

# PROGRAMAÇÃO AVANÇADA ORIENTADA A OBJETOS

Prof. Me. Renan Caldeira Menechelli



The top of the slide features a dark red header with a series of overlapping semi-circular shapes. Each semi-circle contains a different pattern: concentric solid lines, concentric dotted lines, or concentric dashed lines.

# CONTEXTO

Autenticação de Usuário

Aula 06

# CONTEXTO

- Isolamento de funções específicas da API;
- Assim, as mudanças não interferem no andamento do projeto e nem em outras funcionalidades;
- Reutiliza código das classes principais
- Cria domínios exclusivos e desacoplados do núcleo do projeto
- A autenticação utiliza majoritariamente a tabela de Usuarios

# CRIANDO O CONTEXTO

- Pacote exclusivo dos contextos
- Dentro dos contextos, criar o pacote de Autenticação
- A autenticação inicial será básica: formada por verificação de e-mail e senha do usuário
- O retorno será um DTO com o nome, cargo, e-mail e perfil do usuário autenticado

```

v  faterc
  v  HelpDesk
    v  contexts
      v  autenticacao
        © AutenticacaoController
        ® AutenticacaoDTO
        © AutenticacaoService
      > controllers
      > dtos
      > entities
      > repositories
      > services
      > utils
```

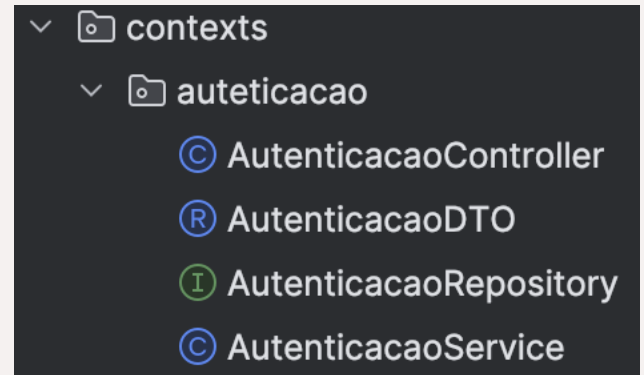
The top of the slide features a dark red background with a series of overlapping semi-circular patterns. These patterns are composed of concentric lines and dots, creating a textured, organic feel.

# AUTENTICAÇÃO

Primeiro Contexto

# AUTENTICAÇÃO

- Basicamente, recebe o e-mail e a senha do usuário.
- Busca o usuário que possui o e-mail específico e confere a senha
- Estando OK, devolve os dados do usuário
- Os elementos a seguir serão direcionados para a criação desse contexto com essa funcionalidade básica.



# DTO

- Os dados que queremos receber / informar para o cliente
- Note que IDs e senha não são revelados

```
public record AutenticacaoDTO( 8 usages
    String nome, no usages
    String email, no usages
    String cargo, no usages
    String perfil no usages
) implements Serializable {
```

```
@Serial
public static final long serialVersionUID = 3982284292937232983L;

public static AutenticacaoDTO valueOf(Usuario usuario) { 2 usages
    if (usuario != null) {
        return new AutenticacaoDTO(
            usuario.getNome(),
            usuario.getEmail(),
            usuario.getCargo(),
            usuario.getPerfil().getPerfil()
        );
    } else return null;
}
```

# REPOSITORY

- Como trata de uma consulta específica, podemos implementar manualmente
- Toda consulta SQL fica nessa classe, separada da regra de negócio
- Nesse momento, estamos usando uma SQL Nativa

```
9 public interface AutenticacaoRepository extends JpaRepository<Usuario, Long> { 1 usage
10
11     @Query(value = "SELECT usu FROM Usuario usu" + 1 usage
12         "WHERE usu.email=:email AND usu.senha=:senha", nativeQuery = true)
13     Optional<Usuario> buscarUsuarioEmailSenhaSQL(String email, String senha);
14
15 }
```



# REPOSITORY / HQL

- Embora útil, a SQL Nativa possui pontos negativos, inclusive de otimização
- Usaremos no projeto o formato HQL
- HQL: a SQL é escrita com base no mapeamento das classes Java
- Nesse momento, o impacto / diferença é praticamente nula
- Em exemplos futuros, iremos notar as diferenças

```
9  public interface AutenticacaoRepository extends JpaRepository<Usuario, Long> { 1 usage
10
11      @Query(value = "SELECT usuario FROM Usuario usuario " + 1 usage
12              "WHERE usuario.email=:email AND usuario.senha=:senha", nativeQuery = false)
13      Optional<Usuario> buscarUsuarioEmailSenha(String email, String senha);
```

# SERVICE

- Implementa a regra de negócio de autenticação
- Nesse primeiro momento, usaremos a autenticação por SQL

```
10  @Service 1 usage
11  public class AutenticacaoService {
12
13      @Autowired
14      AutenticacaoRepository autenticacaoRepository;
15
16      public AutenticacaoDTO autenticarViaSQL(String email, String senha) { no usages
17          Usuario usuario = autenticacaoRepository.buscarUsuarioEmailSenhaSQL(email, senha).orElse(other: null);
18          return AutenticacaoDTO.valueOf(usuario);
19      }
20
21  }
```

# SERVICE

- O Service deve ser alterado para utilizar a HQL

```
11 public class AutenticacaoService {  
12  
13     @Autowired  
14     AutenticacaoRepository autenticacaoRepository;  
15  
16     public AutenticacaoDTO autenticar(String email, String senha) { 1 usage  
17         Usuario usuario = autenticacaoRepository.buscarUsuarioEmailSenha(email, senha).orElse(other: null);  
18         return AutenticacaoDTO.valueOf(usuario);  
19     }
```

# CONTROLLER

- Endpoint específico para realizar a autenticação dos clientes

```
7  @RestController
8  @RequestMapping("/autenticacao")
9  public class AutenticacaoController {
10
11      @Autowired
12      AutenticacaoService autenticacaoService;
13
14      @GetMapping("/{email}/{senha}")
15      public @ResponseBody ResponseEntity<AutenticacaoDTO> autenticar(@PathVariable String email,
16                                                                    @PathVariable String senha) {
17          AutenticacaoDTO autenticacaoDTO = autenticacaoService.autenticar(email, senha);
18          if (autenticacaoDTO != null) {
19              return ResponseEntity.ok().body(autenticacaoDTO);
20          } else
21              return ResponseEntity.badRequest().build();
22      }
23
24 }
```

*Devolve um erro e não  
autoriza o usuário*

# TESTE

- Simulando o erro de autenticação e a autenticação satisfatória

HelpDesk / Autenticacao

GET http://localhost:8080/HelpDesk/api/autenticacao/karina.almeida@fatec.br/karina1234

Params Authorization Headers (6) Body Scripts Settings

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

This request does not have a body

Body Cookies Headers (3) Test Results **400 Bad Request** • 164 ms • 103 B

Raw Preview Visualize

1

GET http://localhost:8080/HelpDesk/api/autenticacao/karina.almeida@fatec.br/karina4321

Params Authorization Headers (6) Body Scripts Settings

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

This request does not have a body

Body Cookies Headers (5) Test Results **200 OK** • 45 ms • 272

{ } JSON Preview Visualize

```
1 {
2   "nome": "Karina Almeida",
3   "email": "karina.almeida@fatec.br",
4   "cargo": "gerente de projetos",
5   "perfil": "GERENTE"
6 }
```

# PROGRAMAÇÃO AVANÇADA ORIENTADA A OBJETOS

Prof. Me. Renan Caldeira Menechelli

