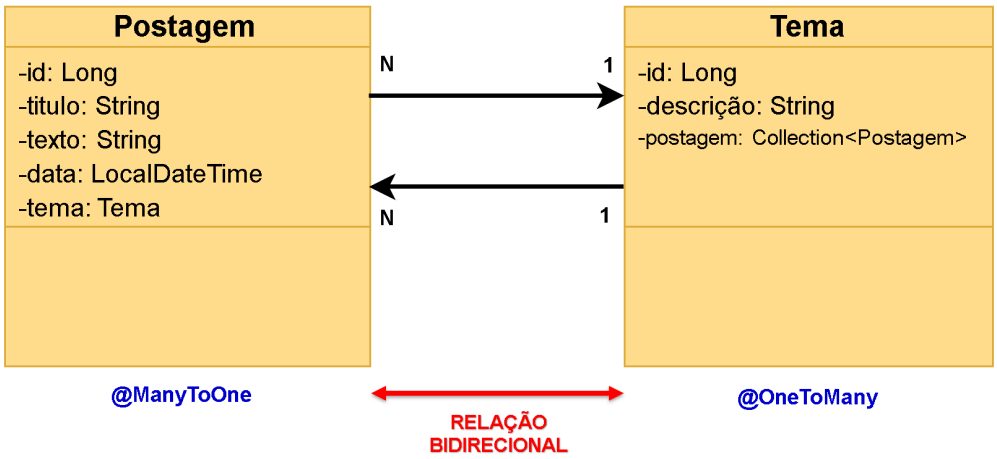
Projeto 02 - Blog Pessoal - Relacionamento entre Classes 02

O que veremos por aqui:

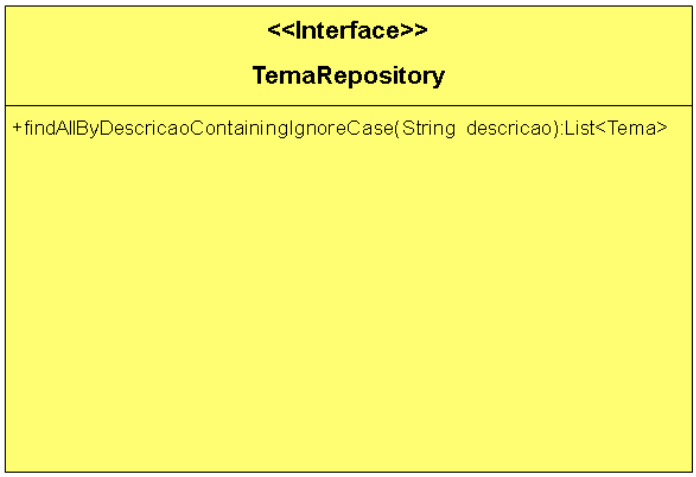
1. Criar a Interface TemaRepository
2. Criar a Classe TemaController

1. O Recurso Tema

Na etapa anterior, começamos a construir o Recurso Tema e criamos o Relacionamento entre as Classes Tema e Postagem. Veja o Diagrama de Classes abaixo:



Nesta etapa, vamos construir a **Interface TemaRepository** e a **Classe TemaController**. O Diagrama de Classes da nossa Interface será igual a imagem abaixo:



ALERTA DE BSM: ***Mantenha a Atenção aos Detalhes ao criar o Recurso Tema. Todas as Camadas (Pacotes: Model, Repository e Controller), já foram criadas no Recurso Postagem.***

DICA: ***Caso você tenha alguma dúvida sobre como criar a Interface TemaRepository e a Classe TemaController, executar o projeto, entre outras, consulte a Documentação do Recurso Postagem.***

🦶 Passo 01 - Criar a Interface TemaRepository na Camada Repository

Agora vamos criar a Interface Repository que chamaremos de **TemaRepository**.

1. Clique com o botão direito do mouse sobre o **Pacote Repository** (**com.generation.blogpessoal.repository**), na Source Folder Principal (**src/main/java**):
2. Na sequência, clique na opção **New → Interface**
3. Na janela **New Java Interface**, no item **Name**, digite o nome da Interface (**TemaRespository**), e na sequência clique no botão **Finish** para concluir.

Agora vamos criar o código da **Interface Repository TemaRepository**:

```
@Repository
public interface TemaRepository extends JpaRepository<Tema, Long> {

  public List<Tema> findAllByDescricaoContainingIgnoreCase(String descricao);

}
```

Observe que foi criada uma Query Method, conforme detalhado abaixo:

Query Method

```
public List<Postagem> findAllByDescricaoContainingIgnoreCase(String Descricao);
```

Instrução SQL equivalente

```
SELECT * FROM tb_temas WHERE descricao LIKE "%descricao%";
```

Palavra		Instrução SQL
find	→	SELECT
All	→	*
By	→	WHERE
Descricao	→	Atributo da Classe Tema
Containing	→	LIKE "%descricao%"
IgnoreCase	→	Ignorando letras maiúsculas ou minúsculas
String descricao	→	Parâmetro do Método contendo a descrição que você deseja procurar.

Como esta consulta retornará um ou mais Objetos da Classe Tema, o Método foi assinado com uma **Collection List** de Objetos da Classe Tema.

ATENÇÃO: ***A instrução FROM tb_temas será inserida pelo JPA ao checar o nome da tabela gerada pela Classe Tema.***

Para concluir, não esqueça de Salvar o código (**File → Save All**).

🦶 Passo 02 - Criar a Classe TemaController na Camada Controller

Agora, vamos criar a Classe Controladora que chamaremos de **TemaController**.

1. Clique com o botão direito do mouse sobre o pacote controller da aplicação (**com.generation.blogpessoal.controller**).
2. Na sequência, clique na opção **New → Class**
3. Na janela **New Java Class**, no item **Name**, digite o nome da Classe (**TemaController**), e na sequência clique no botão **Finish** para concluir.

Agora vamos criar o código da **Casse Controladora TemaController**:

```
@RestController
@RequestMapping("/temas")
@CrossOrigin(origins = "*", allowedHeaders = "*")
public class TemaController {

    @Autowired
    private TemaRepository temaRepository;

    @GetMapping
    public ResponseEntity<List<Tema>> getAll(){
        return ResponseEntity.ok(temaRepository.findAll());
    }

    @GetMapping("/{id}")
    public ResponseEntity<Tema> getById(@PathVariable Long id){
        return temaRepository.findById(id)
            .map(resposta -> ResponseEntity.ok(resposta))
            .orElse(ResponseEntity.status(HttpStatus.NOT_FOUND).build());
    }

    @GetMapping("/descricao/{descricao}")
    public ResponseEntity<List<Tema>> getByTitle(@PathVariable
    String descricao){
        return ResponseEntity.ok(temaRepository
            .findAllByDescricaoContainingIgnoreCase(descricao));
    }

    @PostMapping
    public ResponseEntity<Tema> post(@Valid @RequestBody Tema tema){
        return ResponseEntity.status(HttpStatus.CREATED)
                .body(temaRepository.save(tema));
    }
```

```
    @PutMapping
    public ResponseEntity<Tema> put(@Valid @RequestBody Tema tema){
        return temaRepository.findById(tema.getId())
            .map(resposta -> ResponseEntity.status(HttpStatus.CREATED)
            .body(temaRepository.save(tema)))
            .orElse(ResponseEntity.status(HttpStatus.NOT_FOUND).build());
    }

    @ResponseStatus(HttpStatus.NO_CONTENT)
    @DeleteMapping("/{id}")
    public void delete(@PathVariable Long id) {
        Optional<Tema> tema = temaRepository.findById(id);

        if(tema.isEmpty())
            throw new ResponseStatusException(HttpStatus.NOT_FOUND);

        temaRepository.deleteById(id);
    }

}
```

Para concluir, não esqueça de Salvar o código (**File → Save All**).

👣 Passo 03 - Testar o Recurso Tema no Insomnia

Vamos criar no Insomnia todas as requisições necessárias para testar os 6 Métodos do Recurso Tema. Veja abaixo como ficam as requisições para testar o Recurso Tema.

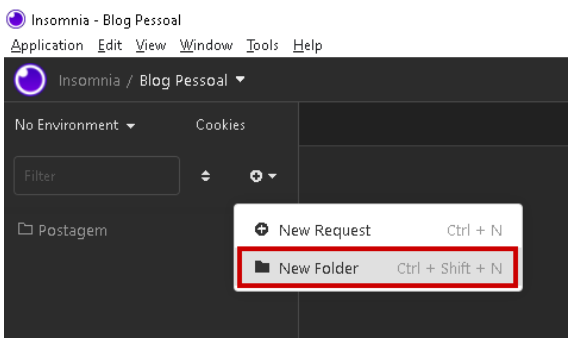
DICA: ***Caso você tenha alguma dúvida sobre como criar as Requisições, consulte a Documentação do Recurso Postagem.***

ATENÇÃO: ***Depois de criar o Relacionamento entre Classes, todas as Consultas dos Recursos Postagem e Tema trarão os Objetos associados, ou seja, Cada Objeto da Classe Postagem trará o Objeto Tema associado e cada Objeto da Classe Tema trará a lista de Objetos Postagem associados.***

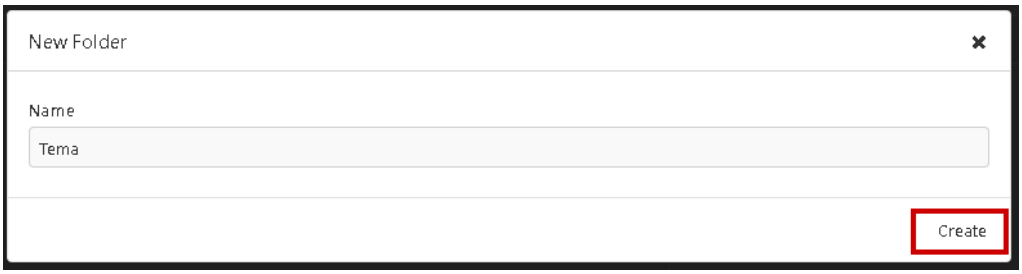
3.1. Criando a Pasta Tema

Vamos criar dentro da **Collection Blog Pessoal** a **Pasta Tema**, que guardará todas as requisições do **Recurso Tema**.

1. Na **Collection Blog Pessoal**, clique no botão ⊕. No menu que será aberto, clique na opção **New Folder**.

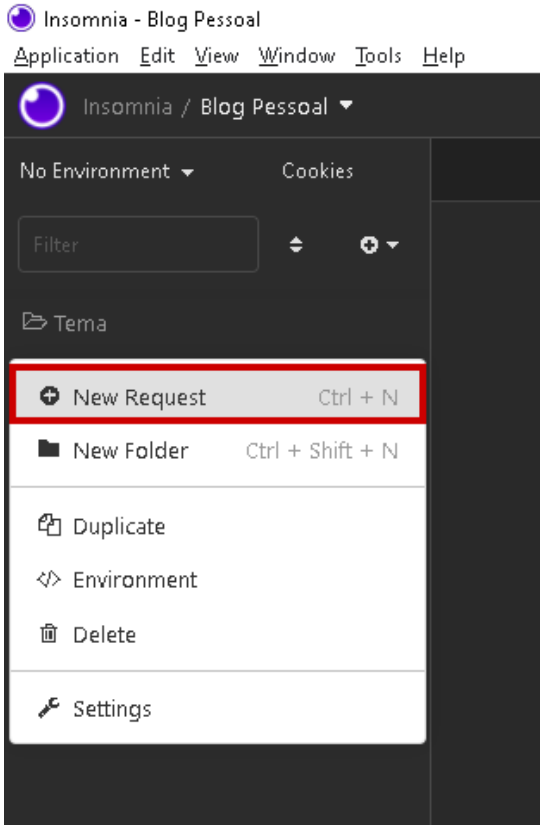


2. Na janela que será aberta, informe o nome da pasta (**Tema**) e clique no botão **Create** para concluir.



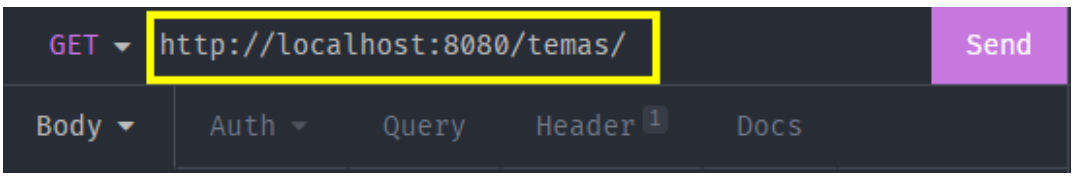
3.2. Criando a Request - Consultar todos os Temas

1. Clique com o botão direito do mouse sobre a **Pasta Tema** para abrir o menu e clique na opção **New Request**.



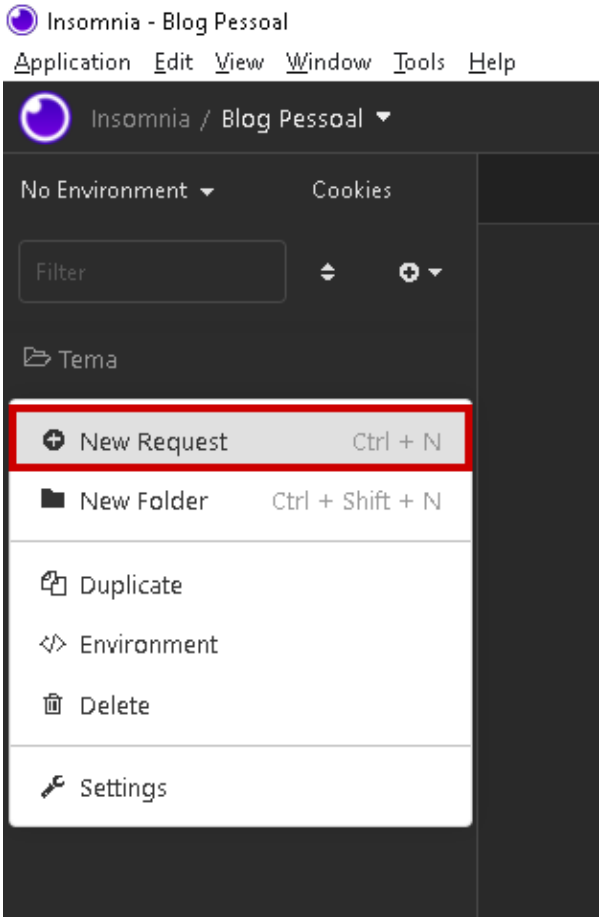
2. Na janela que será aberta, informe o nome da requisição e o Método HTTP que será utilizado (**GET**). Clique no botão **Create** para concluir.

3. Configure a requisição conforme a imagem abaixo:



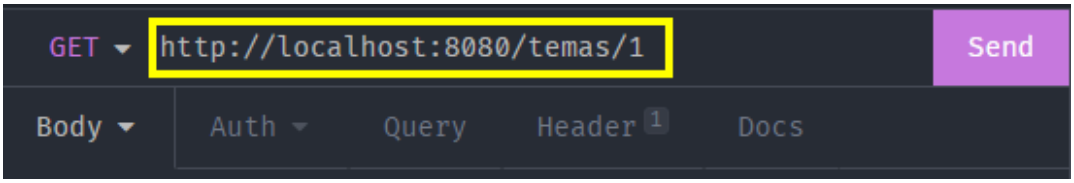
3.3. Criando a Request - Consultar Tema por ID

1. Clique com o botão direito do mouse sobre a **Pasta Tema** para abrir o menu e clique na opção **New Request**.



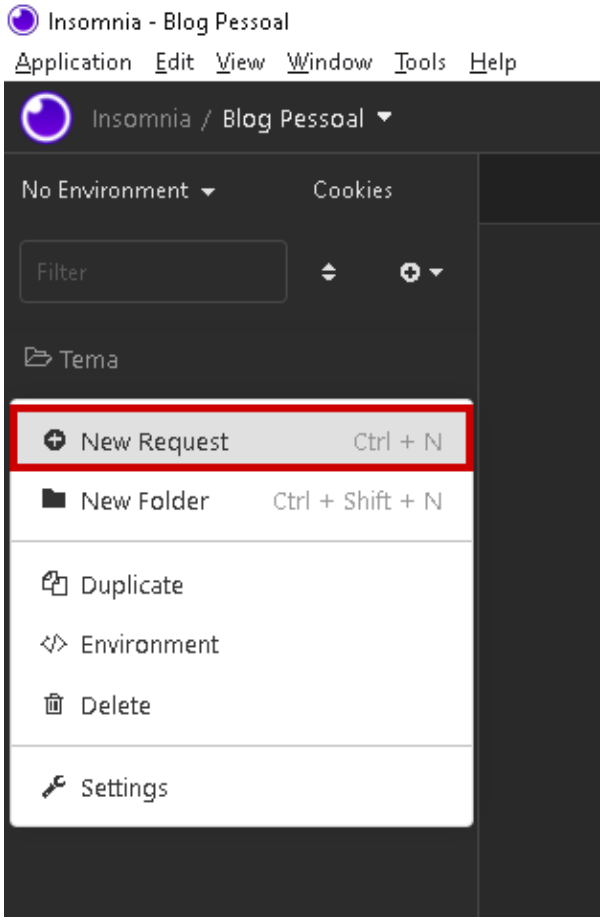
2. Na janela que será aberta, informe o nome da requisição e o Método HTTP que será utilizado (**GET**). Clique no botão **Create** para concluir.

3. Configure a requisição conforme a imagem abaixo:



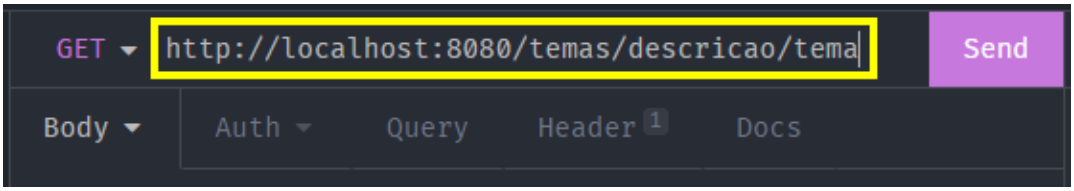
3.4. Criando a Request - Consultar Tema por Descrição

1. Clique com o botão direito do mouse sobre a **Pasta Tema** para abrir o menu e clique na opção **New Request**.



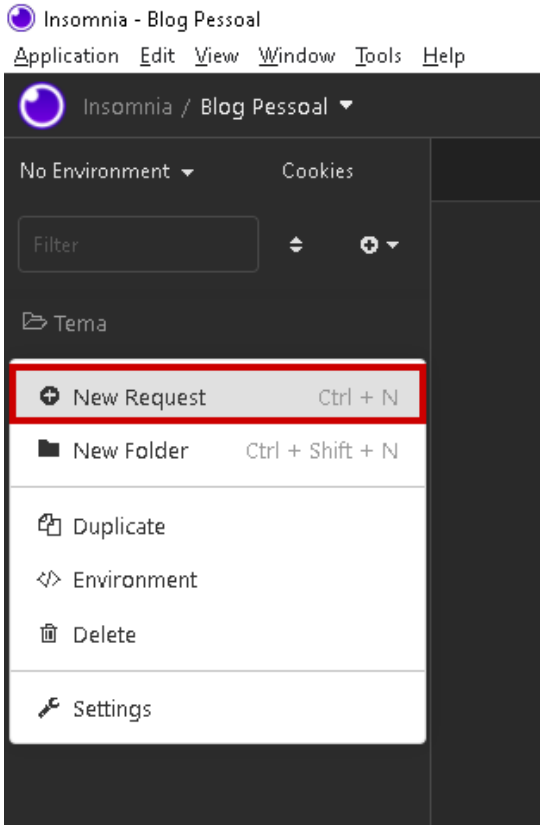
2. Na janela que será aberta, informe o nome da requisição e o Método HTTP que será utilizado (**GET**). Clique no botão **Create** para concluir.

3. Configure a requisição conforme a imagem abaixo:



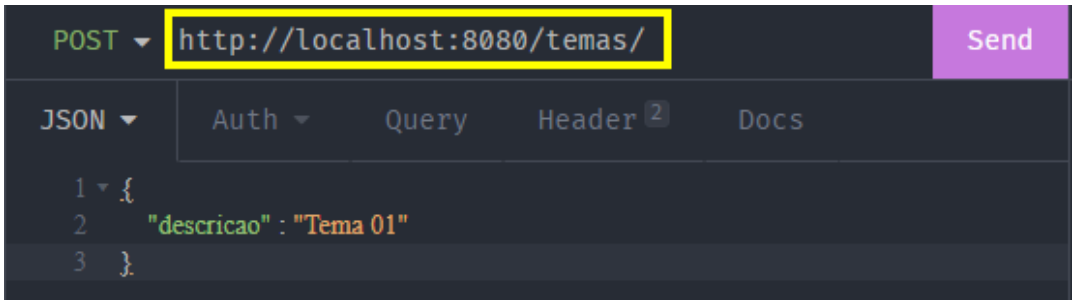
3.5. Criando a Request - Cadastrar Tema

1. Clique com o botão direito do mouse sobre a **Pasta Tema** para abrir o menu e clique na opção **New Request**.



2. Na janela que será aberta, informe o nome da requisição e o Método HTTP que será utilizado (**POST**). Depois de selecionar o Método **POST**, observe que será necessário informar o formato em que os dados serão enviados através do Corpo da Requisição. Selecione o formato **JSON**, como indicado na imagem abaixo em vermelho e clique no botão **Create** para concluir.

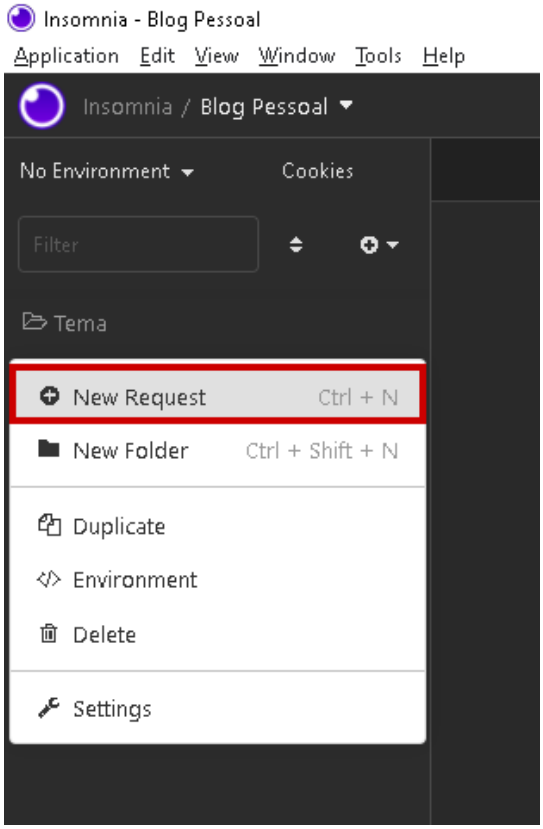
3. Configure a requisição conforme a imagem abaixo:



4. Observe que na requisição do tipo POST o Corpo da requisição (Request Body), deve ser preenchido com um JSON contendo os dados do tema que você deseja persistir no Banco de dados, exceto o ID que será gerado pelo Banco de dados.

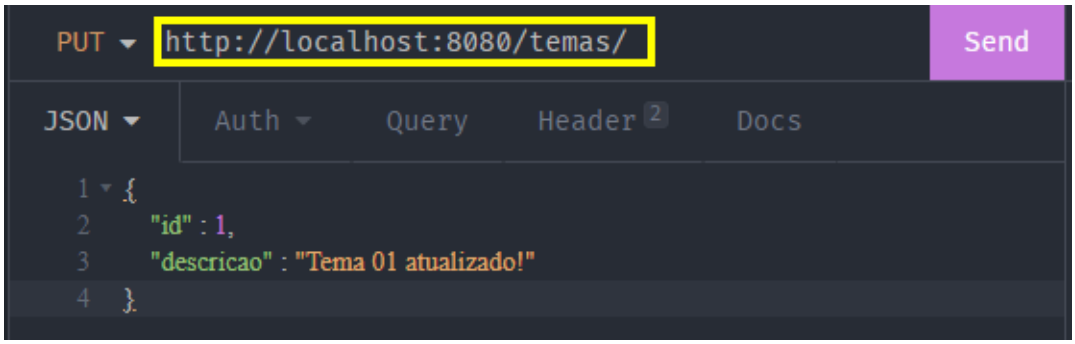
3.6. Criando a Request - Atualizar Tema

1. Clique com o botão direito do mouse sobre a **Pasta Tema** para abrir o menu e clique na opção **New Request**.



2. Na janela que será aberta, informe o nome da requisição e o Método HTTP que será utilizado (**PUT**). Depois de selecionar o Método **PUT**, observe que será necessário informar o formato em que os dados serão enviados através do Corpo da Requisição. Selecione o formato **JSON**, como indicado na imagem abaixo em vermelho e clique no botão **Create** para concluir.

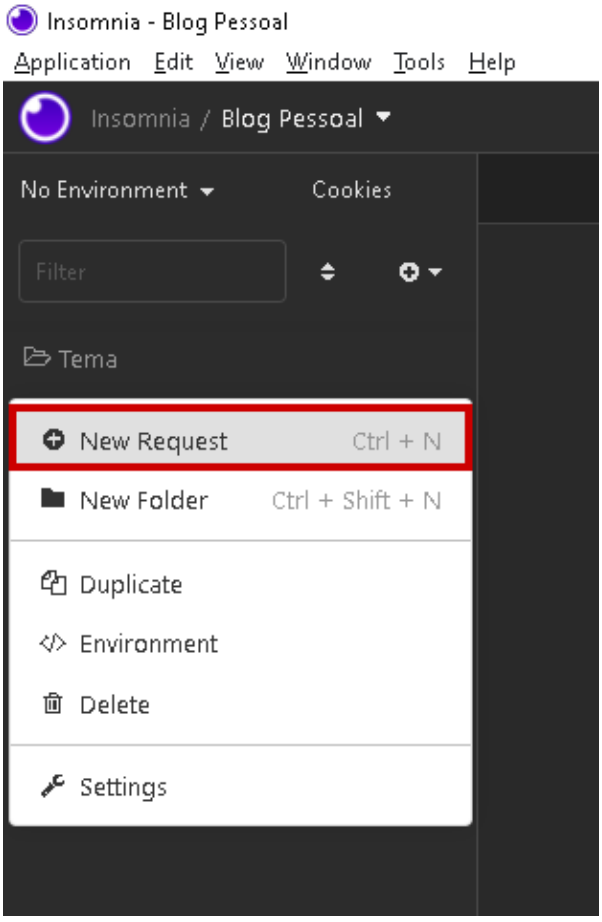
3. Configure a requisição conforme a imagem abaixo:



4. Observe que na requisição do tipo PUT o Corpo da requisição (Request Body), deve ser preenchido com um JSON contendo os dados do tema que você deseja persistir no Banco de dados, inclusive o ID que será utilizado para localizar o tema no Banco de dados.

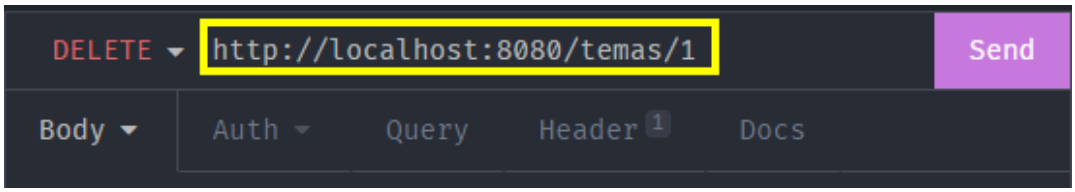
3.7. Criando a Request - Deletar Tema

1. Clique com o botão direito do mouse sobre a **Pasta Tema** para abrir o menu e clique na opção **New Request**.



2. Na janela que será aberta, informe o nome da requisição e o Método HTTP que será utilizado (**DELETE**). Clique no botão **Create** para concluir.

3. Configure a requisição conforme a imagem abaixo:



ATENÇÃO: ***Ao Deletar um Objeto da Classe Tema, todos os Objetos da Classe Postagem associados serão Deletados.***

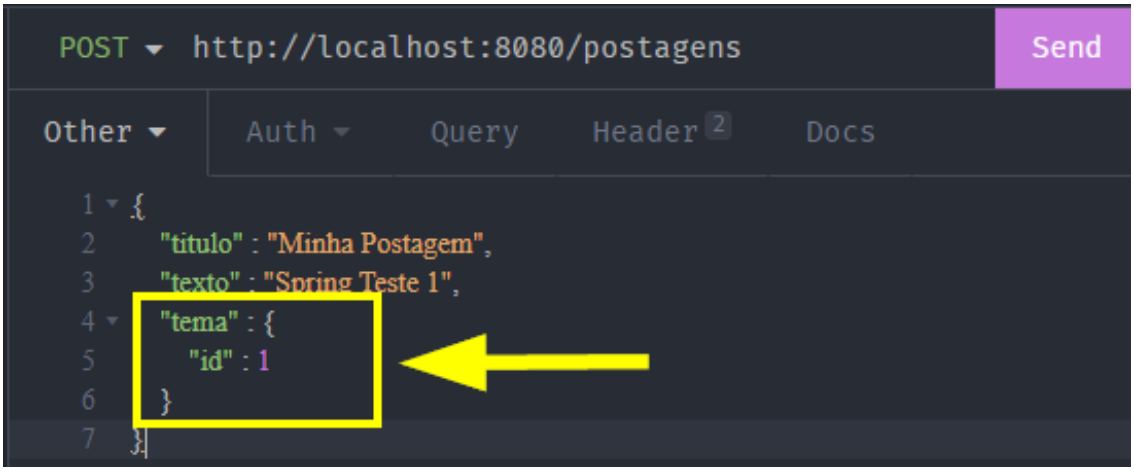
🦶 Passo 04 - Atualizar as Requisições Cadastrar e Atualizar Postagem no Insomnia

Como habilitamos o Relacionamento entre as Classes, para Cadastrarmos e Alterarmos as Postagens vamos precisar atender alguns requisitos:

- Os Temas devem ser persistidos antes de criar uma nova Postagem.
- Nas requisições Cadastrar e Atualizar Postagem, o JSON enviado no Corpo da Requisição deve conter um Objeto da Classe Tema identificado apenas pelo **Atributo id**.

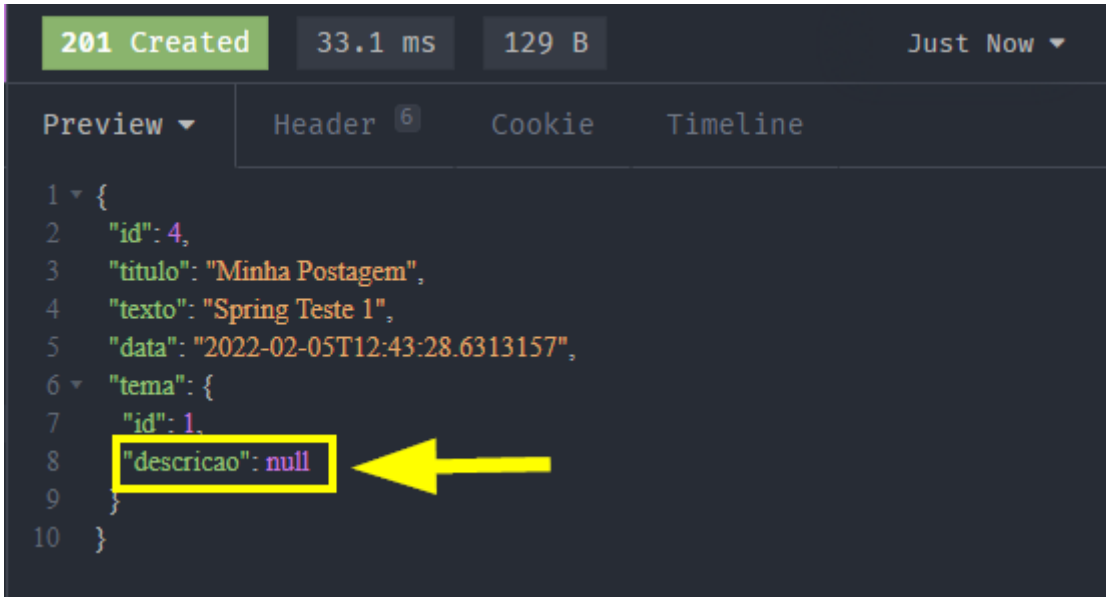
4.1. Atualização - Requisição Cadastrar Postagem

Vamos alterar o Corpo da requisição (Body), conforme a imagem abaixo:



No item marcado em amarelo na imagem acima, observe que está sendo passado dentro do JSON um **Objeto da Classe Tema** chamado **tema**, identificado apenas pelo **Atributo id**.

A Resposta da Requisição será semelhante a figura abaixo:

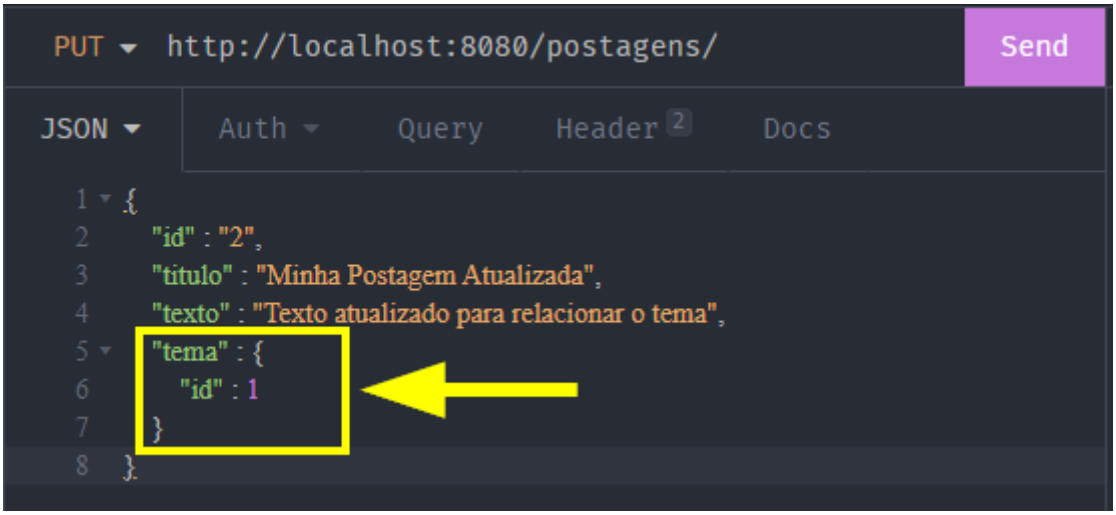


Observe que o **Atributo descricao** aparece como **null**. Isso acontece porquê o Objeto Postagem foi persistido, mas não foi executado um Método de Consulta para buscar os Atributos do Objeto Tema persistidos no Banco de dados. Quando você efetuar um Método de Consulta no Recurso Postagem, o Tema aparecerá com o Atributo descricao preenchido.

ATENÇÃO: ***O Objeto Tema deve ser persistido no Banco de dados antes de ser inserido no JSON da requisição Cadastrar Postagem.***

4.2. Atualização - Requisição Atualizar Postagem

Vamos alterar o Corpo da requisição (Body), conforme a imagem abaixo:



No item marcado em amarelo na imagem acima, observe que está sendo passado dentro do JSON um **Objeto da Classe Tema** chamado **tema**, identificado apenas pelo **Atributo id**.

A Resposta da Requisição será semelhante a figura abaixo:

Diferente do Método Cadastrar, o Método Atualizar Postagem exibe o Objeto Postagem e Tema completos porquê o Objeto tema já está associado ao Objeto postagem e foi apenas atualizado.

ATENÇÃO: ***O Objeto Tema deve ser persisitido no Banco de dados antes de ser inserido no JSON da requisição Atualizar Postagem.***

DESAFIO: ***O que acontecerá se você inserir no JSON das requisições Cadastrar e Atualizar Postagem, no Objeto da Classe Tema chamado tema, um id que não existe? Insira no Atributo id, do Objeto Tema, um id como 100, por exemplo, e veja o que acontece.***

👣 Passo 05 - Atualizar os Métodos post e put na Classe PostagemController

Atualmente os Métodos **post** e **put** estão implementados na **Classe PostagemController** conforme as imagens abaixo:

Método Post:

```
46
47    @PostMapping
48    public ResponseEntity<Postagem> post(@Valid @RequestBody Postagem postagem){
49        return ResponseEntity.status(HttpStatus.CREATED)
50                .body(postagemRepository.save(postagem));
51    }
52
```

Método Put:

```
52
53    @PutMapping
54    public ResponseEntity<Postagem> put(@Valid @RequestBody Postagem postagem){
55        return postagemRepository.findById(postagem.getId())
56                .map(resposta -> ResponseEntity.status(HttpStatus.OK)
57                        .body(postagemRepository.save(postagem)))
58                .orElse(ResponseEntity.status(HttpStatus.NOT_FOUND).build());
59    }
60
```

Se você fez o desafio acima, percebeu que estas implementações não conseguem checar se o **Objeto da Classe Tema existe**, logo se você inserir um Objeto que não existe (um Id que não existe no Banco de dados), devido ao Relacionamento entre as Classes, será retornado o **HTTP Status 500 - Internal Server Error**.

Para evitar este erro, faremos alguns ajustes na **Classe PostagemController**.

5.1. Inserir uma Injeção de Dependência da Classe Tema na Classe PostagemController

```
31
32    @Autowired
33    private PostagemRepository postagemRepository;
34
35    @Autowired
36    private TemaRepository temaRepository;
37
```

Linhas 35 e 36: Para termos acesso aos **Métodos das Classes Tema e TemaController**, precisamos inserir uma uma Injeção de Dependência da Classe Tema, logo abaixo da uma Injeção de Dependência da Classe Postagem.

5.2. Atualização do Método post da Classe PostagemController

```
54
55    @PostMapping
56    public ResponseEntity<Postagem> post(@Valid @RequestBody Postagem postagem){
57        if (temaRepository.existsById(postagem.getTema().getId()))
58            return ResponseEntity.status(HttpStatus.CREATED)
59                    .body(postagemRepository.save(postagem));
60
61        return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
62    }
63
```

Linha 57: Através do Método **existsById(Long id)**, da Interface TemaRepository (Herança da Interface JPA), checamos se o id passado no Objeto tema, da Classe Tema, inserido no Objeto postagem, da Classe Postagem, existe.

Para obter o id do tema, utilizamos os Métodos get das 2 Classes: **postagem.getTema().getId()**

Linha 58: Executa o Método padrão da Interface JpaRepository (save(postagem)), se o Objeto tema existir, e retorna o **HTTP Status CREATED→201** se o Objeto foi persistido no Banco de dados.

Linha 61: Se o Objeto tema não for encontrado pelo Método **existsById(Long id)**, será retornado o **HTTP Status BAD REQUEST → 400**. O método **build()** constrói a Resposta com o HTTP Status retornado.

 **Documentação: *existsById()***

5.3. Atualização do Método put da Classe PostagemController

```
63
64    @PutMapping
65    public ResponseEntity<Postagem> put(@Valid @RequestBody Postagem postagem){
66        if (postagemRepository.existsById(postagem.getId())){
67
68            if (temaRepository.existsById(postagem.getTema().getId()))
69                return ResponseEntity.status(HttpStatus.OK)
70                        .body(postagemRepository.save(postagem));
71
72            return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
73
74        }
75
76        return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
77
78    }
79
```

Linha 66: Através do Método **existsById(Long id)**, da Interface PostagemRepository (Herança da Interface JPA), checamos se o id passado no Objeto postagem, da Classe Postagem, existe. Caso o Objeto não exista, não é possível atualizar.

Linha68: Através do Método **existsById(Long id)**, da Interface TemaRepository (Herança da Interface JPA), checamos se o id passado no Objeto tema, da Classe Tema, inserido no Objeto postagem, da Classe Postagem, existe.

Para obter o id do tema, utilizamos os Métodos get das 2 Classes: **postagem.getTema().getId()**

Linha 69: Executa o Método padrão da Interface JpaRepository (save(postagem)), se o Objeto tema existir, e retorna o **HTTP Status OK→200** se o Objeto foi atualizado no Banco de dados.

Linha 72: Se o Objeto tema não for encontrado pelo Método **existsById(Long id)**, será retornado o **HTTP Status BAD REQUEST → 400**. O método **build()** constrói a Resposta com o HTTP Status retornado.

Linha 76: Se o Objeto Postagem não for encontrado pelo Método **existsById(Long id)**, será retornado o **HTTP Status NOT FOUND → 404** (Não Encontrado!), indicando que a Postagem não existe. O método **build()** constrói a Resposta com o HTTP Status retornado.

Para concluir, não esqueça de Salvar o código (**File → Save All**).

Refaça o Desafio no Postman e veja que o **HTTP Status 500 - Internal Server Error** foi substituído pelo **HTTP Status 400 - Bad Request**, indicando que o id deve ser válido.

 **Código fonte do projeto**