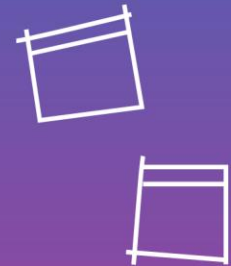




 **VEM SER**
DBC



Informática Básica 2

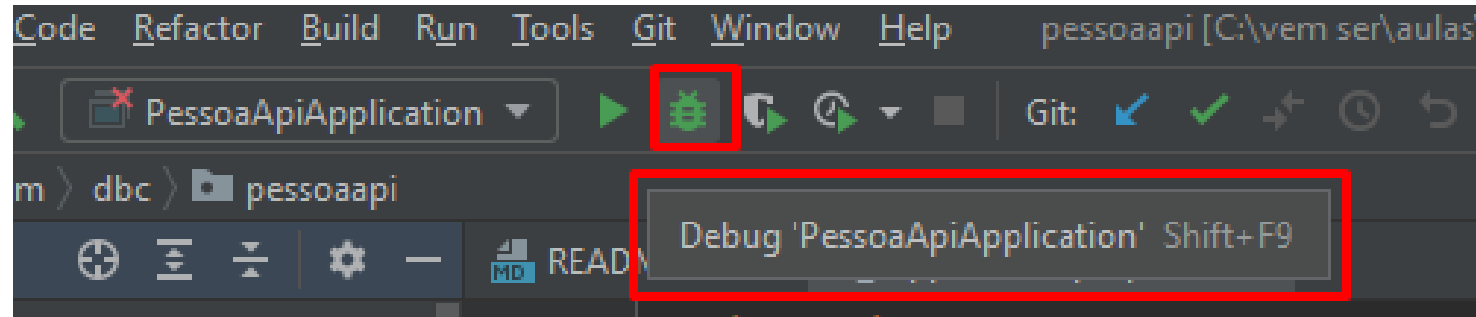
Sumário

- Oportunidade Quality Assurance (QA)
- Debug
- Generics
- Stream
- IntelliJ TIPS
- GIT
- Trabalho Final

Oportunidade Quality Assurance (QA)

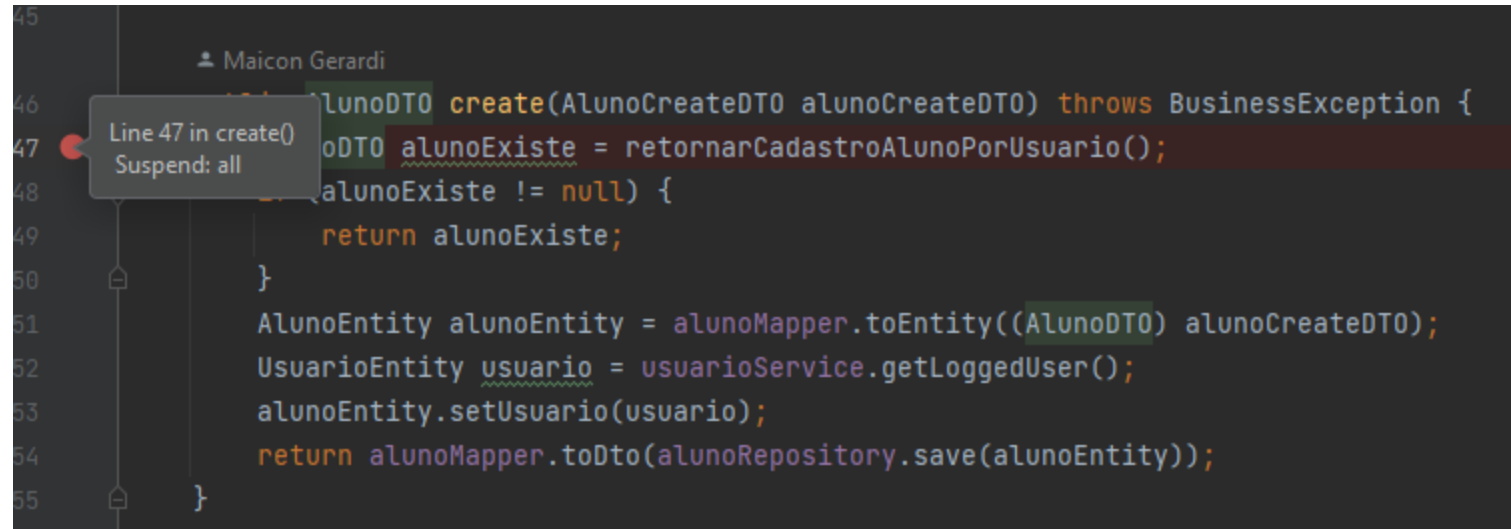
- Automação de Testes
- Quality Assurance (QA)
- Cliente Sicredi
- 10 pessoas
- Selenium / testes manuais

DEBUG



- É possível rodar nosso código linha à linha
- Com ele podemos inspecionar variáveis, trocar valores, fazer modificações em tempo de execução

Breakpoints



```

45
46 Maicon Gerardi
47 AlunoDTO create(AlunoCreatedDTO alunoCreatedDTO) throws BusinessException {
48     AlunoDTO alunoExiste = retornarCadastroAlunoPorUsuario();
49     if (alunoExiste != null) {
50         return alunoExiste;
51     }
52     AlunoEntity alunoEntity = alunoMapper.toEntity((AlunoDTO) alunoCreatedDTO);
53     UsuarioEntity usuario = usuarioService.getLoggedUser();
54     alunoEntity.setUsuario(usuario);
55     return alunoMapper.toDto(alunoRepository.save(alunoEntity));
56 }
  
```

- Podemos colocar um breakpoint padrão (parará ao alcançar a linha de código)
- Podemos criar condições de parada
- A linha parada ainda não foi executada

Inspeção

The screenshot displays an IDE interface with a code editor and a debugger window. The code editor shows a Java method `findAll` in `AlunoService`, which is highlighted with a red box. The debugger window at the bottom shows the execution stack and the current state of the program, also highlighted with a red box.

Code Editor (Highlighted):

```

66     return alunoMapper.toDto(alunoRepository.save(alunoEntity));
67 }
68
69 Maicon Gerardi
70 public PageOut<AlunoCompletoDTO> findAll(String nome, Stack stack,
71 PageRequest pageRequest = PageRequest.of(page, size); page: 0
72 Page<AlunoCompletoDTO> allWithUser = alunoRepository.findAllWithUser
73 return new PageOut<>(allWithUser.getTotalElements(), allWithUser
74 }
75

```

Debugger Window (Highlighted):

Debug: VemserApplication (1) x

Debugger Console: "http-nio-808...main": RUNNING

Stack Trace:

- findAll:70, AlunoService (com.dbc.vemser.ser)
- findAll:47, AlunoController (com.dbc.vemser.ser)
- invoke-1, AlunoController\$\$FastClassBySpringAopProxy (org.springframework.aop.framework.adapter)
- invoke:218, MethodProxy (org.springframework.aop.framework.adapter)
- invokeJoinpoint:793, CglibAopProxy\$CglibMethodInvocation (org.springframework.aop.framework.adapter)
- proceed:163, ReflectiveMethodInvocation (org.springframework.aop.framework.adapter)
- proceed:763, CglibAopProxy\$CglibMethodInvocation (org.springframework.aop.framework.adapter)
- invoke:123, MethodValidationInterceptor (org.springframework.aop.framework.adapter)

Current State:

- this = {AlunoService@14375}
- nome = null
- stack = {Stack@14376} "BACKEND"
- idEdicao = null
- page = {Integer@14377} 0
- size = {Integer@14378} 20
- alunoRepository = {Proxy133@14382} "org.springframework.data.jpa.repository.support.SimpleJpaRepository@cd41078"



Exercício #1

- Baixar o código java e fazer o que é pedido nos comentários...

GENERICs (o famoso «T»)

- Responsável por generalizar códigos redundantes na nossa app
- Como é o caso de casting excessivo
- Este foi introduzido desde o Java SE 5.0



Let's Practice



Exercício #2

- Crie uma aplicação que tenha as seguintes classes:
 - Praia: extensao: int, nome: String
 - PraiaCalma: podeLevarAnimais: boolean
 - PraiaBrava: quantidadeDeOndasPorMinuto: int
- Crie um método main e:
 - Crie uma lista de praias e adicione 2 calmas e 2 bravas
 - Crie um método estático que receba somente lista de praia e imprima a extensão e o nome da praia
 - Crie um método estático que receba qualquer tipo de lista de praias e imprima todas as informações da praia
 - Execute esses métodos para testar seus comportamentos

STREAMS

- **Optional<T>** => classe responsável por tratar os valores nulos
- **.findFirst()** => retorna o primeiro elemento
- **.peek()** => executa o comando passado na expressão
- **.map(obj => outroObj)** => transforma um objeto em outro

PERFORMANCE (FILTROS WHERE E LISTAS)

- Qual a melhor abordagem mais performática para selecionar 1 milhão de registros?
 - 1:
 - Listar todos (select * from tabela)
 - Filtrar a lista (com for ou stream)
 - Retornar
 - 2:
 - Filtrar no banco de dados com where
 - Retornar

INTELLIJ TIPS

- **CTRL + ALT + L** = "Identar" (arranjar) o código
- **CTRL + ALT + O** = Organizar Imports
- Selecionar trecho de código, botão direito, Refactor -> extract method
- **CTRL + ALT + V** = selecionar código e aperta tecla, gera uma variável do resultado
- **CTRL + ALT + C** = selecionar código e aperta tecla, gera uma variável do resultado

GIT

- Operações básicas
- Git Log
- Git Commit
- Git Push
- Git Revert / Reset
- Git Branch
- Git Merge



Trabalho Final Spring Web



Obrigado!

DBC

DIGITAL BUSINESS COMPANY®



 /dbc.company

 /dbccompany

 /dbccompany.com.br

 /company/dbc-company