FECAP

SKL – Aplicativo de gestão e divulgação da Fecap Social

Requisitos da disciplina Modelagem de Software e Arquitetura de Sistemas

São Paulo

2024

INTEGRANTES DO PROJETO e RA’S

Grupo 2

Ana Flavia Lorêdo	-	23025092
Fernanda Mayumi Kuba Kato	-	23024484
Kevin Makoto Shiroma	-	20020925
Renato Riichi Kato	-	23024516

Contents

1. INTRODUÇÃO

Este documento de requisitos apresenta uma visão abrangente dos objetivos, funcionalidades e especificações técnicas do aplicativo Fecap Social, que já foi desenvolvido e está em uso na comunidade acadêmica da FECAP Social. O Fecap Social é uma entidade comprometida com a promoção do conhecimento, interação e colaboração entre seus membros, e o aplicativo foi concebido para fortalecer esses laços, oferecendo um ambiente acadêmico mais integrado e dinâmico.

No desenvolvimento do aplicativo Fecap Social, foram definidos e implementados requisitos funcionais e não funcionais para garantir sua eficácia e usabilidade. Além disso, foram realizados testes de software, incluindo testes unitários, testes de componentes e teste de sistema, para validar e verificar o comportamento do aplicativo em diferentes cenários de uso.

Os testes unitários e de componentes foram realizados utilizando o framework JUnit, enquanto os testes de componente foram automatizados por meio do framework Espresso. Essas ferramentas foram escolhidas pela sua eficiência e confiabilidade na validação do comportamento do aplicativo.

O aplicativo Fecap Social atende a diversos atributos de qualidade de software, incluindo usabilidade, completude funcional, compatibilidade e capacidade de interação. Esses atributos foram cuidadosamente considerados e aplicados durante o desenvolvimento do projeto integrador (PI), garantindo que o aplicativo atendesse às necessidades e expectativas da comunidade FECAP.

Este documento também apresenta um modelo de qualidade de software adotado para orientar o desenvolvimento do aplicativo Fecap Social, bem como um processo de gerenciamento de qualidade de software que foi

implementado para garantir a conformidade com os padrões e requisitos estabelecidos.

O aplicativo Fecap Social tem se mostrado uma ferramenta indispensável para a comunidade FECAP, contribuindo significativamente para a promoção do conhecimento, interação e engajamento entre seus membros. Este documento serve como um guia abrangente que documenta os requisitos, testes e processos de qualidade do aplicativo Fecap Social, oferecendo uma visão clara de todas as etapas do projeto.

2. Teste de Software

2.1. Apresentar 2 testes unitários.

2.1.1 Teste: Os get, set e construtor das classes

Nas figuras abaixo os testes unitários consistem na verificação das funcionalidades da classeUsuario.

2.1.1.1 Teste unitário 1: Este teste verifica a funcionalidade do construtor completo da classe `ClasseUsuario`. Primeiro, criamos um objeto `ClasseUsuario` com todos os atributos preenchidos: nome, email e senha.

Em seguida, verificamos se o objeto criado não é nulo usando o método `assertNotNull()`. Isso garante que o construtor esteja funcionando corretamente e que um objeto válido tenha sido criado.

Depois disso, usamos o método `assertEquals()` para verificar se os atributos do objeto `ClasseUsuario` foram definidos corretamente. Comparamos o nome, o email e a senha do objeto com os valores esperados ("Nome Teste", "email@teste.com" e "senha123", respectivamente).

Essa abordagem de teste garante que o construtor da classe `ClasseUsuario` esteja configurando corretamente os atributos do objeto com os valores fornecidos durante a criação. Isso ajuda a garantir a integridade dos dados e o correto funcionamento do código que depende desses objetos `ClasseUsuario`.

Figura 01: Teste unitário 1

```
@Test
public void testConstrutorCompleto() {
    // Criar um objeto ClasseUsuario com todos os atributos
    ClasseUsuario usuario = new ClasseUsuario( nome: "Nome Teste",  email: "email@teste.com",  senha: "senha123");

    // Verificar se o objeto não é nulo
    assertNotNull(usuario);

    // Verificar se os atributos foram definidos corretamente
    assertEquals( expected: "Nome Teste", usuario.getNome());
    assertEquals( expected: "email@teste.com", usuario.getEmail());
    assertEquals( expected: "senha123", usuario.getSenha());
}
```

2.1.1.2 Teste unitário 2: Este teste verifica a funcionalidade dos métodos setters e getters da classe `ClasseUsuario`.

Primeiro, criamos um objeto `ClasseUsuario` vazio, passando apenas o email como parâmetro no construtor.

Em seguida, usamos os métodos setters para definir os valores dos atributos `nome` e `senha`. Definimos o nome como "Novo Nome" e a senha como "novaSenha123".

Por fim, usamos os métodos getters para verificar se os valores dos atributos foram definidos corretamente. Utilizamos o método `assertEquals()` para comparar os valores retornados pelos getters com os valores esperados.

Essa abordagem de teste garante que os métodos setters e getters da classe `ClasseUsuario` estejam funcionando corretamente, permitindo a manipulação e obtenção dos valores dos atributos do objeto `ClasseUsuario`.

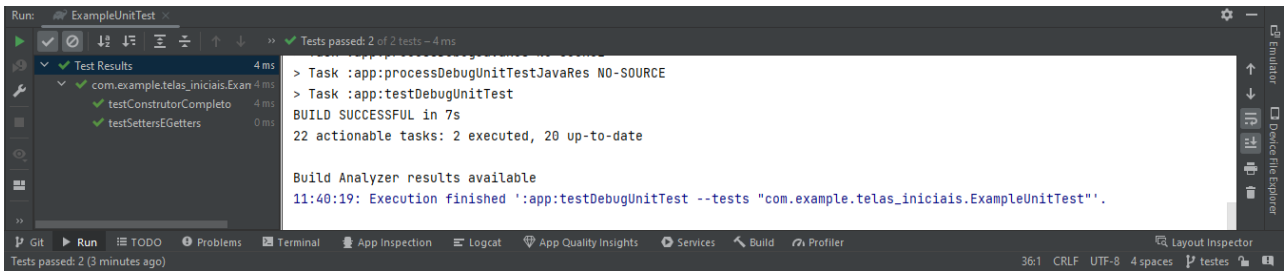
Figura 02: Teste unitário 2

```
@Test
public void testSettersEGetters() {
    // Criar um objeto ClasseUsuario vazio
    ClasseUsuario usuario = new ClasseUsuario( email: "email@teste.com");

    // Definir os valores dos atributos usando os setters
    usuario.setNome("Novo Nome");
    usuario.setSenha("novaSenha123");

    // Verificar se os getters retornam os valores corretos
    assertEquals( expected: "Novo Nome", usuario.getNome());
    assertEquals( expected: "email@teste.com", usuario.getEmail());
    assertEquals( expected: "novaSenha123", usuario.getSenha());

}
```

2.1.2 Resultado dos testes unitários

A figura abaixo, mostra o sucesso dos testes realizados:

Figura 03: Resultado dos testes unitários



2.2. Apresentar 2 testes de componentes

Nos testes de componentes foi feito os códigos usando o junit e também em formato de vídeo, que acompanha a documentação no Github com o nome:

Testequaldevops_Teste_componente

2.2.1 Teste: Login do usuário

Nas figuras abaixo os testes consistem em o usuário fazer o login com o email e a senha, e é retornado se o acesso teve sucesso ou erro.

2.2.1.1 Teste componente 1: Este teste verifica se o login é bem-sucedido no aplicativo. Primeiro, simulamos a inserção de um email digitando "skl.projetopi2@gmail.com" no campo de email e, em seguida, digitamos a senha "valid6" no campo de senha. Depois disso, clicamos no botão de login.

O objetivo é garantir que o usuário consiga fazer login com sucesso fornecendo um email e uma senha válidos. Este teste nos ajuda a verificar se o fluxo de login está funcionando corretamente e se os campos de email e senha estão aceitando os dados corretamente.

Figura 01: teste componente 1

```
@Rule
public ActivityScenarioRule<LoginActivity> activityScenarioRule = new ActivityScenarioRule<>(LoginActivity.class);

@Test
public void testLoginSucesso() {
    // Simula um input
    Espresso.onView(ViewMatchers.withId(R.id.inputEmailLogin)).perform(ViewActions.typeText( stringToBeTyped: "skl.projetopi2@gmail.com"));
    Espresso.onView(ViewMatchers.withId(R.id.inputSenhaLogin)).perform(ViewActions.typeText( stringToBeTyped: "valid6"), ViewActions.closeSoftKeyboard());

    // Clica no botão de login
    Espresso.onView(ViewMatchers.withId(R.id.btnEntrar)).perform(ViewActions.click());

}
```

2.2.1.2 Teste componente 2: Este teste verifica o comportamento do aplicativo quando o login é inválido. Primeiro, simulamos a inserção de um email válido ("skl.projetopi2@gmail.com") no campo de email e uma senha inválida ("invalid") no campo de senha. Em seguida, fechamos o teclado virtual após a inserção da senha.

Depois disso, clicamos no botão de login para tentar efetuar o login com as credenciais fornecidas.

O objetivo deste teste é garantir que, quando as credenciais de login são inválidas, um alerta seja exibido ao usuário com o título "Erro de Login" e a mensagem "Email ou senha incorretos!!!". Além disso, o alerta deve conter um botão com o texto "Tentar novamente" para que o usuário possa tentar fazer login novamente.

Figura 02: teste componente 2

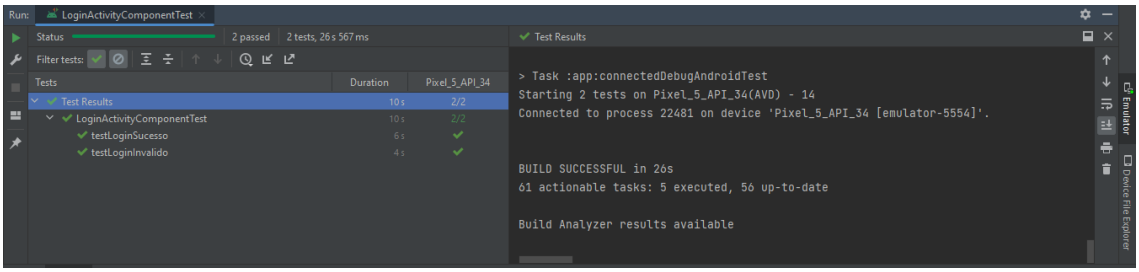
```
@Test
public void testLoginInvalido() {
    // Simula um input
    Espresso.onView(ViewMatchers.withId(R.id.inputEmailLogin)).perform(ViewActions.typeText( stringToBeTyped: "skl.projetopi2@gmail.com"));
    Espresso.onView(ViewMatchers.withId(R.id.inputSenhaLogin)).perform(ViewActions.typeText( stringToBeTyped: "invalid"), ViewActions.closeSoftKeyboard());

    // Clica no botão de login
    Espresso.onView(ViewMatchers.withId(R.id.btnEntrar)).perform(ViewActions.click());

    // Verifica se o alerta é exibido com o titulo correto
    Espresso.onView(ViewMatchers.withText("Erro de Login")).inRoot(RootMatchers.isDialog()).check(ViewAssertions.matches(ViewMatchers.isDisplayed()));

    // Verifica se o alerta contém a mensagem correta
    Espresso.onView(ViewMatchers.withText("Email ou senha incorretos!!!")).inRoot(RootMatchers.isDialog()).check(ViewAssertions.matches(ViewMatchers.isDi

    // Verifica se o botão no alerta contém o texto correto
    Espresso.onView(ViewMatchers.withText("Tentar novamente")).inRoot(RootMatchers.isDialog()).check(ViewAssertions.matches(ViewMatchers.isDisplayed()));
}
```

Essas verificações são feitas usando as funções do Espresso para verificar se o alerta é exibido corretamente com o título, mensagem e botão esperados. Isso ajuda a garantir que o aplicativo esteja fornecendo feedback apropriado ao usuário quando as credenciais de login são inválidas.

2.2.2 Resultado dos testes de componente

A figura abaixo, mostra o sucesso dos testes realizados:

Figura 3: Resultado dos testes de componente



2.3. Apresentar um teste de sistema.

O teste de sistema foi realizado 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)

3.1.1 Atributo 01: Usabilidade

No intuito de proporcionar uma experiência de usuário excepcional em nosso aplicativo, adotamos uma abordagem centrada no usuário, fundamentada no processo de Design Thinking. Este processo é composto por várias fases, cada uma delas dedicada a compreender, definir, idealizar, prototipar e testar soluções que atendam às necessidades e expectativas dos nossos usuários.

1. **Empatia:** Compreendendo as Necessidades dos Usuários

A fase inicial do nosso processo de Design Thinking concentra-se em compreender profundamente os usuários do nosso aplicativo. Utilizamos técnicas como entrevistas, observação e análise de dados para entender os desejos, necessidades, frustrações e comportamentos dos usuários. Essa empatia nos permite obter insights valiosos que orientam todo o desenvolvimento do aplicativo.

2. **Definição:** Identificação de Problemas e Oportunidades

Com base nos insights obtidos na fase de empatia, concentramo-nos em definir claramente os problemas e oportunidades que os usuários enfrentam ao interagir com o nosso aplicativo. Definimos metas específicas que queremos alcançar e identificamos as áreas que precisam ser melhoradas para garantir uma experiência de usuário de alta qualidade.

3. **Idealização:** Geração de Soluções Criativas

Na fase de idealização, reunimos uma equipe multidisciplinar para gerar uma ampla gama de soluções criativas para os problemas identificados. Utilizamos técnicas como brainstorming, mapas mentais e prototipagem rápida para explorar diferentes abordagens e conceitos inovadores que possam resolver os desafios de usabilidade do aplicativo.

4. **Prototipação:** Visualização de Ideias em Ação

Com base nos conceitos gerados na fase de idealização, desenvolvemos protótipos de baixa e alta fidelidade que representam as diferentes soluções propostas. Esses protótipos nos permitem visualizar as ideias em ação e obter feedback dos usuários antes de investir recursos significativos no desenvolvimento do aplicativo final.

5. **Testes:** Validando e Refinando as Soluções

Na fase de testes, conduzimos uma série de testes com usuários reais para validar as soluções propostas e identificar possíveis pontos de melhoria. Observamos atentamente como os usuários interagem com o aplicativo, coletamos feedback qualitativo e quantitativo e refinamos continuamente as soluções com base nos resultados dos testes.

Ao seguir esse processo iterativo e centrado no usuário, garantimos que nosso aplicativo seja desenvolvido com base nas necessidades e expectativas reais dos usuários, resultando em uma experiência de usuário otimizada, intuitiva e altamente satisfatória.

3.1.2 Atributo 02: Completude Funcional

FECAP Social é um aplicativo desenvolvido para facilitar a interação e colaboração entre membros da comunidade FECAP. Para garantir que todas as necessidades do usuário sejam plenamente atendidas, seguimos um cuidadoso processo de desenvolvimento, com foco na “completude funcional” da aplicação.

1. **Identificação de requisitos:** Antes de iniciar o desenvolvimento, analisamos minuciosamente as necessidades do usuário, identificando todas as funções e funcionalidades necessárias para uma experiência completa.

2. **Planejamento detalhado:** Com base nos requisitos identificados, criamos um plano detalhado para implementar cada função e funcionalidade, garantindo que todos os aspectos importantes foram cobertos.

3. **Necessidades de implementação:** Nossa equipe de desenvolvimento tem trabalhado para implementar todos os aspectos do FECAP Social com precisão, garantindo que tudo funcione conforme planejado.

4. **Ensaios rigorosos:** Antes do lançamento, realizamos testes extensivos para garantir que todos os recursos funcionassem bem em vários cenários.

5. **Comentários do usuário:** Envolvemos ativamente os usuários durante todo o processo, coletando feedback e fazendo ajustes para garantir uma experiência otimizada.

6. **Documentação clara:** Documentamos todos os recursos e funcionalidades de forma clara e acessível para que os usuários possam aproveitar ao máximo o FECAP Social. Com estas práticas, garantimos que o FECAP Social seja uma ferramenta completa e eficaz para facilitar a comunicação e colaboração.

3.1.3 Atributo 03: Compatibilidade

O FECAP Social foi projetado com o compromisso de garantir uma experiência consistente e acessível para todos os membros da comunidade FECAP, independentemente do dispositivo que estão utilizando. Para alcançar isso, dedicamos especial atenção à compatibilidade do aplicativo com uma ampla gama de dispositivos.

Além da compatibilidade com dispositivos, também nos preocupamos com o desempenho do FECAP Social em diferentes condições de rede e hardware. Otimizamos o aplicativo para garantir que ele seja leve e responsivo, mesmo em dispositivos mais antigos ou com conexões de internet mais lentas, garantindo uma experiência de usuário fluida e sem interrupções.

3.1.4 Atributo 04: Capacidade de interação

A capacidade de interação é um aspecto essencial em qualquer aplicativo, especialmente em plataformas que visam facilitar a comunicação e colaboração entre os usuários. No caso do FECAP Social, essa capacidade de interação é fundamental para promover o engajamento da comunidade FECAP e facilitar a troca de informações, ideias e experiências entre os membros.

Recursos de Interatividade: Eventos e Calendários: participar de eventos, além de acompanhar calendários de atividades da entidade FECAP Social.

Facilidade de Uso: Além de oferecer uma ampla gama de recursos interativos, o FECAP Social foi projetado para ser fácil de usar, mesmo para usuários inexperientes. Uma interface intuitiva e amigável torna a interação com o aplicativo simples e direto, incentivando os usuários a participarem ativamente e contribuírem para a comunidade FECAP.

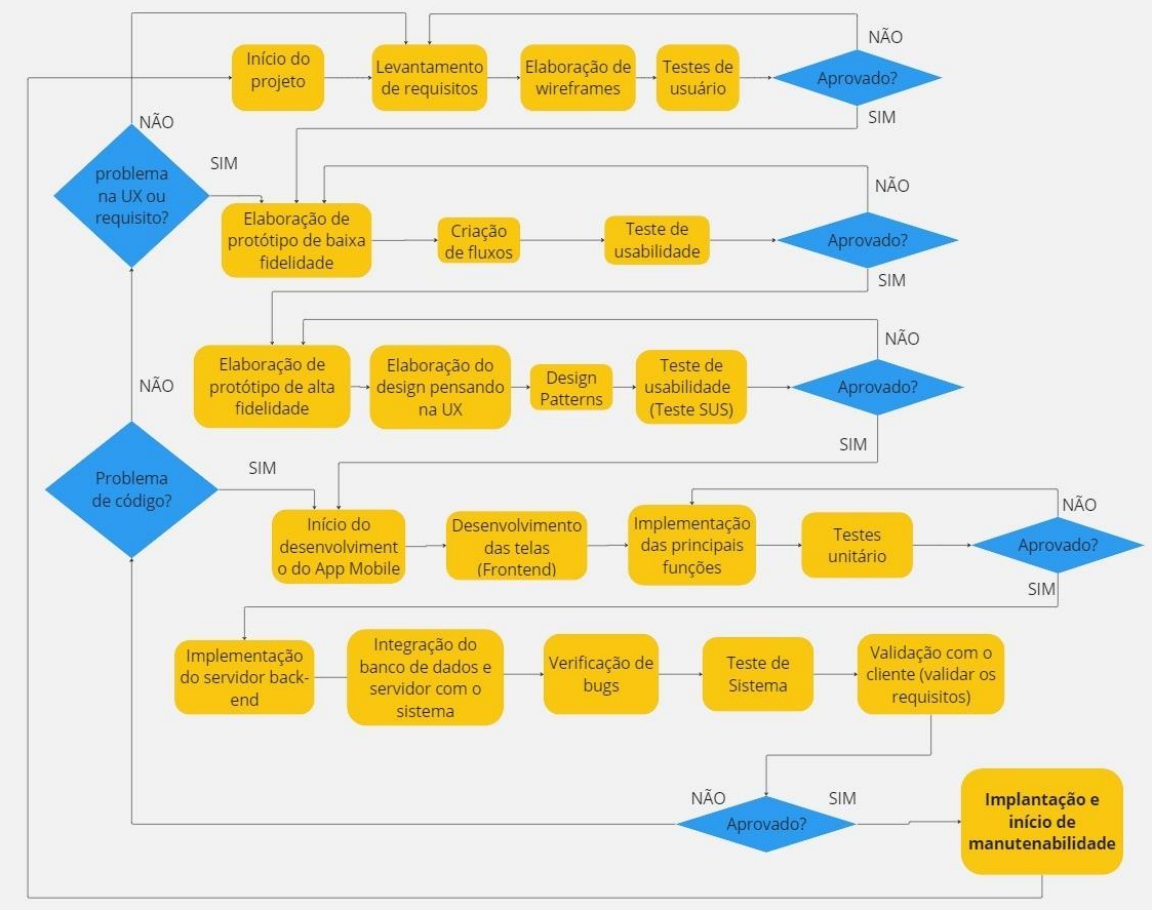
3.2. Apresentar um Modelo que qualidade de software

Este modelo segue o ISO/IEC 25010 de qualidade de software e as seguintes características foram destacadas:

Qualidade de Software		
Adequação Funcional	**Capacidade de Interação**	**Confiabilidade**
Completude Funcional: O APP não possui funções complexas, possui caráter informativo acerca dos aventos feito pela FECAP Social. O APP tem como função abordar essa pretensão.	**Reconhecibilidade de adequação:** Mesmo ao instalar ao APP o usuário rapidamente já sabe se ele irá satisfazer suas necessidades ou não, pois o único objetivo do app é: informar ao usuario sobre a FECAP Social e seus eventos futuros.	**Disponibilidade:** O APP precisa estar disponível sempre que um usuário quiser verificar os eventos próximos da FECAP Social.

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

O processo de avaliação é baseado na ISO 25010 e baseada no modelo acima:



4. REFERÊNCIAS BIBLIOGRÁFICAS

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