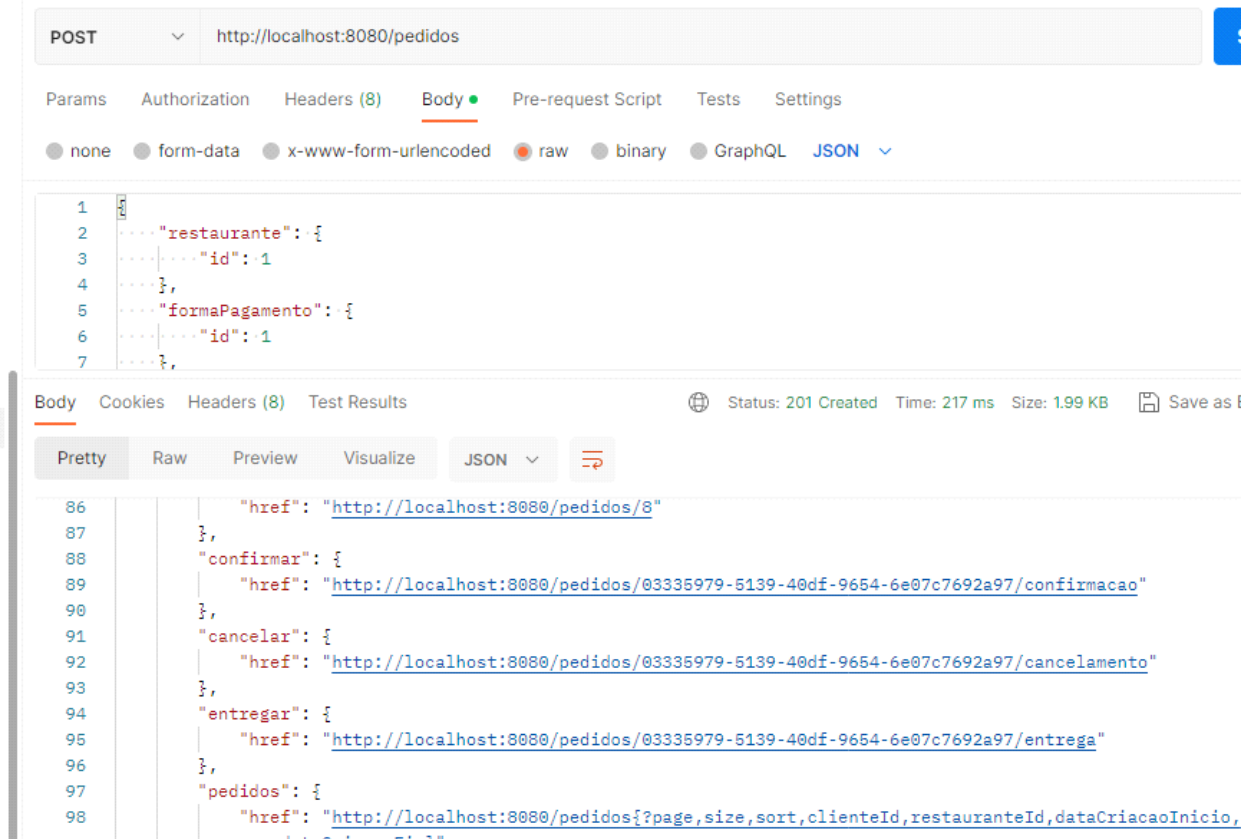
    return ResponseEntity.noContent().build();
}

@Override
@PutMapping("/{entrega}")
@ResponseStatus(HttpStatus.NO_CONTENT)
public ResponseEntity<Void> entregar(@PathVariable String codigoPedido){
    fluxoPedidoService.entregar(codigoPedido);

    return ResponseEntity.noContent().build();
}

@Override
@PutMapping("/{cancelamento}")
@ResponseStatus(HttpStatus.NO_CONTENT)
public ResponseEntity<Void> cancelar(@PathVariable String codigoPedido){
    fluxoPedidoService.cancelar(codigoPedido);
}
```

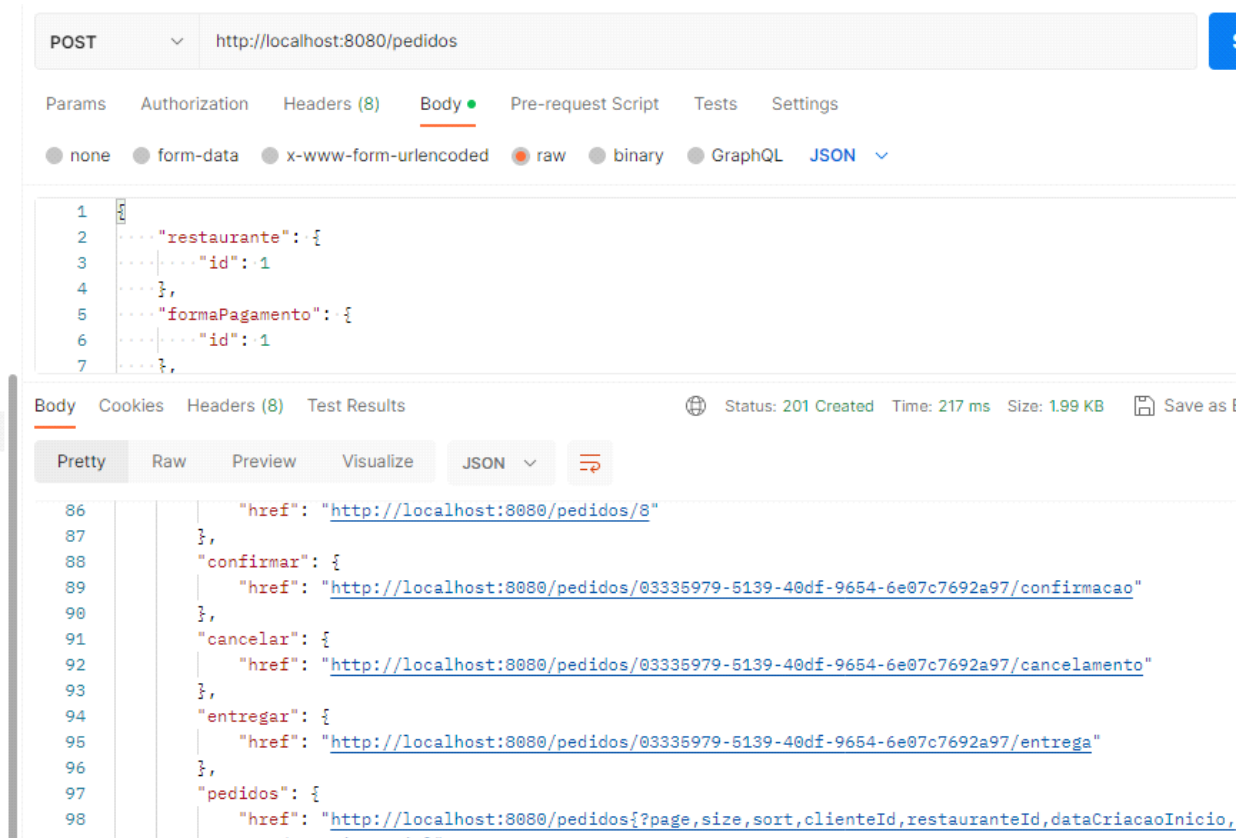
Como os métodos de status de pedido retornavam void, trocamos para retornar ResponseEntity<Void>



19.23. Adicionando links condicionalmente

sexta-feira, 14 de abril de 2023 11:41

Quando adicionamos um pedido, ele automaticamente entra em estado de "CRIADO" e logo, no hypermedia, dá as 3 possibilidades (entregar, cancelar e confirmar)



logo, dependendo do status do pedido, devemos mostrar o hypermedia contendo o link par a transição adequada

Na Enumeração StatusPedido

```
3 usages
public boolean podeAlterarPara(StatusPedido novoStatus){
    return !naoPodeAlterarPara(novoStatus);
}
```

Na entidade Pedido



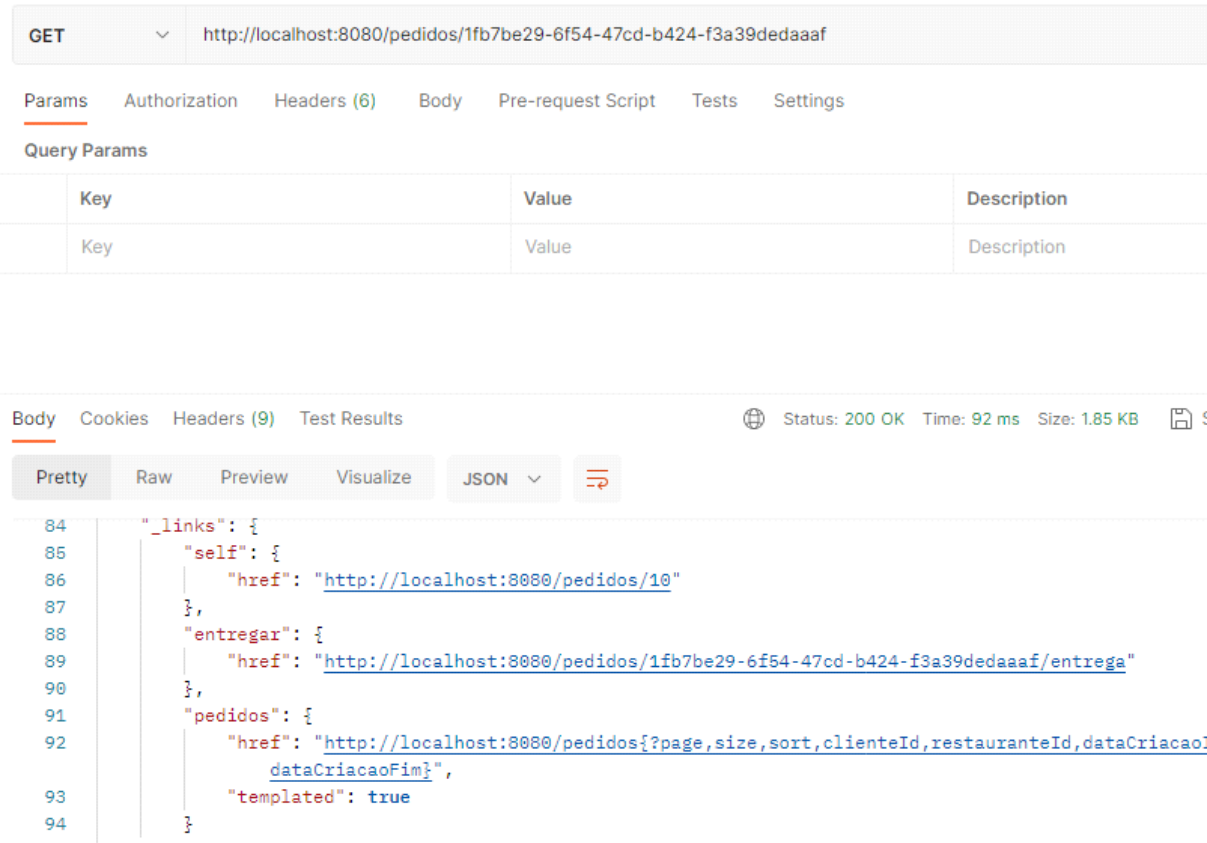
no Assembler de Pedido

```
@Override
public PedidoDTO toModel(Pedido pedido){
    final PedidoDTO pedidoDTO = createModelWithId(pedido.getId(), pedido);
    modelMapper.map(pedido, pedidoDTO);

    if(pedido.podeSerConfirmado())
        pedidoDTO.add(algaLinks.linkToConfirmacaoPedido(pedido.getCodigo(), rel: "confirmar"));

    if(pedido.podeSerCancelado())
        pedidoDTO.add(algaLinks.linkToCancelamentoPedido(pedido.getCodigo(), rel: "cancelar"));

    if(pedido.podeSerEntregue())
        pedidoDTO.add(algaLinks.linkToEntregaPedido(pedido.getCodigo(), rel: "entregar"));
}
```



19.24. Desafio: adicionando hypermedia nos recursos de restaurantes

sexta-feira, 14 de abril de 2023 15:10

19.25. Desafio: adicionando links condicionais no recurso de restaurante

sexta-feira, 14 de abril de 2023 15:24

19.26. Desafio: adicionando template variable de projeção de restaurantes

sexta-feira, 14 de abril de 2023 15:26

19.28. Adicionando links para desassociação de formas de pagamento com restaurante

sexta-feira, 14 de abril de 2023 15:44

```
AlgaLinks.java x RestauranteFormaPagamentoController.java x
public class RestauranteFormaPagamentoController {
    @GetMapping
    public CollectionModel<FormaPagamentoDTO> listar(@PathVariable Long restauranteId) {
        final Restaurante restaurante = restauranteService.buscar(restauranteId);

        final CollectionModel<FormaPagamentoDTO> formasPagamentoDTO = fPAssembler.toCollectionModel(restaurante.getFormasPagamento())
            .removeLinks()
            .add(algaLinks.linkToRestauranteFormasPagamento(restauranteId));

        formasPagamentoDTO.getContent().forEach(formaPagamentoDTO -> {
            formaPagamentoDTO.add(algaLinks.linkToRestauranteFormasPagamentoDesassociacao(restauranteId, formaPagamentoDTO.getId(), rel: "desassociar"));
        });

        return formasPagamentoDTO;
    }
}
```

```
Usage
public Link linkToRestauranteFormasPagamentoDesassociacao(Long restauranteId, Long formaPagamentoId, String rel){
    return linkTo(methodOn(RestauranteFormaPagamentoController.class).desassociar(formaPagamentoId, restauranteId)).withRel(rel);
}
```

nota: o método do controlador tem que retornar ResponseEntity<Void>

19.29. Adicionando links com template variable de caminho de formas de pagamento do restaurante

sexta-feira, 14 de abril de 2023 16:23

- AlgaLinks

```
1 usage
public Link linkToRestauranteFormasPagamentoAssociacao(Long restauranteId, String rel){

    return linkTo(methodOn(RestauranteFormaPagamentoController.class).associar(restauranteId, formaPagamentoId: null)).withRel(rel);
}
```

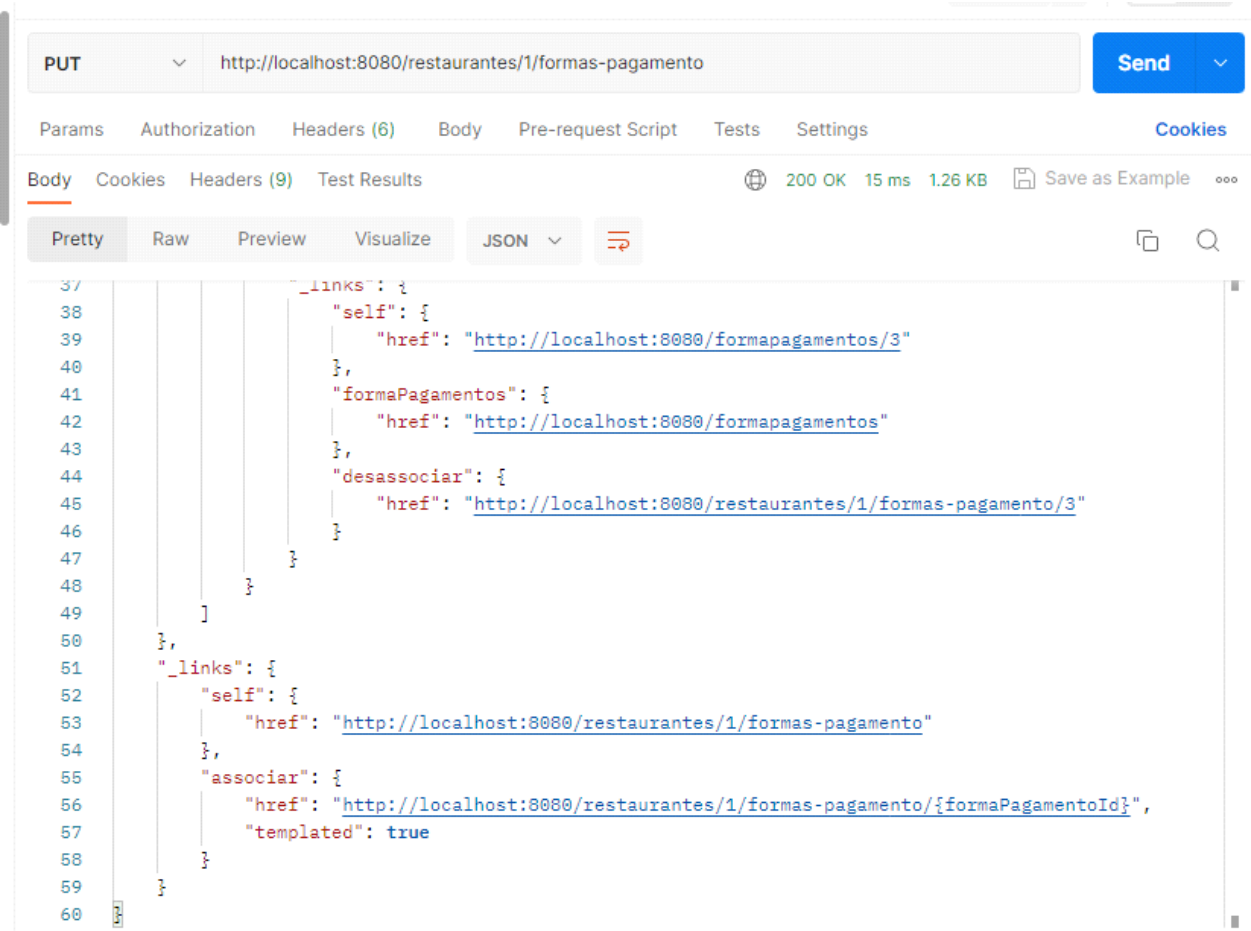
- Controlador do método RestauranteFormaPagamentoController

```
@GetMapping
public CollectionModel<FormaPagamentoDTO> listar(@PathVariable Long restauranteId) {
    final Restaurante restaurante = restauranteService.buscar(restauranteId);

    final CollectionModel<FormaPagamentoDTO> formasPagamentoDTO = fPAssembler.toCollectionModel(restaurante.getFormasPagamento())
        .removeLinks()
        .add(algaLinks.linkToRestauranteFormasPagamento(restauranteId))
        .add(algaLinks.linkToRestauranteFormasPagamentoAssociacao(restauranteId, rel: "associar"));

    formasPagamentoDTO.getContent().forEach(formaPagamentoDTO -> {
        formaPagamentoDTO.add(algaLinks.linkToRestauranteFormasPagamentoDesassociacao(restauranteId, formaPagamentoDTO.getId(), rel: "desassociar"));
    });

    return formasPagamentoDTO;
}
```



19.30. Desafio: adicionando links de associação de restaurantes com responsáveis

sexta-feira, 14 de abril de 2023 16:51

19.31. Desafio: adicionando hypermedia nos recursos de produtos

sexta-feira, 14 de abril de 2023 17:15

19.35. Desafio: adicionando links de associação de usuários com grupos

sábado, 15 de abril de 2023 08:57

19.36. Implementando o Root Entry Point da API

sábado, 15 de abril de 2023 09:15

- Em uma API REST, nunca devemos iniciar o consumo da api com um endpoint conhecido ou específico de um recurso
- Na aula vamos implementar o Root Entry Point ou Home da nossa aAPI. O endpoint <http://localhost:8080/>

```
package com.algaworks.algafood.api.controller;

import ...

@RestController
@RequestMapping(produces = MediaType.APPLICATION_JSON_VALUE)
public class RootEntryPointController {

    9 usages
    @Autowired
    private AlgaLinks algaLinks;

    @GetMapping
    public RootEntryPointModel root() {
        var rootEntryPointModel = new RootEntryPointModel();

        rootEntryPointModel.add(algaLinks.linkToCozinhas( rel: "cozinhas"));
        rootEntryPointModel.add(algaLinks.linkToPedidos( rel: "pedidos"));
        rootEntryPointModel.add(algaLinks.linkToRestaurantes( rel: "restaurantes"));
        rootEntryPointModel.add(algaLinks.linkToGrupos( rel: "grupos"));
        rootEntryPointModel.add(algaLinks.linkToUsuarios( rel: "usuarios"));
        rootEntryPointModel.add(algaLinks.linkToPermissoes( rel: "permissoes"));
        rootEntryPointModel.add(algaLinks.linkToFormasPagamento( rel: "formas-pagamento"));
        rootEntryPointModel.add(algaLinks.linkToEstados( rel: "estados"));
        rootEntryPointModel.add(algaLinks.linkToCidades( rel: "cidades"));

        return rootEntryPointModel;
    }

    3 usages
    private static class RootEntryPointModel extends RepresentationModel<RootEntryPointModel> {
    }
}
```

19.38. Comprimindo as respostas HTTP com Gzip

sábado, 15 de abril de 2023 10:19

- Com a adição de HATEOAS na API, o payload consequentemente fica grande
- Podemos ativar a compressão da resposta da requisição

```
#19.38. Comprimindo as respostas HTTP com Gzip
#Ativar compressão da resposta HTTP
server.compression.enabled=true
#A partir de 2KB de tamanho na resposta será compressado. Padrão 2KB
server.compression.min-response-size=2KB
```

Toda compressão exige também do servidor, logo, é um trade-off ativar a compressão de dados do payload

19.39. Corrigindo as propriedades de links na documentação

sábado, 15 de abril de 2023 10:30

Na documentação do Swagger
endpoint /cidades

Responses

Code	Description
200	OK

Media type
application/json ▾
Controls Accept header.

Example Value | Schema

```
{
  "content": [
    {
      "estado": {
        "id": 1,
        "links": {
          "empty": true
        },
        "nome": "Minas Gerais"
      },
      "id": 0,
      "links": {
        "empty": true
      },
      "nome": "string"
    }
  ],
  "links": {
    "empty": true
  }
}
```

Há uma divergência de links alterados durante o desenvolvimento do HATEOAS da API, no qual não mostra os links

GET ▼ http://localhost:8080/cidades

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (9) Test Results 🌐 Status: 20

Pretty Raw Preview Visualize JSON ▼ 🔍

```

1  {
2    "_embedded": {
3      "cidades": [
4        {
5          "id": 1,
6          "nome": "Uberlândia",
7          "estado": {
8            "id": 1,
9            "nome": "Minas Gerais",
10           "_links": {
11             "self": {
12               "href": "http://localhost:8080/estados/1"
13             }
14           }
15         }
16       ]
17     }
18   }

```

```

return new Docket(DocumentationType.OAS_30)
    .ignoredParameterTypes(ignoredParameterTypes())
    .select()
    //.apis(RequestHandlerSelectors.any())/*Todos os endpoints*/
    .apis(RequestHandlerSelectors.basePackage("com.algaworks.algafood.api"))
    .paths(PathSelectors.any())/*Caminho padrão*/
    .paths(PathSelectors.ant("/restaurantes/*"))
    .build()
    .apiInfo(apiInfo())
    .tags(tags()[0], tags())
    .useDefaultResponseMessages(apply: false)
    .globalResponses(HttpMethod.GET, globalGetResponses())
    .globalResponses(HttpMethod.POST, globalPutResponses())
    .globalResponses(HttpMethod.PUT, globalPutResponses())
    .globalResponses(HttpMethod.DELETE, globalDeleteResponses())
    .additionalModels(typeResolver.resolve(Problem.class))
    .directModelSubstitute(Pageable.class, PageableModelOpenApi.class)
    .directModelSubstitute(Links.class, LinksModelOpenApi.class)
    .alternateTypeRules(AlternateTypeRules.newRule(
        typeResolver.resolve(Page.class, CozinhaDTO.class), CozinhasModelOpenApi.class)
    );

```

```

package com.algaworks.algafood.api.openapi.model;

import ...

2 usages
@Setter
@Getter
@ApiModel("Links")
public class LinksModelOpenApi {

    private LinkModel rel;

    1 usage
    @Setter
    @Getter
    @ApiModel("Link")
    private class LinkModel {

        private String href;
        private boolean templated; /*Em alguns casos temos a propriedade templated*/

    }
}

```

Responses

Code	Description
200	<p>OK</p> <p>Media type</p> <p>application/json ▾</p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre> content: { "estado": { "id": 1, "links": { "rel": { "href": "string", "templated": true } } }, </pre>

X

19.40. Corrigindo a documentação dos endpoints de cidades

sábado, 15 de abril de 2023 10:38

No Json de resposta do endpoint /cidades

AlgaFood / Cidade / Cidade - listar

GET

http://localhost:8080/cidades

ParamsAuthorizationHeaders (6)BodyPre-requests

Query Params

	Key	Value
	Key	Value

BodyCookiesHeaders (9)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   "_embedded": {
3     "cidades": [
4       {
5         "id": 1,
6         "nome": "Uberlândia",
7         "estado": {
8           "id": 1,
9           "nome": "Minas Gerais",
```

Temos um objeto _embedded com outro objeto alinhado de cidades, porém na documentação

```
{
  "content": [
    {
      "estado": {
        "id": 1,
        "links": {
          "rel": {
            "href": "string",
            "templated": true
          }
        },
        "nome": "Minas Gerais"
      },
      "cidades": [
        {
          "id": 1,
          "nome": "Uberlândia",
          "links": {
            "rel": {
              "href": "string",
              "templated": true
            }
          }
        }
      ]
    }
  ]
}
```

Temos um array

19.41. Corrigindo a paginação na documentação

sábado, 15 de abril de 202310:58

Mesmo caso da aula 19.40, porém, no endpoint /cozinhas retorna um Page de Cozinhas pois é um recurso paginado

19.48. Removendo modelo de representação inutilizado da documentação

domingo, 16 de abril de 2023 18:01