204 = Requisição processada e sem conteúdo.

@DeleteMapping("/{id}") // parametro dinâmico  
@Transactional  
public **ResponseEntity** excluirMedico(@PathVariable Long id) {  
 var medico = repository.getReferenceById(id);

medico.excluir();  
  
 **// ResponseEntity é uma classe do Spring que podemos detalhar o retorno.  
 // noContent ele cria um objeto e vamos chamar o biuld para construir  
 return ResponseEntity.*noContent*().build();  
   
 // Agora vai retornar o 204 no Content quer dizer que foi prcessado com sucesso e nao tem conteudo na resposta.  
}**

**Agora observa esse caso que retorna uma lista, com o sem o Response**

@GetMapping()  
public Page<DadosListagemMedico> listaDeMedicos(@PageableDefault(size = 20, sort = {"nome"}) Pageable paginacao) {

return repository.findAllByAtivoTrue(paginacao).map(DadosListagemMedico::new);  
  
}

Agora com olha como ficaria

@GetMapping()

<PAGE> é nosso genérico é assim que fazemos.

public ResponseEntity<Page<DadosListagemMedico>> listaDeMedicos(@PageableDefault(size = 20, sort = {"nome"}) Pageable paginacao) {  
 var page = repository.findAllByAtivoTrue(paginacao).map(DadosListagemMedico::new);

// Agora nessa sim quero devolver o código 200

E no parâmetro do OK vamos devolver a lista de medicos  
 return ResponseEntity.*ok*(page);  
  
}

**Agora vamos fazer com o atualizar dados, mas queremos devolver os dados atualizados.**

@PutMapping()  
@Transactional   
public ResponseEntity atualizarDados(@RequestBody @Valid DadosAtualizacaoMedicos dados) {  
  
 var medico = repository.getReferenceById(dados.id());  
 medico.atualizarInformacoes(dados);  
  
  
 **// Aqui queremos passar os dados que foram atualizados  
 // Mas nao podemos passar a entidade por segurança e sim um DTO  
 // Aqui mandamos ele criar um novo DTO  
 return ResponseEntity.*ok*(new DadosDetalhamentoMedico(medico));  
  
}**

package com.allephnogueira.api.medico;  
  
import com.allephnogueira.api.endereco.Endereco;  
  
public record DadosDetalhamentoMedico(Long id, String nome, String email, String crm, String telefone, Especialidade especialidade, Endereco endereco) {  
 // Aqui vamos receber os dados do medico  
  
 // Agora sim vamos pegar os dados que vem la do medico e vamos colocar aqui.  
 public DadosDetalhamentoMedico(Medico medico) {  
 this(medico.getId(), medico.getNome(), medico.getEmail(), medico.getCrm(), medico.getTelefone(), medico.getEspecialidade(), medico.getEndereco());  
 }  
}

Repara que criamos um construtor dentro do Record e nesse construtor passamos os dados que vem lá do médico.