INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PERNAMBUCO

Curso	Disciplina	Turno
Análise e Desenv.de Sistemas	Desenvolvimento para Web IV	Noite
Professor	**Data**	**Pontuação**
Roberto Alencar	04/09/2024	10,0 pontos
Aluno		
CARLOS GABRIEL DE MELO NETO		

01) Considerando as duas classes abaixo, responda:

```java
public class Cliente {

    @ManyToOne
    @JoinColumn(nullable = false)
    private Usuario usuario;

    @Column(nullable = false, length = 100)
    private String nome;

    @Column
    private LocalDate dataNascimento;

    @Column(unique = true)
    private String cpf;

    @Column
    private String fone;

    @Column
    private String foneAlternativo;

}
```

```java
public class ClienteRequest {

    @NotNull(message = "O Nome é de preenchimento obrigatório")
    @NotBlank(message = "O Nome é de preenchimento obrigatório")
    private String nome;

    @NotBlank(message = "O e-mail é de preenchimento obrigatório")
    @Email
    private String email;

    @NotBlank(message = "A senha é de preenchimento obrigatório")
    private String password;

    @NotNull(message = "O CPF é de preenchimento obrigatório")
    @NotBlank(message = "O CPF é de preenchimento obrigatório")
    @CPF
    private String cpf;

    @JsonFormat(pattern = "dd/MM/yyyy")
    private LocalDate dataNascimento;

    private String fone;

    private String foneAlternativo;

}
```

a) Como pode ser observado, as duas classes possuem anotações de validação colocadas em determinados atributos da classe. Qual a diferença entre colocar essas validações na classe `Cliente` e colocar na classe `ClienteRequest` ? (0,5 ponto)

Resposta:

validações na classe `Cliente` = VALIDAÇÃO DE BANCO
validações na classe `ClienteRequest` = VALIDAÇÃO DO CORPO DA REQUISIÇÃO

b) Explique o que significa cada validação colocada nas classes acima. (0,5 ponto)

Resposta:

(unique = true)ESSA VALIDAÇÃO INDICA QUE O ATRIBUTO NÃO PODE SER DUPLICADO NO BANCO, OU SEJA ÚNICO.

(nullable = false)ESSA VALIDAÇÃO INDICA QUE O ATRIBUTO NÃO PODE SER NULO NO BANCO, OU SEJA DEVE SER PREENCHIDO (O ATRIBUTO PASSA A SER OBRIGATÓRIO COM esse nullable = falso).

(length = 100) ESSA VALIDAÇÃO INDICA QUE O ATRIBUTO DEVE TER NO MÁXIMO 100 CARACTERES

@NotNull ESSE DECORATOR OBRIGA QUE O VALOR PARA O ATRIBUTO QUE O TEM A NÃO SER NULO (EX: ""," "…)

@NotBlank ESSE DECORATOR OBRIGA QUE O VALOR PARA O ATRIBUTO QUE O TEM A NÃO SER EM BRANCO (EX: ""," ", null…) (NotBlank já cobre NotNull)

@Email ESSE DECORATOR OBRIGA QUE ESSE ATRIBUTO SEJA PASSADO COM O UM EMAIL VALIDO (EX: @example.com…)

@CPF ESSE DECORATOR OBRIGA QUE ESSE ATRIBUTO SEJA PASSADO COM O UM CPF VALIDO (ELE NÃO SÓ VALIDA A QUANTIDADE DE CARACTERES, COMO TAMBÉM FAZ O CALCULO QUE INDICA SE O CPF É VALIDO OU NÃO, SENDO ASSIM DISPENSA OUTROS DECORATORS, COMO NOTNULL, NOTBLANK… ETC)

@JsonFormat(pattern = "dd/MM/yyyy") VALIDA SE O DATA ESTÁ DE ACORDO COM O "TEMPLATE" ESTABELECIDO

c) Qual seria a validação que precisaríamos acrescentar ao campo `foneAlternativo` para validar o tamanho máximo do campo para não permitir mais que 30 caracteres? (0,5 ponto)

Resposta:

```
@Length(max = 30, message = "não é permitido mais que 30 caracteres")
```

02) Considerando os conceitos aprendidos na aula "C18 - Back-end - Relacionando Entidades", implemente um relacionamento de "um para muitos" bidirecional entre as classes Empresa e Cliente de forma que:

- um cliente tenha uma empresa;
- uma empresa tenha uma lista de clientes

Faça a alteração no código das classes abaixo (1,0 ponto)

```
public class Cliente extends EntidadeAuditavel {

    @Column(nullable = false, length = 100)
    private String nome;

    @Column
    private LocalDate dataNascimento;

    @Column(unique = true)
    private String cpf;

    @Column
    private String fone;

    @Column
    private String foneAlternativo;

    @JoinColumn(name = "cliente_id")
    @ManyToOne()
    @JsonIgnore
    private Cliente cliente;

}
```

```
public class Empresa extends EntidadeAuditavel {

    @Column
    private String site;

    @Column
    private String cnpj;

    @Column
    private String inscricaoEstadual;

    @Column
    private String nomeEmpresarial;

    @Column
    private String nomeFantasia;

    @Column
    private String fone;

    @Column
    private String foneAlternativo;


@OneToMany(mappedBy = "cliente", orphanRemoval = true,
fetch = FetchType.EAGER)
@Column
List<Cliente> clientes;

}
```

03) Considerando o método abaixo, modifique o código para permitir:

- Acesso público ao *endpoint* no Controller que consulta uma empresa por ID. (0,5 ponto)

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

    http
            .httpBasic().disable().csrf().disable().cors().and().sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS).and().exceptionHandling()
            .authenticationEntryPoint(authenticationEntryPoint).and().authorizeRequests()

            .antMatchers(AUTH_WHITELIST).permitAll()
            .authorizeHttpRequests(authorize -> authorize
            .requestMatchers(HttpMethod.GET, "/empresa/*").permitAll()

            .anyRequest()
            .hasAnyAuthority(Usuario.ROLE_CLIENTE, Usuario.ROLE_EMPRESA, Usuario.ROLE_CLIENTE)
             )
            .and().addFilterBefore(
                    new JwtTokenAuthenticationFilter(jwtTokenProvider),
                    UsernamePasswordAuthenticationFilter.class);

    return http.build();
}
```

04) No contexto da aula de Controle de Acesso, qual a diferença entre autenticação, autorização e auditoria? (1,0 ponto)

Resposta:

Autenticação = verificar se o usuario tem acesso, autorização = nivel de acesso que o usuario autenticado tem

05) Durante a implementação do projeto trabalhado ao longo da disciplina nós criamos um arquivo `.env` e o colocamos na raiz do projeto. Responda:

a) Para que serve este arquivo? Qual a vantagem/importância dele? (0,5 ponto)

Resposta:

Proteger credencias de cloud, banco etc, informações que em geral não devem ficar publicas no github, nos usamos esse .env para armazenar essas informações e aplicando assim uma camada de segurança extra mantendo essas informações mais seguras. Na verdade é importante também reforçar que devemos adicionar o .env ao arquivo gitignore para não subir as informações confidenciais para o repositório remoto,

b) No arquivo onde definimos as configurações do projeto, escreva abaixo a linha em que informamos ao spring que o projeto poderá utilizar (ou não) um arquivo `.env` (0,5 ponto)

Resposta:

```
spring.config.import=optional:file:.env[.properties]
```

06) Observando o código abaixo, responda:

"caso ocorra um erro e seja levantado alguma exceção na linha 9, os objetos inseridos nas linhas 4 e 7 serão gravados no banco de dados, pois os comandos são executados antes da linha 9."

A afirmação acima é verdadeira? Justifique sua resposta. (1,0 ponto)

```
1   @Transactional
2   public Cliente save(Cliente cliente) {
3
4       usuarioService.save(cliente.getUsuario());
5
6
7       super.preencherCamposAuditoria(cliente);
8       Cliente clienteSalvo = repository.save(cliente);
9
10      emailService.enviarEmailConfirmacaoCadastroCliente(clienteSalvo);
11
12
        return clienteSalvo;
    }
```

Resposta:

```
Não o , preencherCamposAuditoria garante que todo o save seja feito completamente ou o registro é
desfeito
```

07) Observando o código abaixo, responda:

"Após criada esta interface (`ClienteRepository`), já é possível ter acesso a métodos para consultar um cliente por id, consultar todos os clientes, incluir, alterar e remover um cliente no banco de dados."

A afirmação acima é verdadeira? Justifique sua resposta. (1,0 ponto)

```
1   package br.com.ifpe.oxefood.modelo.cliente;
2
3   import org.springframework.data.jpa.repository.JpaRepository;
4   import org.springframework.data.jpa.repository.JpaSpecificationExecutor;
5
6   public interface ClienteRepository extends JpaRepository<Cliente,
7   Long>,JpaSpecificationExecutor<Cliente> {
8
9
10  }
```

Resposta:

Sim, JpaRepository é uma interface que já tem alguns metodos que já podemos chamar no service da nossa aplicação atraves da injeção de dependencia, nós fazemos a injeção do repository no service e podemos criar os metodos que incluem, alteram, removem e consultam a entidade referenciando os metodos já presentes nessa interface

08) Considerando a classe abaixo:

```
1  public class CategoriaProduto extends EntidadeAuditavel {
2
3      @NotNull
4      @Column(nullable = false, length = 100)
5      private String descricao;
6
7  }
```

a) O código abaixo funciona? Se não funcionar, explique o porquê. (0,5 ponto)

```
1  public interface CategoriaProdutoRepository extends JpaRepository<CategoriaProduto, Long> {
2
3      List<CategoriaProduto> findByChaveEmpresaOrderByNome(String chaveEmpresa);
4
5  }
```

Resposta:

A propriedade 'chaveEmpresa' não existe em Cliente, logo não é possível executar essa consulta, por isso o programa não compila

b) O código abaixo funciona? Se não funcionar, explique o porquê. (0,5 ponto)

```
1  public interface CategoriaProdutoRepository extends JpaRepository<CategoriaProduto, Long> {
2
3      List<CategoriaProduto> findByChaveEmpresaOrderByDescricao();
4
5  }
```

Resposta:

A propriedade 'chaveEmpresa' não existe em Cliente, logo não é possível executar essa consulta, por isso o programa não compila.

09) Na aula de controle de acesso do projeto do front-end, implementamos um componente para restringir o acesso não autenticado às telas do sistema, posteriormente esse componente foi adicionado em cada definição `<Route>` no arquivo `Rotas.js`. Qual foi o componente criado e como ele foi utilizado no arquivo `Rotas.js` ? (1,0 ponto)

Resposta:

ProtectedRoute

10) O que é uma API WEB considerada RESTful? (1,0 ponto)

Resposta:

Em poucas palavras é uma aplicação que permite que sistemas possam se comunicar entre si,

permissão para criar, deletar, editar e ler informações.