Documento de Visão

Projeto: EducaSis

Autores: Grupo 6

Marcos Gabriel de Albuquerque da Silva - UC24102315

Michel Borges de Araújo - UC24102016

Saulo Ribeiro Dourado Araújo - UC24101062

Controle de Versão:

Versão	Data	Descrição	Autor
1.0	22/03/2025	Primeira versão do documento.	Todos
1.1	14/04/2025	Revisão do tópico 4	Saulo
1.2	17/04/2025	Revisão do tópico 1	Marcos
1.3	20/04/2025	Revisão final	Michel

Sumário

Sumário

1. Introdução

Este documento de visão tem como objetivo apresentar a visão geral do sistema de login desenvolvido com SpringBoot e MySQL, integrado a um sistema escolar.

1.1. Objetivos do Projeto

O EducaSis é um *software* feito com o intuito de simular um site de uma instituição escolar, permitindo cadastro, login, publicação e realização de tarefas, diferenciando professores de alunos.

1.2. Escopo do Projeto

Será entregue, junto a este documento, o repositório do GitHub com o código-fonte. Além disso, o grupo responsável fará uma apresentação explicando as principais funcionalidades da aplicação.

2. Visão Geral

O sistema escolar oferecerá cadastro e login de usuários, diferenciando professores e alunos. Após autenticação, o usuário poderá acessar sua área pessoal para gerenciar tarefas de sua lista de afazeres.

2.1. Perfil dos Usuários

Usuário	Descrição
Professor	Publica atividades por meio do sistema
Aluno	Realiza as atividades publicadas

2.2. Requisitos Funcionais

Funcionalidade	Prioridade
Cadastro & Login	Alta
CRUD dos perfis	Alta

Publicação de atividades	Alta
CRUD das atividades	Alta

2.3. Restrições

- Uso da internet;
- Conhecimento em Língua Portuguesa;

3. Histórias do Usuário

Cenário 1: Cadastro de Conta

Dado que o usuário está na página de cadastro

Quando ele informa um nome, um e-mail não cadastrado e uma senha

Então a conta deve ser criada com sucesso e uma mensagem de confirmação deve ser exibida.

Cenário 2: Cadastro de Conta com e-mail já existente

Dado que já existe uma conta cadastrada com determinado e-mail

Quando o usuário tentar se cadastrar com o mesmo e-mail

Então o sistema deve exibir uma mensagem de erro informando que o e-mail já está em uso.

Cenário 3: Login de conta

Dado que o usuário está na página de login

Quando ele informa e-mail e senha válidos

Então o sistema deve permitir o acesso e redirecioná-lo para a página inicial.

Cenário 4: Login de conta com credenciais inválidas

Dado que o usuário está na página de login

Quando ele informa e-mail ou senha inválidos

Então o sistema deve exibir uma mensagem de erro e permanecer na tela de login.

Cenário 5: Listar contas

Dado que o administrador está autenticado no sistema

Quando ele acessa a opção de visualizar usuários

Então o sistema deve exibir uma lista com nome e e-mail de todos os usuários cadastrados.

Cenário 6: Editar conta

Dado que o usuário está autenticado e acessa a página de edição de perfil

Quando ele altera o nome ou a senha e confirma a operação

Então o sistema deve atualizar os dados e exibir uma mensagem de confirmação.

Cenário 7: Excluir conta

Dado que o usuário deseja excluir sua conta e está autenticado

Quando ele solicita a exclusão e confirma a ação

Então o sistema deve apagar a conta e redirecioná-lo para a página inicial.

Cenário 8: Criar tarefa

Dado que o usuário está autenticado e acessa a tela de cadastro de tarefa

Quando ele informa um nome para a tarefa

Então o sistema deve salvar a tarefa e exibi-la na lista.

Cenário 9: Listar tarefas

Dado que o usuário está autenticado

Quando ele acessa a opção de listar tarefas

Então o sistema deve exibir todas as tarefas cadastradas por ele.

Cenário 10: Editar tarefa

Dado que o usuário está autenticado e possui tarefas cadastradas

Quando ele altera o nome de uma tarefa

Então sistema deve atualizar as informações e exibi-las.

Cenário 11: Excluir tarefa

Dado que o usuário está autenticado e possui tarefas cadastradas

Quando ele solicita a exclusão de uma tarefa e confirma a operação
Então o sistema deve apagar a tarefa e atualizar a lista exibida.

Cenário 12: Realizar tarefa
Dado que o usuário está autenticado e possui tarefas cadastradas
Quando ele altera o status de uma tarefa
Então sistema deve atualizar as informações e exibi-la como feita.

4. Planejamento de Testes

Foram feitos testes unitários, de API e E2E para simular a experiência do usuário ao usar o *software.* As funcionalidades testadas incluem:

- Cadastro e Login
- CRUD de contas
- CRUD de tarefas
- Respostas corretas da API via IntelliJ
- Fluxo completo via IntelliJ

4.1. Casos de Teste

ID	Descrição	Passos	Resultado Esperado	Resultado Obtido	Critério de sucesso
CT-001	Testar cadastro de usuário com dados válidos.	1. Acessar a página de cadastro 2. Inserir e-mail válido e senha válida 3. Clicar no botão “Criar conta”.	A conta é criada e os dados são adicionados ao banco de dados.	A conta é criada e os dados são adicionados ao banco de dados.	Conta criada com sucesso, confirmação exibida e registro no banco de dados.
CT-002	Testar cadastro de usuário com e-mail já existente.	1. Acessar a página de cadastro	Mensagem de erro informando que o e-mail já está em uso.	Mensagem de erro informando que o e-mail já está em uso.	Sistema bloqueia o cadastro.

		2. Inserir e-mail já cadastrado 3. Clicar no botão “Criar conta”.			
CT-003	Testar login com credenciais válidas.	1. Acessar a página de login 2. Inserir e-mail válido e senha válida 3. Clicar no botão “Criar conta”.	Acesso concedido e redirecionamento para página inicial.	Acesso concedido e redirecionamento para página inicial.	Usuário autenticado e redirecionado corretamente.
CT-004	Testar login com credenciais inválidas.	1. Acessar a página de login 2. Inserir e-mail válido e senha válida 3. Clicar no botão “Criar conta”.	Permanência na tela de login.	Permanência na tela de login.	Sistema impede acesso e informa erro corretamente.
CT-005	Testar listagem de contas.	1. Acessar a página do admin 2. Visualizar usuários	Exibir e-mail de todos os usuários cadastrados	Exibir e-mail de todos os usuários cadastrados	Listagem completa e atualizada exibida corretamente.
CT-006	Testar edição de conta.	1. Acessar a página de edição de perfil 2. Alterar e-mail ou senha	Dados atualizados	Dados atualizados	Dados alterados no banco e mensagem de

		3. Confirmar			sucesso exibida.
CT-007	Testar exclusão de conta.	1. Acessar a página de edição de perfil 2. Excluir a conta 3. Confirmar	Conta removida e redirecionamento para tela inicial	Conta removida e redirecionamento para tela inicial	Conta excluída do banco e sistema confirma operação.
CT-008	Testar cadastro de tarefa.	1. Acessar a tela de tarefas 2. Informar título da tarefa 3. Confirmar	Tarefa adicionada à lista e exibida corretamente.	Tarefa adicionada à lista e exibida corretamente.	Tarefa aparece na listagem com dados informados.
CT-009	Testar listagem de tarefas.	1. Acessar a tela de tarefas	Exibir todas as tarefas cadastradas pelo usuário logado	Exibir todas as tarefas cadastradas pelo usuário logado	Lista completa e atualizada, exibida corretamente.
CT-010	Testar edição de tarefa.	1. Acessar a tela de tarefas 2. Clicar para editar tarefas 3. Alterar nome 4. Confirmar	Tarefa atualizada e mensagem de confirmação exibida.	Tarefa atualizada e mensagem de confirmação exibida.	Tarefa salva no banco e alteração visível na interface.
CT-011	Testar exclusão de tarefa.	1. Acessar a tela de tarefas 2. Clicar para excluir tarefa 3. Confirmar	Tarefa removida da lista	Tarefa removida da lista	Tarefa apagada do banco e sumida da interface.
CT-012	Testar conclusão de tarefa.	1. Acessar a tela de tarefas 2. Alterar o status de uma tarefa	Tarefa exibida como “feita” na listagem.	Tarefa exibida como “feita” na listagem.	Status atualizado e indicado na interface.

CT-013	Teste de API - Autenticação de Login	1. Acessar a interface do IntelliJ 2. Realizar a autenticação do Login	Se o login for válido, o usuário vai para a página de tarefas. Se não, ele permanece na página.	Se o login for válido, o usuário vai para a página de tarefas. Se não, ele permanece na página.	Autenticação correta dependendo do caso
CT-014	Teste E2E - Fluxo completo	1. Acessar a aplicação 2. Realizar cadastro 3. Fazer login 4. Cadastrar tarefa 5. Editar tarefa 6. Concluir tarefa 7. Excluir tarefa	Todas as ações executadas com sucesso, conforme previsto nos requisitos.	Todas as ações executadas com sucesso, conforme previsto nos requisitos.	Fluxo completo realizado sem erros ou falhas.

5. Evidências dos Testes

CT-001:

```
// Cenário 1: Cadastro de Conta
@Test  new *
void testCadastroConta() {
    when(usuarioRepository.save(usuarioTeste)).thenReturn(usuarioTeste);

    Usuario resultado = usuarioRepository.save(usuarioTeste);

    assertNotNull(resultado);
    assertEquals( expected: "teste@email.com", resultado.getEmail());
    verify(usuarioRepository, times( wantedNumberOfInvocations: 1)).save(usuarioTeste);
}
```

```
✓ UsuarioRepositoryTes 1 sec 861 ms
  ✓ testLogin_UsuarioE 1 sec 659 ms
  ✓ testLogin_CredenciaisInv 19 ms
  ✓ testEditarConta() 11 ms
  ✓ testListarContas() 130 ms
  ✓ testCadastroConta_Emai 29 ms
  ✓ testExcluirConta() 8 ms
  ✓ testCadastroConta() 5 ms

✓ Tests passed: 7 of 7 tests – 1 sec 861 ms

WARNING: A terminally deprecated method in sun
WARNING: sun.misc.Unsafe::objectFieldOffset ha
WARNING: Please consider reporting this to the
WARNING: sun.misc.Unsafe::objectFieldOffset wi

Process finished with exit code 0
```

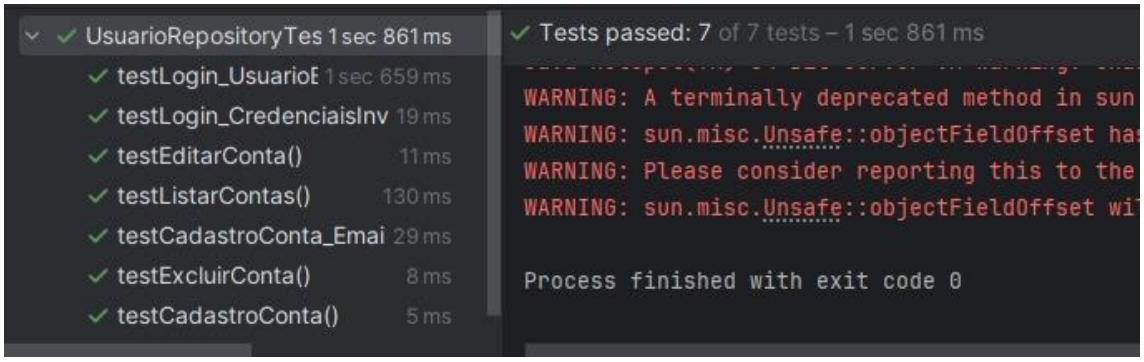
CT-002:

```
// Cenário 2: Cadastro de Conta com e-mail já existente
@Test  new *
void testCadastroConta_EmailExistente() {
    when(usuarioRepository.findByEmail("teste@email.com")).thenReturn(Optional.of(usuarioTeste));

    Optional<Usuario> existente = usuarioRepository.findByEmail("teste@email.com");

    assertTrue(existente.isPresent());
    verify(usuarioRepository, times( wantedNumberOfInvocations: 1)).findByEmail("teste@email.com");
}
```

```
✓ UsuarioRepositoryTes 1 sec 861 ms
  ✓ testLogin_UsuarioE 1 sec 659 ms
  ✓ testLogin_CredenciaisInv 19 ms
  ✓ testEditarConta() 11 ms
  ✓ testListarContas() 130 ms
  ✓ testCadastroConta_Emai 29 ms
  ✓ testExcluirConta() 8 ms
  ✓ testCadastroConta() 5 ms

✓ Tests passed: 7 of 7 tests – 1 sec 861 ms

WARNING: A terminally deprecated method in sun
WARNING: sun.misc.Unsafe::objectFieldOffset ha
WARNING: Please consider reporting this to the
WARNING: sun.misc.Unsafe::objectFieldOffset wi

Process finished with exit code 0
```

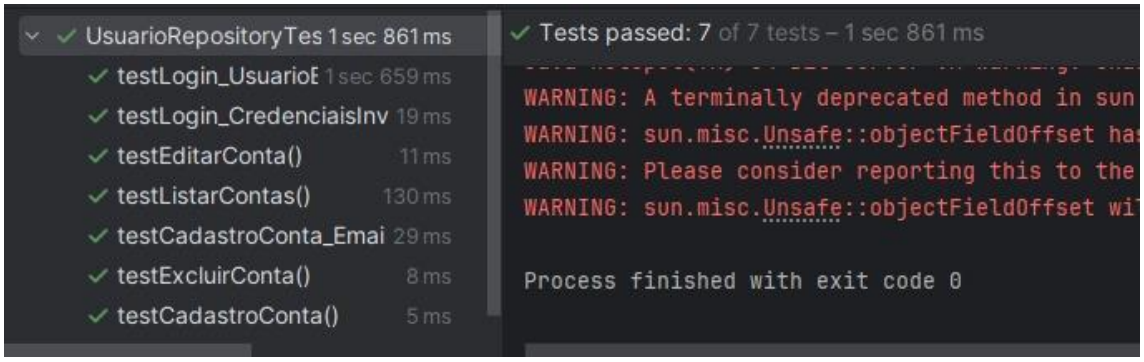
CT-003:

```
// Cenário 3: Login de conta
@Test  Marcos Albuquerque *
void testLogin_UsuarioExistente() {
    when(usuarioRepository.login( email: "teste@email.com",  senha: "123456")).thenReturn(usuarioTeste);

    Usuario resultado = usuarioRepository.login( email: "teste@email.com",  senha: "123456");

    assertNotNull(resultado);
    assertEquals( expected: "teste@email.com", resultado.getEmail());
    verify(usuarioRepository, times( wantedNumberOfInvocations: 1)).login( email: "teste@email.com",  senha: "123456");
}
```



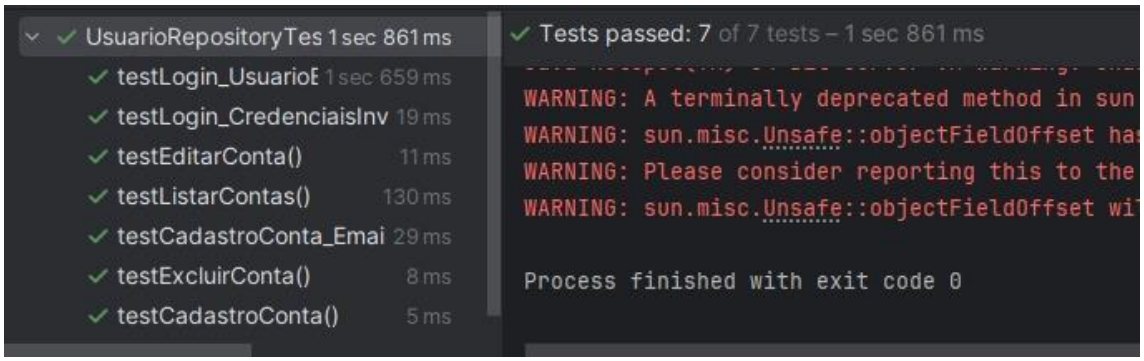
CT-004:

```
// Cenário 4: Login de conta com credenciais inválidas
@Test  Marcos Albuquerque *
void testLogin_CredenciaisInvalidas() {
    when(usuarioRepository.login( email: "email@invalido.com",  senha: "senhaErrada")).thenReturn( t: null);

    Usuario resultado = usuarioRepository.login( email: "email@invalido.com",  senha: "senhaErrada");

    assertNull(resultado);
    verify(usuarioRepository, times( wantedNumberOfInvocations: 1)).login( email: "email@invalido.com",  senha: "senhaErrada");
}
```



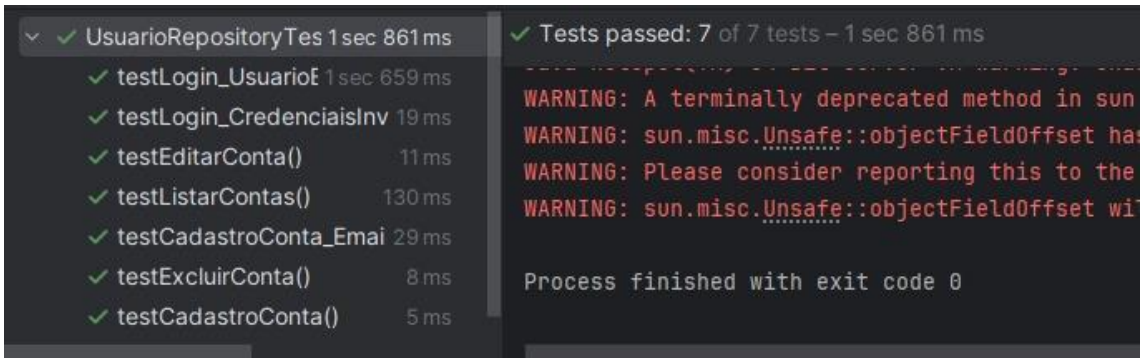
CT-005:

```java
// Cenário 5: Listar contas
@Test  new *
void testListarContas() {
    Usuario outroUsuario = new Usuario();
    outroUsuario.setEmail("outro@email.com");
    outroUsuario.setSenha("654321");

    when(usuarioRepository.findAll()).thenReturn(Arrays.asList(usuarioTeste, outroUsuario));

    List<Usuario> usuarios = (List<Usuario>) usuarioRepository.findAll();

    assertEquals( expected: 2, usuarios.size());
    verify(usuarioRepository, times( wantedNumberOfInvocations: 1)).findAll();
}
```



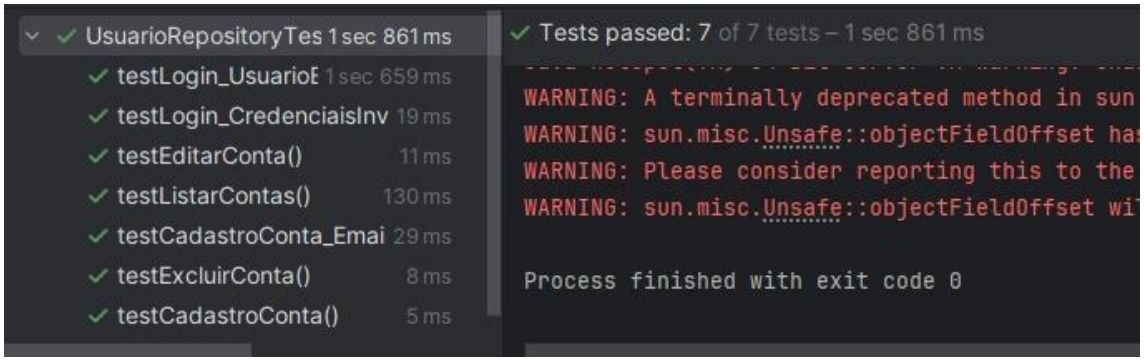
CT-006:

```java
// Cenário 6: Editar conta
@Test  new *
void testEditarConta() {
    Usuario atualizado = new Usuario();
    atualizado.setId(1L);
    atualizado.setEmail("novo@email.com");
    atualizado.setSenha("novaSenha");

    when(usuarioRepository.save(atualizado)).thenReturn(atualizado);

    Usuario resultado = usuarioRepository.save(atualizado);

    assertEquals( expected: "novo@email.com", resultado.getEmail());
    verify(usuarioRepository, times( wantedNumberOfInvocations: 1)).save(atualizado);
}
```



CT-007:

```
    // Cenário 7: Excluir conta
    @Test  new *
    void testExcluirConta() {
        doNothing().when(usuarioRepository).deleteById(1L);

        usuarioRepository.deleteById(1L);

        verify(usuarioRepository, times( wantedNumberOfInvocations: 1)).deleteById(1L);
    }
}
```



CT-008:


Lista de Afazeres
Adicione a sua tarefa:
Trabalho teste de software
+
Pesquisar:
Buscar
Filtrar:
Todos

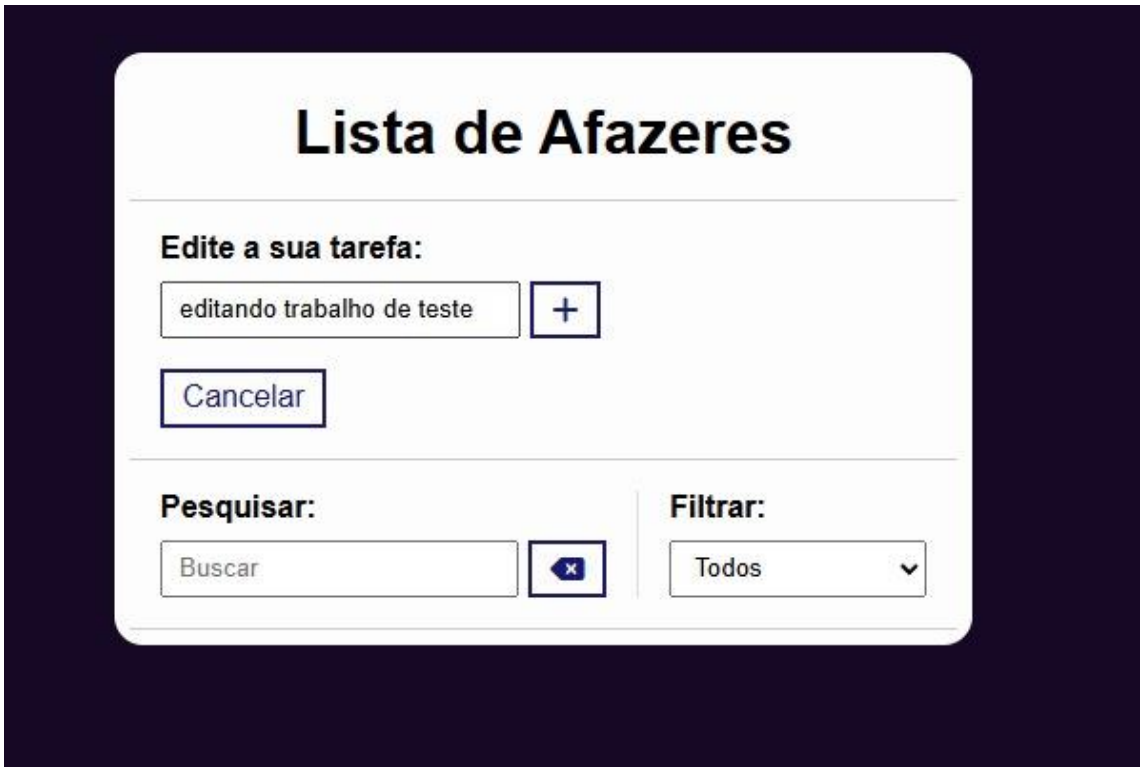

Lista de Afazeres
Adicione a sua tarefa:
O que você vai fazer?
+
Pesquisar:
Buscar
Filtrar:
Todos
Trabalho teste de software

CT-009:


Lista de Afazeres
Adicione a sua tarefa:
O que você vai fazer?
+
Pesquisar:
Buscar
Filtrar:
Todos
Trabalho teste de software
tarefa 02
tarefa 03
tarefa 04

CT-010:


Lista de Afazeres
Adicione a sua tarefa:
O que você vai fazer?
+
Pesquisar:
Buscar
Filtrar:
Todos
Trabalho teste de software
tarefa 02
tarefa 03
tarefa 04


Lista de Afazeres
Edite a sua tarefa:
editando trabalho de teste
+
Cancelar
Pesquisar:
Buscar
Filtrar:
Todos


Lista de Afazeres
Adicione a sua tarefa:
O que você vai fazer?
+
Pesquisar:
Buscar
Filtrar:
Todos
editando trabalho de teste
tarefa 02
tarefa 03
tarefa 04

CT-011:


Lista de Afazeres
Adicione a sua tarefa:
O que você vai fazer?
+
Pesquisar:
Buscar
Filtrar:
Todos
editando trabalho de teste
tarefa 02
tarefa 03
tarefa 04

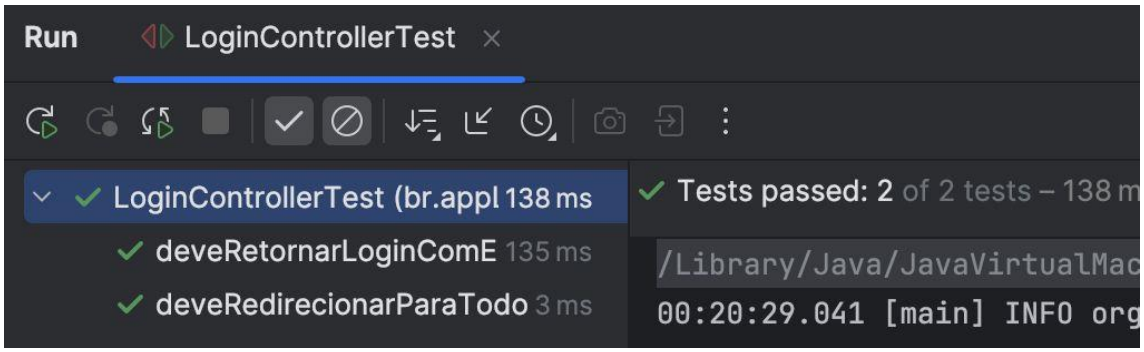

Lista de Afazeres
Adicione a sua tarefa:
O que você vai fazer?
+
Pesquisar:
Buscar
Filtrar:
Todos
tarefa 02
tarefa 03
tarefa 04

CT-012:

CT-013:

```
@Test  new *
void deveRedirecionarParaTodolist_QuandoLoginForValido() throws Exception {
    // simula um usuário encontrado
    Usuario usuarioMock = new Usuario();
    usuarioMock.setEmail("teste@email.com");
    usuarioMock.setSenha("123456");

    when(usuarioRepository.login( email: "teste@email.com",  senha: "123456"))
            .thenReturn(usuarioMock);

    mockMvc.perform(post( uriTemplate: "/logar")
                    .param( name: "email",  ...values: "teste@email.com")
                    .param( name: "senha",  ...values: "123456"))
            .andExpect(status().is3xxRedirection())
            .andExpect(redirectedUrl( expectedUrl: "/todolist"));
}
```

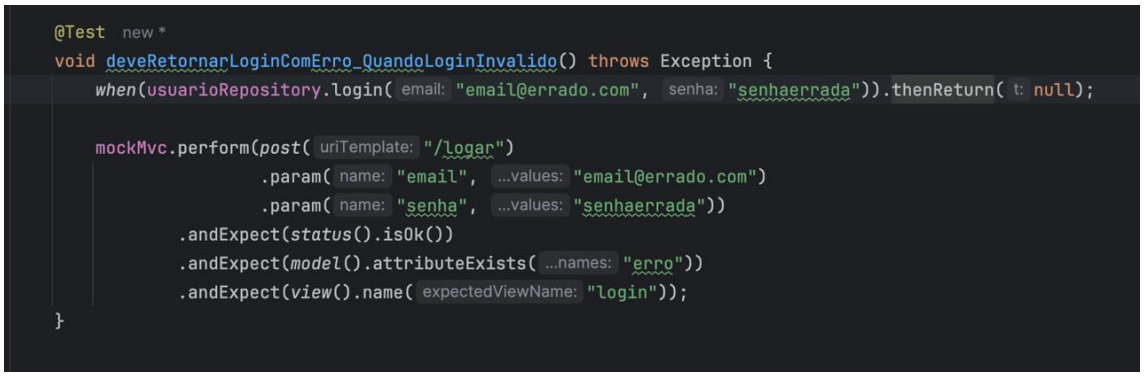
```java
@Test  new *
void deveRetornarLoginComErro_QuandoLoginForInvalido() throws Exception {
    when(usuarioRepository.login( email: "errado@email.com",  senha: "senhaerrada"))
            .thenReturn( t: null);

    mockMvc.perform(post( uriTemplate: "/logar")
                    .param( name: "email",  ...values: "errado@email.com")
                    .param( name: "senha",  ...values: "senhaerrada"))
            .andExpect(status().isOk())
            .andExpect(model().attributeExists( ...names: "erro"))
            .andExpect(view().name( expectedViewName: "login"));
    }
}
```



CT-014:

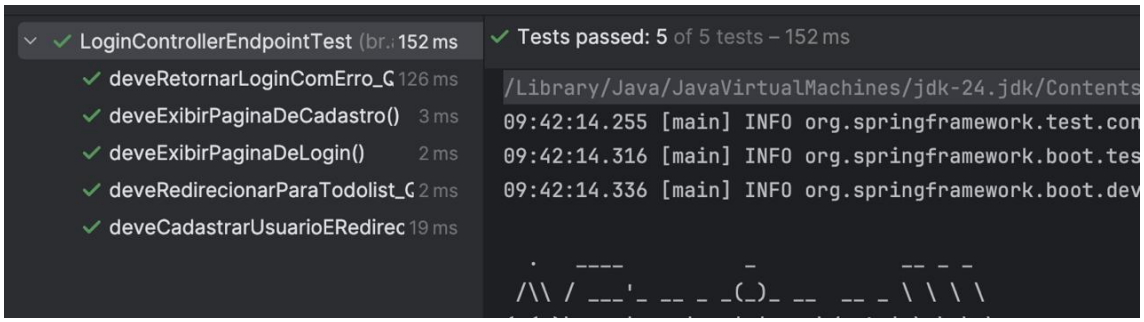
```java
@Test  new *
void deveExibirPaginaDeLogin() throws Exception {
    mockMvc.perform(get( uriTemplate: "/login"))
            .andExpect(status().isOk())
            .andExpect(view().name( expectedViewName: "login"));
    }
```

```java
@Test  new *
void deveRedirecionarParaTodolist_QuandoLoginValido() throws Exception {
    Usuario usuarioMock = new Usuario();
    usuarioMock.setEmail("teste@email.com");
    usuarioMock.setSenha("123456");

    when(usuarioRepository.login( email: "teste@email.com",  senha: "123456")).thenReturn(usuarioMock);

    mockMvc.perform(post( uriTemplate: "/logar")
                    .param( name: "email",  ...values: "teste@email.com")
                    .param( name: "senha",  ...values: "123456"))
            .andExpect(status().is3xxRedirection())
            .andExpect(redirectedUrl( expectedUrl: "/todolist"));
    }
```



```
@Test  new *
void deveRetornarLoginComErro_QuandoLoginInvalido() throws Exception {
    when(usuarioRepository.login( email: "email@errado.com",  senha: "senhaerrada")).thenReturn( t: null);

    mockMvc.perform(post( uriTemplate: "/logar")
                    .param( name: "email",  ...values: "email@errado.com")
                    .param( name: "senha",  ...values: "senhaerrada"))
            .andExpect(status().isOk())
            .andExpect(model().attributeExists( ...names: "erro"))
            .andExpect(view().name( expectedViewName: "login"));
}
```

```
@Test  new *
void deveCadastrarUsuarioERedirecionarParaLogin() throws Exception {
    mockMvc.perform(post( uriTemplate: "/cadastro")
                    .param( name: "nome",  ...values: "Usuário Teste") // adicione todos os campos obrigatórios
                    .param( name: "email",  ...values: "novo@usuario.com")
                    .param( name: "senha",  ...values: "123456"))
            .andExpect(status().is3xxRedirection())
            .andExpect(redirectedUrl( expectedUrl: "/login")); // aqui espera o redirect certo


    // Verifica se salvou o usuário
    verify(usuarioRepository, times( wantedNumberOfInvocations: 1)).save(any(Usuario.class));
}
}
```



```
LoginControllerEndpointTest (br.  152 ms
  deveRetornarLoginComErro_C 126 ms
  deveExibirPaginaDeCadastro()  3 ms
  deveExibirPaginaDeLogin()  2 ms
  deveRedirecionarParaTodolist_C 2 ms
  deveCadastrarUsuarioERedirec 19 ms

Tests passed: 5 of 5 tests – 152 ms

/Library/Java/JavaVirtualMachines/jdk-24.jdk/Contents
09:42:14.255 [main] INFO org.springframework.test.con
09:42:14.316 [main] INFO org.springframework.boot.tes
09:42:14.336 [main] INFO org.springframework.boot.dev

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
```

6. Conclusão e Lições Aprendidas

O desenvolvimento do EducaSis proporcionou à equipe a oportunidade de aplicar conceitos fundamentais de engenharia de software, desenvolvimento web com SpringBoot e integração com banco de dados MySQL, além da elaboração e execução de testes funcionais, de API e E2E.

O projeto cumpriu seu objetivo de simular um sistema escolar, disponibilizando funcionalidades de cadastro e login de usuários, gestão de tarefas por perfil e diferenciação entre professores e alunos. A implementação de casos de teste detalhados garantiu que os requisitos fossem validados de forma eficiente, resultando em um produto estável e funcional para fins acadêmicos.

Além disso, a experiência com ferramentas como o IntelliJ IDEA para testes de API e a estruturação das histórias de usuário em BDD contribuiu para a melhoria dos processos de validação e controle de qualidade do software.

7. **Link para o repositório**

   https://github.com/marcosalbuquerque/appLoginSpring/tree/main