
Desenvolvimento WEB

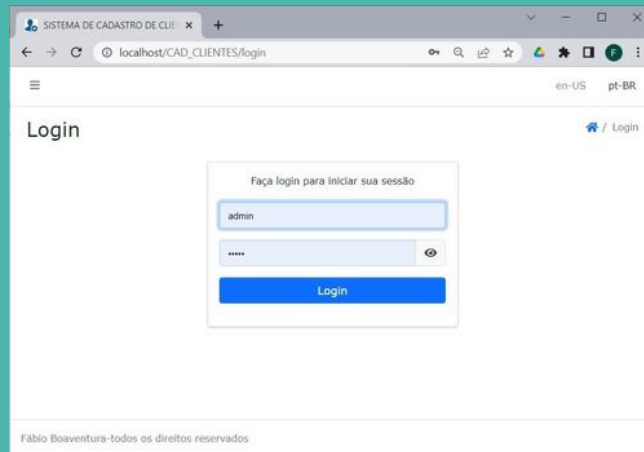
-

Full Stack Completo: Java + React

Controle de Acesso

Autenticação, Autorização e Auditoria

Parte 02



The screenshot shows a web browser window with the title 'SISTEMA DE CADASTRO DE CLIENTES'. The address bar displays 'localhost/CAD_CLIENTES/login'. The page has a header with a hamburger menu icon, language selectors for 'en-US' and 'pt-BR', and a 'Login' link. The main content area is titled 'Login' and contains a form with the instruction 'Faça login para iniciar sua sessão'. The form has two input fields: the first contains 'admin' and the second contains masked characters '****' with an eye icon for toggling visibility. Below the inputs is a blue 'Login' button. The footer of the page reads 'Fábio Boaventura-todos os direitos reservados'.

SISTEMA DE CADASTRO DE CLIENTES

localhost/CAD_CLIENTES/login

en-US pt-BR

Login

Faça login para iniciar sua sessão

admin

Login

Fábio Boaventura-todos os direitos reservados

Triplo A



- ~~• Autenticação: Quem é?~~
- Autorização: Pode fazer?
- Auditoria: O que fez?

Implementando a Autorização com Spring Security

- A autorização define o que pode e o que não pode acessar
- Utilizada para verificar se um determinado usuário previamente autenticado possui permissão para usar, manipular ou executar o recurso em questão.
 - **Simples**: uma vez autenticado, o usuário pode acessar tudo
 - **Permissões**: o usuário faz parte de um perfil com permissões específicas

De modo geral, controle de acesso consiste na **verificação de autorização de um usuário autenticado**

Implementando a Autorização com Spring Security

- Defina quais os perfis de acesso poderão ser usados no seu sistema

```
...  
  
public class Usuario extends EntidadeNegocio implements UserDetails {  
  
    public static final String ROLE_CLIENTE = "CLIENTE"; // Realizar compras no sistema  
    public static final String ROLE_EMPRESA_ADMIN = "EMPRESA ADMIN"; // READ, DELETE, WRITE, UPDATE.  
    public static final String ROLE_EMPRESA_USER = "EMPRESA_USER"; // READ, WRITE, UPDATE.  
  
    @Column(nullable = false, unique = true)  
    private String username;  
  
    @JsonIgnore  
    @Column(nullable = false)  
    private String password;  
  
    ...  
}
```

Implementando a Autorização com Spring Security

- Modifique a entidade `Empresa` para que esta se relacione com a entidade `Usuario` e seja possível utilizar uma empresa para logar no sistema:

```
...
public class Empresa extends EntidadeAuditavel {

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

    @Column
    private String site;

    @Column
    private String cnpj;

    ...
}
```

Implementando a Autorização com Spring Security

- Atualize a classe `EmpresaRequest` acrescentando o código **em vermelho** ao lado:

```
public class EmpresaRequest {  
  
    private String email;  
    private String password;  
    private String perfil;  
  
    private String site;  
    private String cnpj;  
    private String inscricaoEstadual;  
    private String nomeEmpresarial;  
    private String nomeFantasia;  
    private String fone;  
    private String foneAlternativo;  
  
    public Empresa build() {  
  
        Empresa empresa = Empresa.builder()  
            .usuario(buildUsuario())  
            .site(site)  
            .cnpj(cnpj)  
            .inscricaoEstadual(inscricaoEstadual)  
            .nomeEmpresarial(nomeEmpresarial)  
            .nomeFantasia(nomeFantasia)  
            .fone(fone)  
            .foneAlternativo(foneAlternativo)  
            .build();  
  
        return empresa;  
    }  
  
    public Usuario buildUsuario() {  
  
        return Usuario.builder()  
            .username(email)  
            .password(password)  
            .build();  
    }  
}
```

Implementando a Autorização com Spring Security

- Implemente na classe `EmpresaController` lógica para setar o perfil da empresa:

```
public class EmpresaController {

    @Autowired
    private EmpresaService empresaService;

    @PostMapping
    public ResponseEntity<Empresa> save(@RequestBody @Valid EmpresaRequest request) {

        Empresa empresa = request.build();

        if (request.getPerfil() != null && !"".equals(request.getPerfil())) {

            if (request.getPerfil().equals("EMPRESA_USER")) {

                empresa.getUsuario().getRoles().add(Usuario.ROLE_EMPRESA_USER);

            } else if (request.getPerfil().equals("EMPRESA_ADMIN")) {

                empresa.getUsuario().getRoles().add(Usuario.ROLE_EMPRESA_ADMIN);

            }

        }

        Empresa empresaCriada = empresaService.save(empresa);
        return new ResponseEntity<Empresa>(empresaCriada, HttpStatus.CREATED);

    }

}
```


Implementando a Autorização com Spring Security

- Atualize a classe EmpresaService:

```
@Service
public class EmpresaService {

    @Autowired
    private EmpresaRepository repository;

    @Autowired
    private UsuarioService usuarioService;

    @Transactional
    public Empresa save(Empresa empresa) {

        usuarioService.save(empresa.getUsuario());

        empresa.setHabilitado(Boolean.TRUE);
        empresa.setVersao(1L);
        empresa.setDataCriacao(LocalDate.now());

        return repository.save(empresa);
    }
}
```

Implementando a Autorização com Spring Security

- Implemente as restrições de autorização de acesso no método `securityFilterChain` da classe `SecurityConfiguration`. Através do método `hasAnyAuthority()` é possível definir quais perfis de usuário tem acesso a determinados endpoints do sistema:

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

    http.csrf()
        .disable()
        .authorizeHttpRequests()

        .requestMatchers(HttpMethod.POST, "/api/cliente").permitAll()
        .requestMatchers(HttpMethod.POST, "/api/auth").permitAll()

        //Configuração de autorizações de acesso para Produto

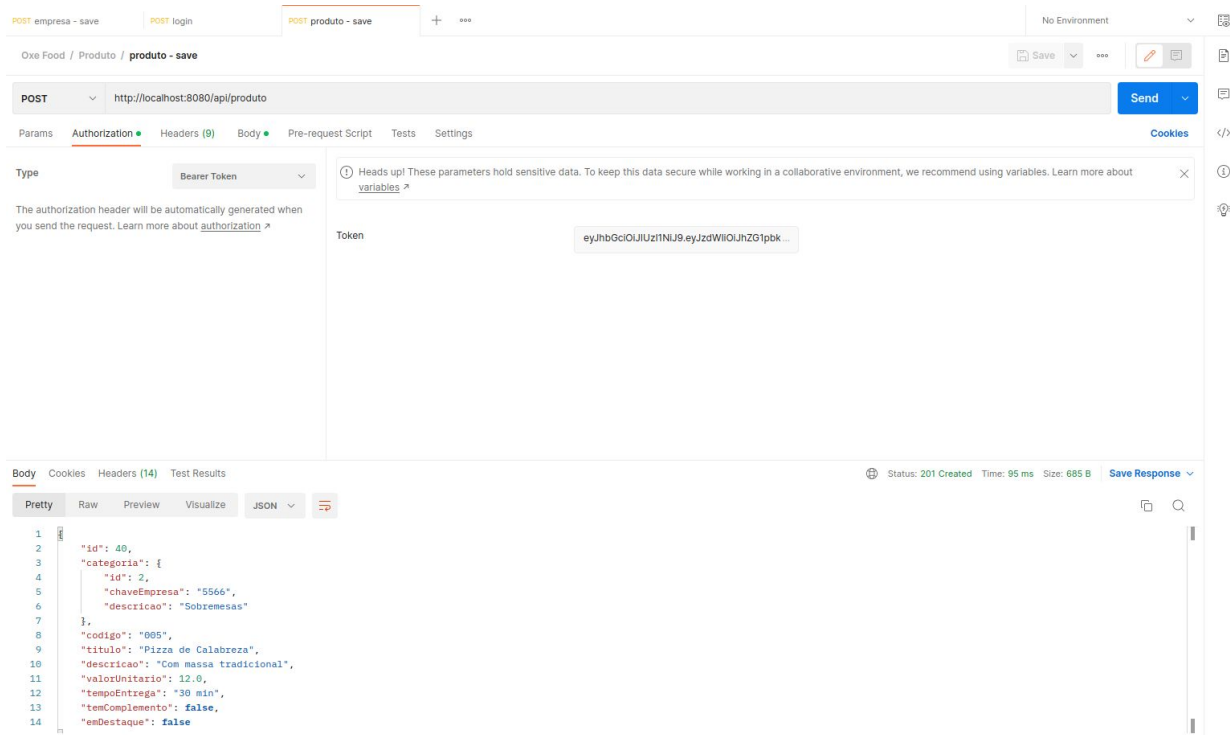
        .requestMatchers(HttpMethod.POST, "/api/produto").hasAnyAuthority(Usuario.ROLE_EMPRESA_ADMIN, Usuario.ROLE_EMPRESA_USER) //Cadastro de produto
        .requestMatchers(HttpMethod.PUT, "/api/produto/*").hasAnyAuthority(Usuario.ROLE_EMPRESA_ADMIN, Usuario.ROLE_EMPRESA_USER) //Alteração de produto
        .requestMatchers(HttpMethod.DELETE, "/api/produto/*").hasAnyAuthority(Usuario.ROLE_EMPRESA_ADMIN) //Exclusão de produto
        .requestMatchers(HttpMethod.GET, "/api/produto/").hasAnyAuthority(Usuario.ROLE_CLIENTE, Usuario.ROLE_EMPRESA_ADMIN, Usuario.ROLE_EMPRESA_USER) //Consulta de produto

        .anyRequest()
        .authenticated()
        .and()
        .sessionManagement()
        .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
        .and()
        .authenticationProvider(authenticationProvider)
        .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);

    return http.build();
}
```

Implementando a Autorização com Spring Security

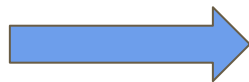
- No postman, salve empresas com perfis diferentes; faça o login no sistema com um cliente e com as empresas de perfis diferentes e teste as rotas protegidas de `ProdutoController` configuradas no `SecurityConfig`.



Dúvidas



Triplo A



- ~~• Autenticação: Quem é?~~
- ~~• Autorização: Pode fazer?~~
- Auditoria: O que fez?

Implementando a Auditoria

“... suporte para acompanhar de forma transparente quem criou ou alterou uma entidade e o momento em que isso aconteceu.”

- Registrar atividades realizadas
- Quem, quando, em qual IP
- Logs, evidências
- Formas de rastrear, filtrar e pesquisar

Implementando a Auditoria

“... suporte para acompanhar de forma transparente quem criou ou alterou uma entidade e o momento em que isso aconteceu.”

- Spring Data JPA Auditing:
 - @Version
 - @CreatedBy
 - @CreatedDate
 - @LastModifiedBy
 - @LastModifiedDate

Implementando a Auditoria

- Em nosso projeto, os atributos de auditoria estão presentes na superclasse das entidades de domínio, chamada de `EntidadeAuditavel`.
- Verifique sua classe `EntidadeAuditavel`, e ajuste as anotações dos atributos conforme código abaixo **em vermelho**.

```
public class Empresa extends EntidadeAuditavel {  
  
    @ManyToOne  
    @JoinColumn(nullable = false)  
    private Usuario usuario;  
  
    @Column  
    private String site;  
  
    @Column  
    private String cnpj;  
  
    ...  
}
```

```
public abstract class EntidadeAuditavel extends EntidadeNegocio {  
  
    @JsonIgnore  
    @Version  
    private Long versao;  
  
    @JsonIgnore  
    @CreatedDate  
    private LocalDate dataCriacao;  
  
    @JsonIgnore  
    @LastModifiedDate  
    private LocalDate dataUltimaModificacao;  
  
    @CreatedBy  
    @ManyToOne  
    @JoinColumn  
    private Usuario criadoPor;  
  
    @LastModifiedBy  
    @ManyToOne  
    @JoinColumn  
    private Usuario ultimaModificacaoPor;  
  
}
```


Implementando a Auditoria

- Na classe `UsuarioService` acrescente o método `obterUsuarioLogado` (código abaixo em vermelho).

```
@Service
public class UsuarioService implements UserDetailsService {

    ...

    @Autowired
    private JwtService jwtService;

    ...

    public Usuario obterUsuarioLogado(HttpServletRequest request) {

        Usuario usuarioLogado = null;
        String authHeader = request.getHeader("Authorization");

        if (authHeader != null) {

            String jwt = authHeader.substring(7);
            String userEmail = jwtService.extractUsername(jwt);
            usuarioLogado = findByUsername(userEmail);
            return usuarioLogado;
        }

        return usuarioLogado;
    }

    ...
}
```

Implementando a Auditoria

- Na classe `ClienteController` ajuste os métodos de incluir e alterar para passar para o service o usuário logado:

```
public class ClienteController {  
  
    ...  
  
    @Autowired  
    private UsuarioService usuarioService;  
  
    ...  
  
    @PostMapping  
    public ResponseEntity<Cliente> save(@RequestBody @Valid ClienteRequest clienteRequest, HttpServletRequest request) {  
  
        Cliente cliente = clienteService.save( clienteRequest.build(), usuarioService.obterUsuarioLogado(request));  
        return new ResponseEntity<Cliente>(cliente, HttpStatus.CREATED);  
    }  
  
    ...  
  
    @PutMapping("/{id}")  
    public ResponseEntity<Cliente> update(@PathVariable("id") Long id, @RequestBody ClienteRequest clienteRequest, HttpServletRequest request) {  
  
        clienteService.update(id, clienteRequest.build(), usuarioService.obterUsuarioLogado(request));  
        return ResponseEntity.ok().build();  
    }  
  
    ...  
}
```

Implementando a Auditoria

- Na classe `ClienteService` já estamos setando alguns campos de auditoria (habilitado, versão e `dataCriacao`).
- Agora acrescente o código para setar o usuário logado nos métodos de incluir e alterar:

```
@Transactional
public Cliente save(Cliente cliente , Usuario usuarioLogado) {

    usuarioService.save(cliente.getUsuario());

    cliente.setHabilitado(Boolean.TRUE);
    cliente.setVersao(1L);
    cliente.setDataCriacao(LocalDate.now());
    cliente.setCriadoPor(usuarioLogado);

    Cliente clienteSalvo = repository.save(cliente);

    //emailService.enviarEmailConfirmacaoCadastroCliente(clienteSalvo);

    return clienteSalvo;
}
```

```
@Transactional
public void update(Long id, Cliente clienteAlterado, Usuario usuarioLogado) {

    Cliente cliente = repository.findById(id).get();
    cliente.setNome(clienteAlterado.getNome());
    cliente.setDataNascimento(clienteAlterado.getDataNascimento());
    cliente.setCpf(clienteAlterado.getCpf());
    cliente.setFoneCelular(clienteAlterado.getFoneCelular());
    cliente.setFoneFixo(clienteAlterado.getFoneFixo());

    cliente.setVersao(cliente.getVersao() + 1);
    cliente.setDataUltimaModificacao(LocalDate.now());
    cliente.setUltimaModificacaoPor(usuarioLogado);

    repository.save(cliente);
}
```

Dúvidas



Exercício

Implemente o controle de permissão e o controle de auditoria do seu projeto.



The image features a white background with two large, solid green abstract shapes. One shape is a semi-circle on the left side, and the other is a more complex, organic shape on the right side. Centered between these shapes is the text "Obrigado !".

Obrigado !