FECAP

PROJETO
ÁGUA PONTO
GRUPO 4

Requisitos da disciplina Modelagem de Software e Arquitetura de Sistemas

São Paulo

2024

INTEGRANTES DO PROJETO e RA'S

Lucca Gomes Vieira	-	23025239
Luigi Augusto Consentino Bezerra	-	23024667
Mateus Macedo Batista de Souza	-	23024410
Mateus Sousa Piccinin	-	22024043

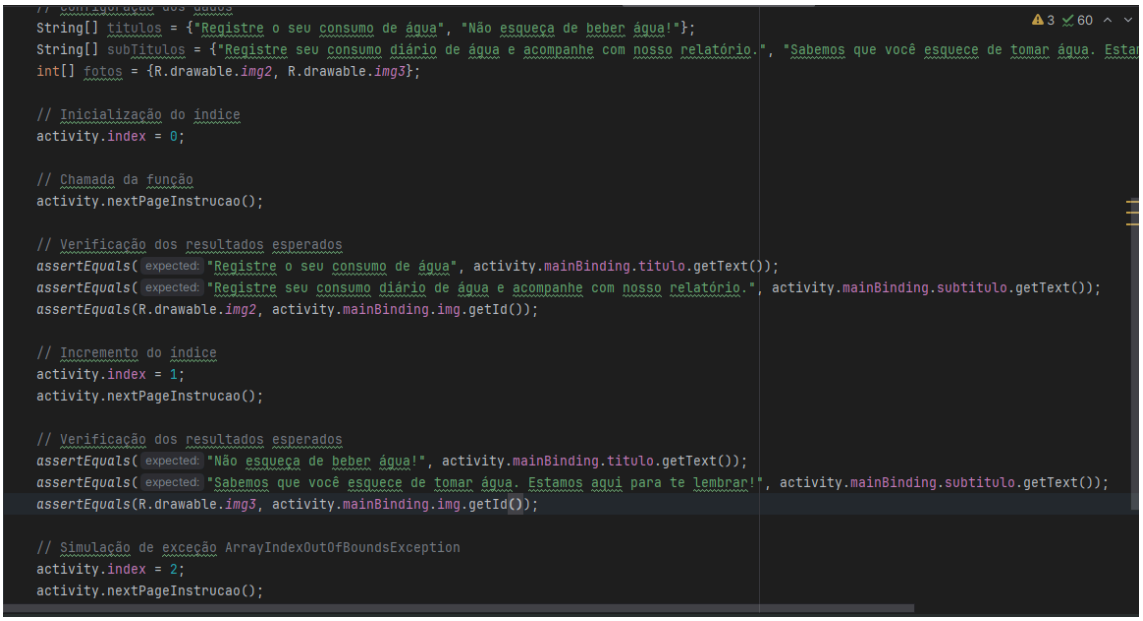
Conteúdo

1. Introdução

O Água Ponto é uma ferramenta inteligente e personalizada que ajuda os usuários a manter uma quantidade suficiente de água do corpo ao longo do dia. O aplicativo usa uma abordagem centrada do usuário para promover hábitos saudáveis de hidratação, fornecendo instruções personalizadas, lembretes frequentes e avaliações detalhadas do desempenho do usuário.

2. Teste de Software

2.1. Apresentar 2 testes unitários.



```
String[] titulos = {"Registre o seu consumo de água", "Não esqueça de beber água!"};
String[] subTitulos = {"Registre seu consumo diário de água e acompanhe com nosso relatório.", "Sabemos que você esquece de tomar água. Estar
int[] fotos = {R.drawable.img2, R.drawable.img3};

// Inicialização do índice
activity.index = 0;

// Chamada da função
activity.nextPageInstrucao();

// Verificação dos resultados esperados
assertEquals( expected: "Registre o seu consumo de água", activity.mainBinding.titulo.getText());
assertEquals( expected: "Registre seu consumo diário de água e acompanhe com nosso relatório.", activity.mainBinding.subtitulo.getText());
assertEquals(R.drawable.img2, activity.mainBinding.img.getId());

// Incremento do índice
activity.index = 1;
activity.nextPageInstrucao();

// Verificação dos resultados esperados
assertEquals( expected: "Não esqueça de beber água!", activity.mainBinding.titulo.getText());
assertEquals( expected: "Sabemos que você esquece de tomar água. Estamos aqui para te lembrar!", activity.mainBinding.subtitulo.getText());
assertEquals(R.drawable.img3, activity.mainBinding.img.getId());

// Simulação de exceção ArrayIndexOutOfBoundsException
activity.index = 2;
activity.nextPageInstrucao();
```

O teste unitário para a função “nextPageInstrucao” verificou se os elementos de imagem, título e subtítulo da interface do usuário foram corretamente atualizados nas duas primeiras chamadas da função e se, na terceira chamada, a exceção ArrayIndexOutOfBoundsException foi tratada adequadamente, finalizando a atividade e iniciando LoginActivity. Esse teste garantiu que a função se comporta conforme esperado em diferentes cenários, prevenindo erros potenciais e facilitando a manutenção futura do código.

```java
@RequiresApi(api = Build.VERSION_CODES.O)
@Test
public void testConfigurarGrafico() {
    RelatorioActivity activity = new RelatorioActivity();

    // Configuração dos dados de entrada
    ArrayList<RotinaModel> todasRotinasDiaria = new ArrayList<>();
    todasRotinasDiaria.add(new RotinaModel( ingestao: "2024-05-20",  mlIngerido: 2000)); // Exemplo de ingestão de 2 litros
    todasRotinasDiaria.add(new RotinaModel( ingestao: "2024-05-21",  mlIngerido: 1500)); // Exemplo de ingestão de 1.5 litros

    // Chamada da função a ser testada
    activity.todasRotinasDiaria = todasRotinasDiaria;
    activity.configurarGrafico();

    // Verificação dos resultados esperados
    assertNotNull(activity.mainBinding.chart.getData()); // Verifica se os dados do gráfico foram configurados
    assertEquals( expected: 2, activity.mainBinding.chart.getData().getEntryCount()); // Verifica se há duas entradas no gráfico

    // Verifica se os valores das entradas do gráfico correspondem aos dados fornecidos
    assertEquals( expected: 2f, activity.mainBinding.chart.getData().getDataSetByIndex(0).getEntryForIndex(0).getY(),  delta: 0.01f); /
    assertEquals( expected: 1.5f, activity.mainBinding.chart.getData().getDataSetByIndex(0).getEntryForIndex(1).getY(),  delta: 0.01f);
}
```

Este teste unitário verifica se a função “configurarGrafico” configura corretamente o gráfico com base nos dados fornecidos. Ele verifica se o gráfico possui duas entradas (uma para cada dia) e se os valores dessas entradas correspondem aos dados de ingestão de água fornecidos.

2.2. Apresentar 2 testes de componentes

```
@Test
public Boolean emailField_inputClick() {
    LoginActivity loginActivity = new LoginActivity();

    return loginActivity.edtEmail.performClick();
}
```

Verifica se o componente de entrada de E-mail da tela Login está clicável para o usuário.

```
2 usages
public boolean isValidEmail(CharSequence target) {
    return target != null && Patterns.EMAIL_ADDRESS.matcher(target).matches();
}
@Test
public void emailField_inputText() {
    LoginActivity loginActivity = new LoginActivity();
    String validEmail = "example@email.com";
    String invalidEmail = "invalidemail.com";

    assertTrue(isValidEmail(validEmail));
    assertFalse(isValidEmail(invalidEmail));
}
```

Verifica se o componente de entrada de E-mail da tela Login está recebendo a entrada de um E-mail válido.

2.3. Apresentar um teste de sistema.

Caso de Teste: Login no Aplicativo

Objetivo:

Verificar se o processo de login do aplicativo está funcionando corretamente, permitindo que os usuários ingressem com sucesso usando um email e senha válidos.

Pré-condições:

O dispositivo está conectado à internet.

O aplicativo está instalado e funcionando corretamente no dispositivo.

O usuário possui um email e senha válidos previamente cadastrados no sistema.

Passos:

Abra o aplicativo no dispositivo.

Na tela de login, insira um email válido e uma senha válida nos campos correspondentes.

Clique no botão "Entrar".

Resultados Esperados:

Se as credenciais estiverem corretas e houver conexão com a internet:

O aplicativo deve exibir uma mensagem de "Logando...".

O sistema deve verificar as credenciais inseridas.

Se as credenciais estiverem corretas e correspondentes a um usuário cadastrado:

O aplicativo deve redirecionar o usuário para a tela de Contagem de Água.

As informações do usuário, como ID e meta de consumo de água, devem ser armazenadas corretamente nas preferências do usuário.

Se as credenciais estiverem incorretas:

O aplicativo deve exibir uma mensagem de "Email ou Senha inválido.".

Se houver algum problema de conexão com o servidor:

O aplicativo deve exibir uma mensagem de "Problema de Conexão.".

Se houver algum erro de conexão:

O aplicativo deve exibir uma mensagem de "Erro de Conexão.".

Condições de Teste Adicionais:

Testar com diferentes combinações de credenciais (email e senha) válidas e inválidas.

Testar em diferentes condições de rede (Wi-Fi, 4G, 3G) para garantir que o aplicativo responda corretamente a problemas de conexão.

Testar em dispositivos com diferentes tamanhos e resoluções de tela para garantir que a interface do usuário seja responsiva e exibida corretamente em todos os dispositivos suportados.

3. Qualidade de Software

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

1. Confiabilidade

A confiabilidade refere-se à capacidade do software de funcionar corretamente sob condições específicas por um período determinado.

Aplicação no Projeto Integrador (PI):

Teste Unitário e de Integração: Implementar testes unitários e de integração para garantir que cada módulo do sistema funcione corretamente e que todos os módulos funcionem bem juntos.

Testes de Stress e de Carga: Realizar testes de stress e carga para avaliar o desempenho do sistema sob condições extremas e garantir que ele não falhe sob alta demanda.

Mecanismos de Recuperação de Falhas: Incluir mecanismos para recuperação automática de falhas e backup de dados, garantindo que o sistema possa se recuperar rapidamente de falhas inesperadas.

2. Usabilidade

A usabilidade refere-se à facilidade com que os usuários podem aprender a usar e operar o sistema de software.

Aplicação no Projeto Integrador (PI):

Design de Interface do Usuário (UI): Criar uma interface de usuário intuitiva e amigável, com botões e fluxos de navegação claros.

Feedback do Usuário: Realizar testes de usabilidade com usuários reais e coletar feedback para melhorar a interface e a experiência do usuário.

Documentação e Tutoriais: Fornecer documentação clara e tutoriais dentro do aplicativo para ajudar os usuários a entender como usar as funcionalidades.

3. Desempenho

O desempenho refere-se à capacidade do software de realizar suas funções dentro de determinados limites de tempo e recursos.

Aplicação no Projeto Integrador (PI)

Otimização de Código: Garantir que o código seja otimizado para velocidade e eficiência, reduzindo a latência e o tempo de resposta.

Monitoramento Contínuo: Implementar monitoramento contínuo do desempenho do sistema para identificar e resolver gargalos ou problemas de desempenho.

Cache e Armazenamento Temporário: Utilizar técnicas de cache para melhorar a velocidade de acesso a dados frequentes e reduzir a carga sobre os sistemas de backend.

4. Segurança

A segurança refere-se à capacidade do software de proteger dados e resistir a ataques maliciosos.

Autenticação e Autorização: Implementar fortes mecanismos de autenticação (como OAuth, JWT) e autorização para garantir que apenas usuários autorizados possam acessar certas funcionalidades e dados.

Criptografia: Utilizar criptografia para proteger dados sensíveis tanto em trânsito quanto em repouso.

Auditoria e Log: Implementar logs de auditoria para monitorar atividades no sistema e detectar possíveis tentativas de invasão ou abuso.

3.2. Apresentar um Modelo de qualidade de software

Modelo no Projeto Integrador (PI)

Funcionalidade:

Adequação Funcional: Implementação de todas as funcionalidades especificadas para o login (como autenticação segura e feedback de erro adequado).

Segurança: Criptografia das senhas e uso de mecanismos seguros de armazenamento de credenciais.

Apreensibilidade: Interface de login intuitiva, com campos de entrada claramente rotulados.

Operabilidade: Mensagens de erro claras e ações corretivas sugeridas para os usuários.

Maturidade: Testes unitários e de integração para garantir que o sistema de login funcione sem falhas.

Disponibilidade: Garantir que o serviço de login esteja disponível durante os períodos de teste**.**

Comportamento em Relação ao Tempo: Testes de tempo de resposta para garantir que o processo de login seja rápido.

Utilização de Recursos: Análise para garantir que o processo de login não consuma recursos excessivos do dispositivo.

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

Plano de Gerenciamento de Qualidade de Software

1. Introdução

Objetivo: Descrever o propósito do plano de gerenciamento de qualidade.

Escopo: Definir os limites do projeto e os aspectos de qualidade que serão gerenciados.

2. Objetivos de Qualidade

Definição dos Objetivos: Estabelecer objetivos claros e mensuráveis de qualidade para o projeto, como conformidade com requisitos funcionais, desempenho, segurança, e usabilidade.

3. Responsabilidades

Equipe de Qualidade: Identificar os membros da equipe responsáveis pela qualidade e suas funções específicas.

Responsabilidades: Definir as responsabilidades de cada membro da equipe, incluindo desenvolvedores, testadores, gerentes de projeto e stakeholders.

4. Processos de Qualidade

Planejamento de Qualidade: Definir os processos e atividades que serão usados para garantir a qualidade, como revisões de código, testes de unidade, integração contínua, testes de sistema e aceitação do usuário.

Controle de Qualidade: Descrever as atividades de controle de qualidade, como revisões de documentos, inspeções, auditorias e validações.

Garantia de Qualidade: Descrever os processos de garantia de qualidade, incluindo revisões de processo, auditorias de qualidade e avaliações de conformidade com os padrões de qualidade.

5. Ferramentas e Tecnologias

Ferramentas de Teste: Listar as ferramentas de teste que serão usadas, como JUnit, Selenium, Appium, etc.

Ferramentas de Integração Contínua: Descrever as ferramentas de CI/CD que serão usadas, como Jenkins, GitLab CI, CircleCI, etc.

Ferramentas de Gestão de Requisitos: Especificar ferramentas como Jira, Trello, ou Asana para o gerenciamento de requisitos e rastreamento de problemas.

6. Métricas de Qualidade

Definição de Métricas: Definir as métricas que serão usadas para medir a qualidade, como cobertura de teste, número de defeitos, tempo de resposta, e satisfação do usuário.

Coleta de Métricas: Descrever como as métricas serão coletadas e analisadas.

Análise de Métricas: Explicar como as métricas serão usadas para melhorar a qualidade do software.

7. Procedimentos de Revisão e Auditoria

Revisões de Código: Definir o processo de revisão de código, incluindo critérios para aprovação.

Inspeções de Documentos: Descrever o processo para revisar e aprovar documentos de projeto, requisitos e design.

Auditorias de Qualidade: Descrever como e quando as auditorias de qualidade serão realizadas para garantir a conformidade com os processos de qualidade.

8. Gerenciamento de Riscos de Qualidade

Identificação de Riscos: Listar possíveis riscos que podem afetar a qualidade do software.

Avaliação de Riscos: Avaliar a probabilidade e o impacto de cada risco.

Mitigação de Riscos: Definir estratégias para mitigar os riscos identificados.

9. Plano de Treinamento

Necessidades de Treinamento: Identificar as necessidades de treinamento da equipe para garantir que todos tenham as habilidades e conhecimentos necessários para manter a qualidade.

Programas de Treinamento: Descrever os programas de treinamento que serão implementados.

10. Cronograma e Orçamento

Cronograma de Atividades de Qualidade: Definir um cronograma detalhado para as atividades de qualidade, incluindo marcos importantes e prazos.

Orçamento de Qualidade: Estimar o custo das atividades de qualidade, incluindo ferramentas, treinamento e tempo de trabalho.

11. Comunicação e Relatórios de Qualidade

Planos de Comunicação: Descrever como a comunicação sobre questões de qualidade será gerenciada entre a equipe de projeto e os stakeholders.

Relatórios de Qualidade: Definir a frequência e o formato dos relatórios de qualidade.

4. REFERÊNCIAS BIBLIOGRÁFICAS

Testes Automáticos + Curso COMPLETO de Teste de Software
https://www.udemy.com/course/teste-software-completo-testes-automaticos/?couponCode=LEADERSALE24A

Características de Qualidade ISO 25010 - Como saber se um software tem qualidade? Entenda!
https://www.youtube.com/watch?v=IvqX5DQH3mk