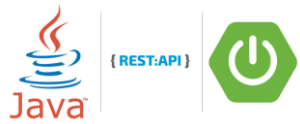


Cache

O **Spring** também sabe trabalhar com cache, em **produção DEVE SE USAR um provedor de cache como redis** por exemplo. Além da dependência de cache ser adicionado, também é preciso **habilitar o seu uso na aplicação:**

```java
@EnableCaching
public class Application {

    public static void main(String[] args) { Spr

}
```

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

A ideia do Cache é ir em algum **componente Spring** da aplicação e no **método que deseja guardar o retorno em cache** anotar com **@Cacheable**. Essa **anotação** precisa de um valor, esse valor serve como um **identificador do cache**, pode ser que sua aplicação tenha mais que um Cache, então ele **precisa associar o cache ao método**:

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

    @Autowired
    private UsuarioRepository usuarioRepository;

    @GetMapping("/pagina")
    @Cacheable(value = "buscaPaginadaUsuario")
    public Page<UsuarioDto> buscaFlexivelPaginada(
            @PageableDefault(sort = "id", direction = Sort.Direction.ASC,page = 0, size = 5) Pageable paginacao){

        Page<Usuario> usuarios = usuarioRepository.findAll(paginacao);
        return UsuarioDto.converterPaginado(usuarios);
    }
}
```

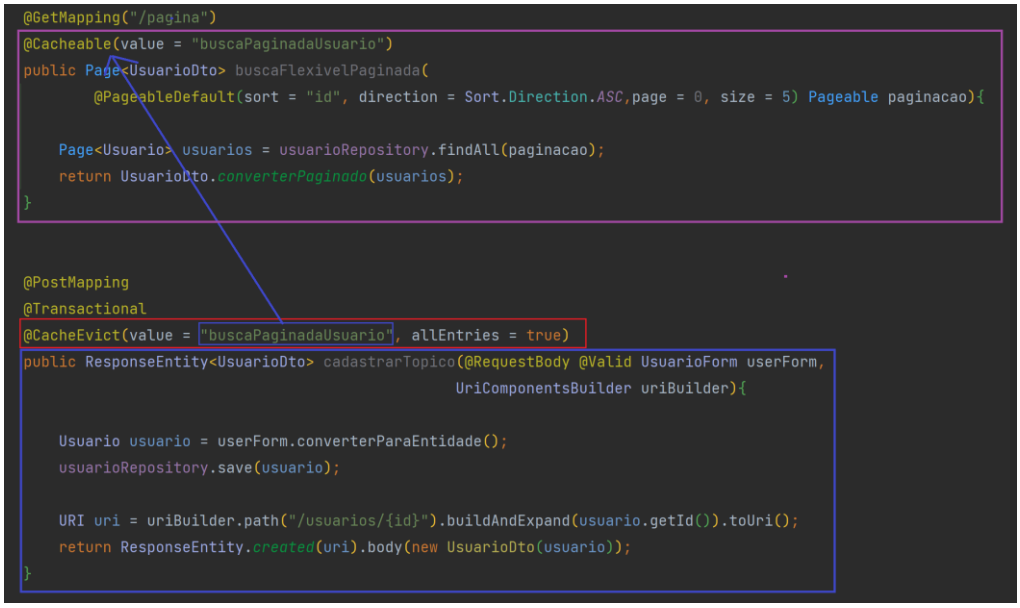
O Spring é inteligente o suficiente para saber quando o cache não é o mesmo pra quando a requisição muda, ele interpreta os parâmetros e valores passados e cria um novo cache caso haja variação

Invalidar Cache

E se no meio tempo que o **resultado da listagem de usuários estiverem em cache** eu **cadastrar um usuário novo**, ou **deletar um**, ou **alterar**, o **cache** fica desatualizado? Na verdade, não.

Eu posso **invalidar o cache** avisando os **métodos que podem gerar alteração nele**, por exemplo **métodos de CRUD**, então com **uma anotação** eu **inválido meu cache** quando **algum método que compromete o estado dele for chamado.**

Na anotação **temos que avisar QUAL CACHE** esse **método** vai **invalidar** e passar um **valor de configuração** sobre o que deve ser limpo (tudo, ou parcialmente):



```java
@GetMapping("/pagina")
@Cacheable(value = "buscaPaginadaUsuario")
public Page<UsuarioDto> buscaFlexivelPaginada(
        @PageableDefault(sort = "id", direction = Sort.Direction.ASC,page = 0, size = 5) Pageable paginacao){

    Page<Usuario> usuarios = usuarioRepository.findAll(paginacao);
    return UsuarioDto.converterPaginado(usuarios);
}


@PostMapping
@Transactional
@CacheEvict(value = "buscaPaginadaUsuario", allEntries = true)
public ResponseEntity<UsuarioDto> cadastrarTopico(@RequestBody @Valid UsuarioForm userForm,
                                                  UriComponentsBuilder uriBuilder){

    Usuario usuario = userForm.converterParaEntidade();
    usuarioRepository.save(usuario);

    URI uri = uriBuilder.path("/usuarios/{id}").buildAndExpand(usuario.getId()).toUri();
    return ResponseEntity.created(uri).body(new UsuarioDto(usuario));
}
```