FECAP

PROJETO 6 - Educaliza

Requisitos da disciplina Modelagem de Software e Arquitetura de Sistemas

São Paulo

2024

INTEGRANTES DO PROJETO e RA’S

Cleiton Lima	-	22024231
Gabriel Ítalo	-	22024115
Matheus Adaniya	-	23025170
Wilson Testoni	-	22024290

Contents

1. INTRODUÇÃO

O Educaliza é um aplicativo que facilita o acesso a cursos e aulas gratuitos na cidade de São Paulo, fornecendo uma plataforma onde parceiros podem publicar informações detalhadas sobre seus cursos, como datas de início e término, localização e mais. Com ele, você encontra o conteúdo que deseja estudar sem nenhum custo! Esses cursos são oferecidos por instituições parceiras, como ONGs, garantindo a diversidade e a qualidade das opções disponíveis.

2. Teste de Software

2.1. Apresentar 2 testes unitários.

Nas figuras abaixo, os testes consistem em validar o nome do usuário no aplicativo, verificando se os dados enviados resultam em sucesso ou falha de acordo com as regras de validação implementadas:

Figura 1: Este código valida o nome do usuário usando expressões regulares (regex) no android studio, permitindo letras maiúsculas e minúsculas, espaços e caracteres especiais de acentuação.

Figura 2: Na figura 2 é feito dois testes unitários usando o código da primeira figura:

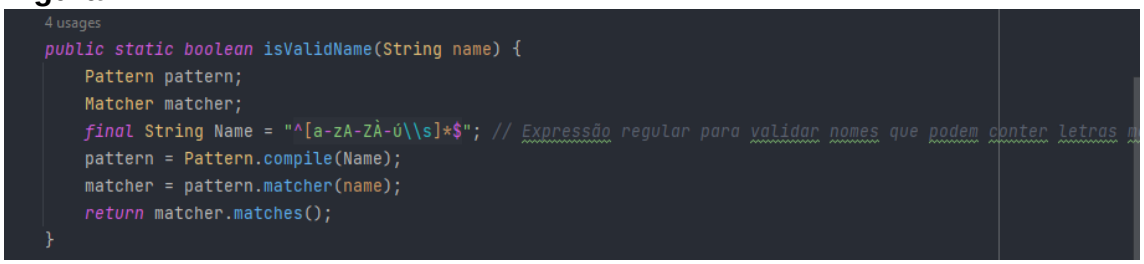
Test 1 isValidName_ValidName_ReturnsTrue:

Este teste verifica se o método isValidName retorna true para um nome válido (John Doe). Ele usa o método assertTrue para garantir que o resultado retornado seja verdadeiro.

Test 2 isValidName_InvalidName_ReturnsFalse:

Este teste verifica se o método isValidName retorna false para nomes inválidos (com caracteres numéricos ou caracteres especiais). Ele usa o método assertFalse para garantir que o resultado retornado seja falso para cada caso de entrada inválida.

Figura-1



```
4 usages
public static boolean isValidName(String name) {
    Pattern pattern;
    Matcher matcher;
    final String Name = "^[a-zA-ZÀ-ú\\s]*$"; // Expressão regular para validar nomes que podem conter letras n
    pattern = Pattern.compile(Name);
    matcher = pattern.matcher(name);
    return matcher.matches();
}
```

Figura-2

```
@Test
public void isValidName_ValidName_ReturnsTrue() {
    assertTrue(ValidacaoFormCadastro.isValidName("John Doe"));
}

@Test
public void isValidName_InvalidName_ReturnsFalse() {
    assertFalse(ValidacaoFormCadastro.isValidName("John123")); // caracteres numéricos
    assertFalse(ValidacaoFormCadastro.isValidName("John@Doe")); // caracteres especiais
}
```

Nas figuras abaixo, os testes consistem em validar o cnpj do parceiro no backend, verificando se o cnpj resultam em sucesso ou falha de acordo com as regras de validação implementadas:

Figura 1: Função para verificar se um CNPJ (Cadastro Nacional da Pessoa Jurídica) é válido, removendo caracteres não numéricos, verificando o tamanho correto e validando os dígitos verificadores através de cálculos específicos.

Figura 2: Testes unitários para a função validateCNPJ, que verificam diferentes cenários de validade de um CNPJ:

Teste 1: valid CNPJ returns true
Verifica se um CNPJ válido ('57256634000127') retorna true.

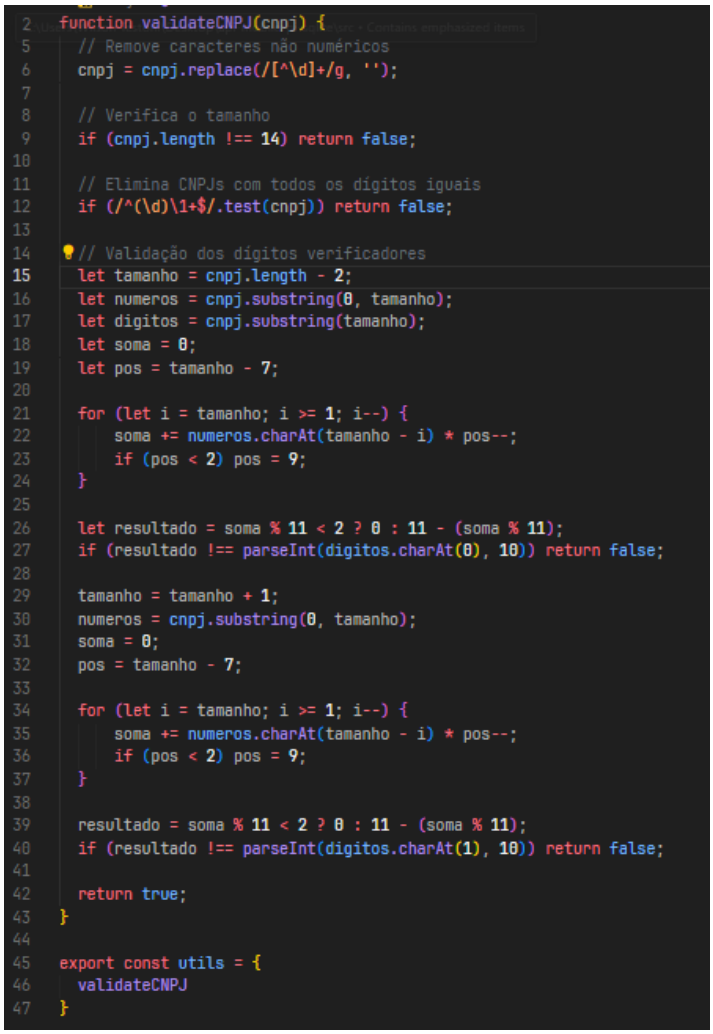
Teste 2: invalid CNPJ returns false
Verifica se um CNPJ inválido ('12345678901234') retorna false.

Teste 3: invalid CNPJ with equals numbers returns false
Verifica se um CNPJ com todos os dígitos iguais ('55555555555555') retorna false.

Teste 4: invalid CNPJ with less numbers returns false
Verifica se um CNPJ com menos dígitos do que o necessário ('12123123123') retorna false.

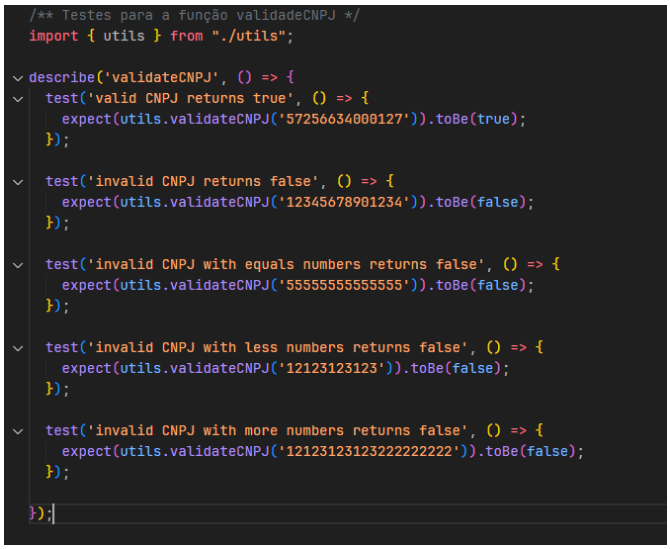
Teste 5: invalid CNPJ with more numbers returns false
Verifica se um CNPJ com mais dígitos do que o necessário ('121231231232222222222') retorna false.

Figura-1



```
function validateCNPJ(cnpj) {
    // Remove caracteres não numéricos
    cnpj = cnpj.replace(/[^\d]+/g, '');

    // Verifica o tamanho
    if (cnpj.length !== 14) return false;

    // Elimina CNPJs com todos os dígitos iguais
    if (/^(\d)\1+$/.test(cnpj)) return false;

    // Validação dos dígitos verificadores
    let tamanho = cnpj.length - 2;
    let numeros = cnpj.substring(0, tamanho);
    let digitos = cnpj.substring(tamanho);
    let soma = 0;
    let pos = tamanho - 7;

    for (let i = tamanho; i >= 1; i--) {
        soma += numeros.charAt(tamanho - i) * pos--;
        if (pos < 2) pos = 9;
    }

    let resultado = soma % 11 < 2 ? 0 : 11 - (soma % 11);
    if (resultado !== parseInt(digitos.charAt(0), 10)) return false;

    tamanho = tamanho + 1;
    numeros = cnpj.substring(0, tamanho);
    soma = 0;
    pos = tamanho - 7;

    for (let i = tamanho; i >= 1; i--) {
        soma += numeros.charAt(tamanho - i) * pos--;
        if (pos < 2) pos = 9;
    }

    resultado = soma % 11 < 2 ? 0 : 11 - (soma % 11);
    if (resultado !== parseInt(digitos.charAt(1), 10)) return false;

    return true;
}

export const utils = {
    validateCNPJ
}
```

Figura-2



```
/** Testes para a função validadeCNPJ */
import { utils } from "./utils";

describe('validateCNPJ', () => {
    test('valid CNPJ returns true', () => {
        expect(utils.validateCNPJ('57256634000127')).toBe(true);
    });

    test('invalid CNPJ returns false', () => {
        expect(utils.validateCNPJ('12345678901234')).toBe(false);
    });

    test('invalid CNPJ with equals numbers returns false', () => {
        expect(utils.validateCNPJ('55555555555555')).toBe(false);
    });

    test('invalid CNPJ with less numbers returns false', () => {
        expect(utils.validateCNPJ('12123123123')).toBe(false);
    });

    test('invalid CNPJ with more numbers returns false', () => {
        expect(utils.validateCNPJ('12123123123222222222')).toBe(false);
    });

});
```

2.2. Apresentar 2 testes de componentes

Cadastro usuário normal:

Nas figuras abaixo os testes consistem em um **usuário** realizando o cadastro no Aplicativo enviando os dados necessário e tendo o retorno de sucesso ou falha do cadastro:

Figura 1: O usuário está passando os dados necessários como **Nome**, **E-mail**, **Senha** e **Telefone**, para fazer a criação da conta e tendo o sucesso dela, retornando para ele **“Usuário cadastrado com sucesso”**

Figura 2: O usuário está passando os dados necessários como **Nome**, **E-mail**, **Senha** e **Telefone,** para fazer a criação da conta, porém ao fazer a validação do E-mail, ele já consta na nossa base de dados, retornando para ele que **“E-mail já está cadastrado”**

Figura-1

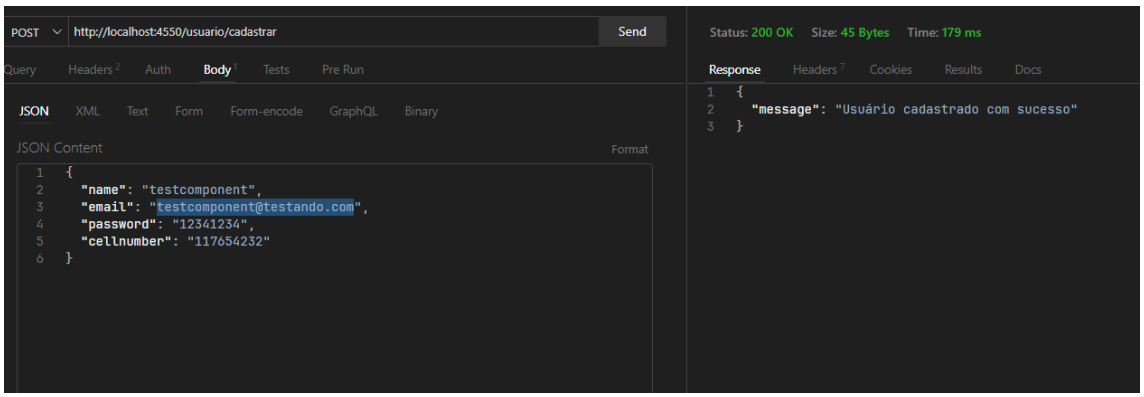
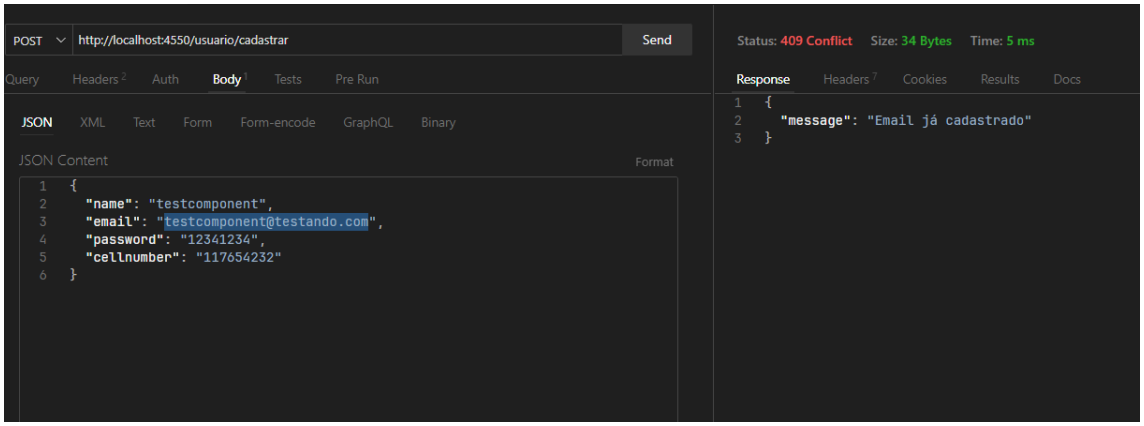


Figura-2



Cadastro usuário tipo parceiro:

Nas figuras abaixo os testes consistem em um parceiro realizando o cadastro no Aplicativo enviando os dados necessário e tendo o retorno de sucesso ou falha do cadastro:

Figura 1: O parceiro está passando os dados necessários como **Nome da instituição**, **CNPJ, E-mail**, **Senha** e **Telefone**, para fazer a criação da conta e tendo o sucesso dela, retornando para ele **“Parceiro cadastrado com sucesso”**

Figura 2: O parceiro está passando os dados necessários como **Nome da instituição**, **CNPJ, E-mail**, **Senha** e **Telefone,** para fazer a criação da conta, porém ao fazer a validação do E-mail, ele já consta na nossa base de dados, retornando para ele que **“E-mail já está cadastrado”.**

Figura 3: O parceiro está passando os dados necessários como **Nome**, **E-mail**, **Senha** e **Telefone,** para fazer a criação da conta, porém ao fazer a validação do CNPJ, ele já consta na nossa base de dados, retornando para ele que **“CNPJ inválido”.**

Figura-1

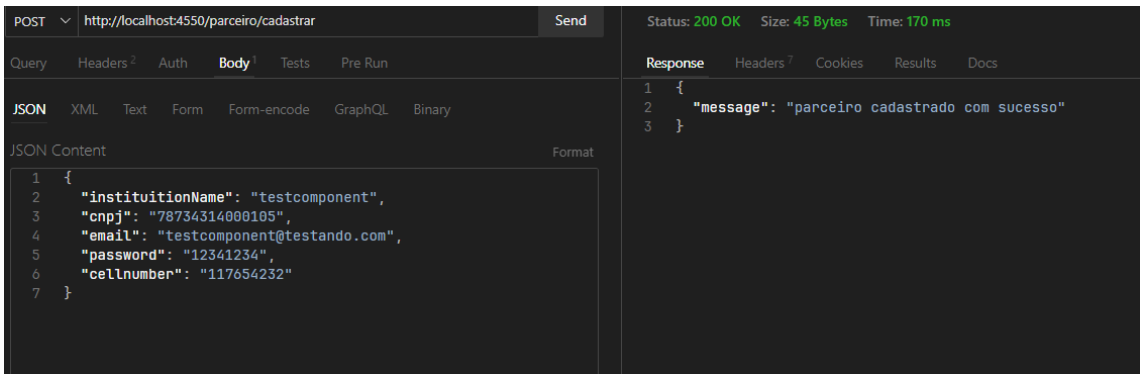


Figura-2

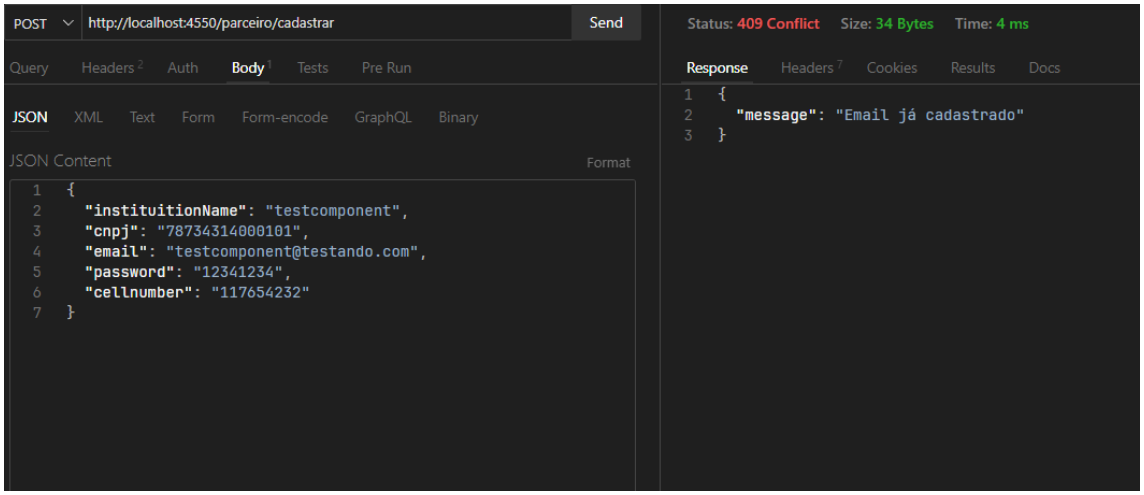
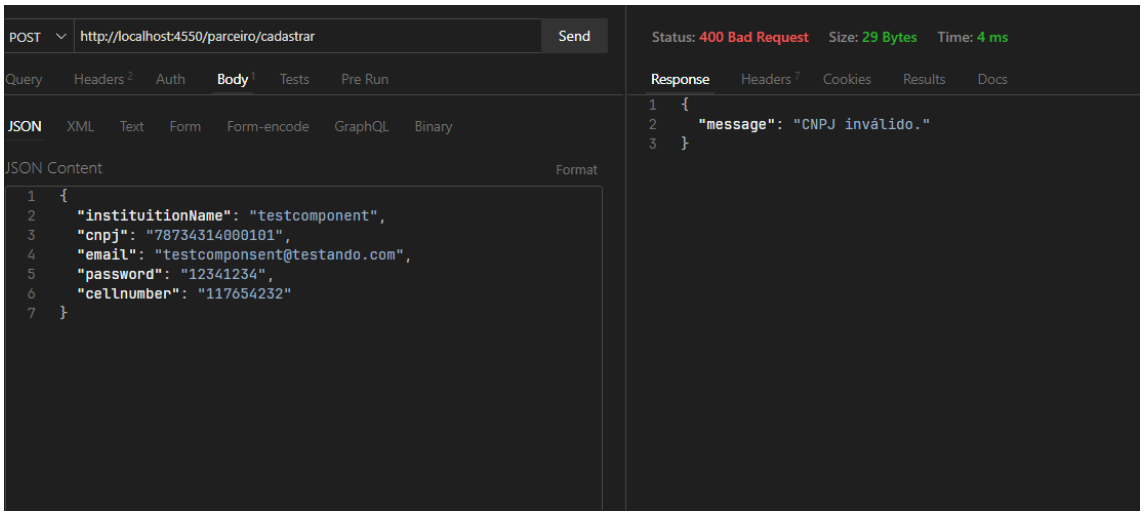


Figura-3



2.3. Apresentar um teste de sistema.

O teste de sistema foi feito em formato de vídeo e acompanha a documentação no Github com o nome:
Testequaldevops_Teste_sistema

3. Qualidade de Software

3.1. Indicar 4 atributos de qualidade de software e informar como foi aplicado no projeto integrador (PI)

Funcionabilidade: A interface intuitiva e amigável do aplicativo torna a navegação e utilização das funcionalidades acessíveis a usuários de todos os níveis de habilidade. Os processos de cadastro, busca e inscrição em cursos são simplificados para garantir uma experiência sem complicações.

Eficiência no desempenho: O aplicativo oferece uma ampla gama de funcionalidades que permitem aos usuários acessarem e se inscrever em cursos de forma rápida e eficiente. Além disso, a divisão em dois tipos de acesso, usuário e parceiro, simplifica a interação e a utilização do aplicativo para diferentes propósitos, seja para aprender ou para disponibilizar cursos.

Confiabilidade: O aplicativo demonstra consistência e estabilidade em seu funcionamento, garantindo uma experiência livre de falhas e travamentos durante o processo de cadastro, navegação e inscrição em cursos. Os dados dos usuários, incluindo informações de cadastro e inscrições em cursos, são protegidos de forma segura pelo aplicativo, garantindo a privacidade e confidencialidade das informações pessoais.

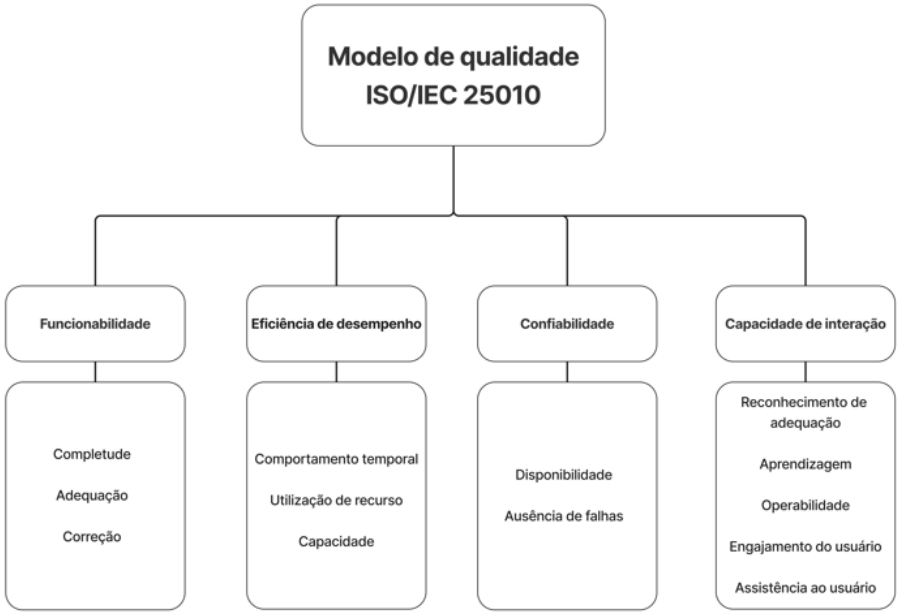
Capacidade de Interação:

Para Usuários Comuns: No aplicativo, os usuários comuns podem encontrar e se inscrever nos cursos desejados de forma fácil e rápida. Se precisarem, também têm a opção de cancelar a inscrição sem complicações. Além disso, oferecemos um perfil personalizado para cada usuário, onde é possível visualizar todos os cursos em que estão inscritos, com a possibilidade de utilizar diversos filtros para uma melhor organização.

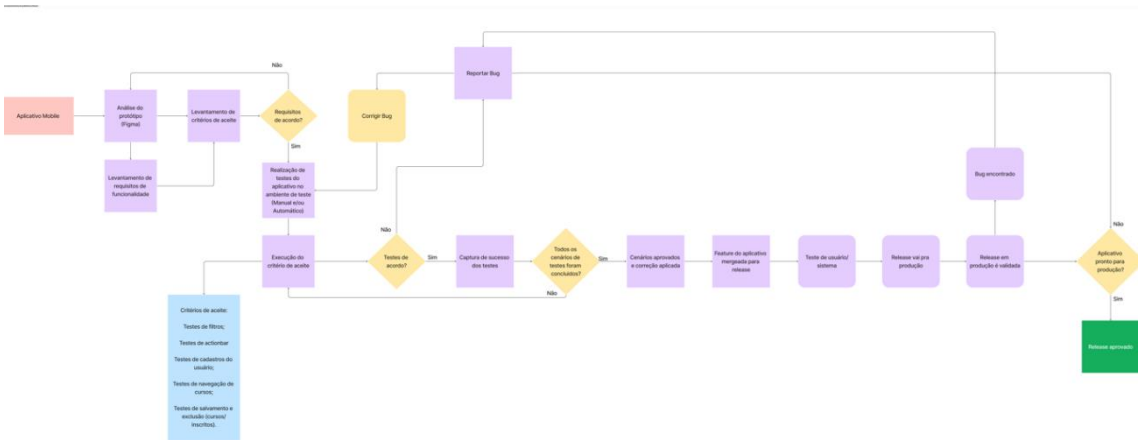
Para Instituições Parceiras: As instituições parceiras têm acesso a ferramentas que facilitam a gestão dos cursos. Elas podem criar, atualizar e excluir cursos diretamente na plataforma, sem a necessidade de intervenção externa. Além

disso, fornecemos um perfil exclusivo para cada instituição, onde é possível visualizar a lista completa de cursos cadastrados, juntamente com informações sobre a quantidade de inscritos em cada curso, permitindo uma gestão mais eficiente e informada.

3.2. Apresentar um Modelo que qualidade de software



3.3. Apresentar um Processo (plano) de gerenciamento de qualidade de software



4. REFERÊNCIAS BIBLIOGRÁFICAS

ISO 25000. "ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models." Retrieved from https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

SOMMERVILLE, I. **Engenharia de Software.** 11ª Edição. São Paulo: Pearson Addison-Wesley, 2017