UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ

Plano de Medição de Qualidade

Disciplina: Projeto Integrado em Engenharia de Software 3
Professor: Camilo Almendra
Equipe: Vinícius Amorim Sales Lopes - 499641
Ruan Diego Guimarães do Nascimento - 495301
Francisco Edvaldo de Oliveira Junior - 494418

ÍNDICE

GLOSSÁRIO

Siglas	Definição
QA	Quality Assurance
app	Aplicativo

HISTÓRICO DE REVISÕES

Data	Versão	Descrição	Responsável
05/10/2022	1.0	Criação do documento.	Vinícius Sales, Ruan Diego, Edvaldo Oliveira
11/10/2022	1.1	Preenchimento de toda a seção 1, seção 2 e seção 3	Vinícius Sales, Ruan Diego, Edvaldo Oliveira
16/10/2022	1.2	Revisão do documento	Vinícius Sales, Ruan Diego, Edvaldo Oliveira

1. INTRODUÇÃO

Para fornecer uma visão mais abrangente do sistema a ser analisado, foi criado o presente documento visando a apresentação das análises de qualidade de forma técnica e detalhada. Essa análise é destinada aqueles que desejam conhecer as métricas mais detalhadas do sistema, que será descrito mais à frente. Esses podem ser desenvolvedores, analistas de qualidade e pessoas interessadas no produto.

1.1. Descrição do produto a ser avaliado

Este documento visa fornecer o plano de medição do sistema "qFashion". O mesmo se trata de um sistema de cadastro de lojas para vendas de roupas para pequenos e grandes empreendedores, visando o sertão central do Ceará. O intuito do sistema, é a realização de vendas de roupas por meio de um aplicativo móvel que faz o cadastro de novas lojas e clientes e que consiga atender as demandas de compras e vendas de roupas na região.

Por conta de se tratar de uma aplicação móvel inteiramente funcional, podemos escolher qual ambiente será analisado. Assim como foi descrito antes, iremos analisar o sistema como um todo, que consiste na parte do back-end e front-end do mesmo.

O sistema descrito será avaliado através de versões do sistema. Essas, serão selecionadas de acordo com tags fornecidas através do sistema de criação de tags do GitHub. As tags criadas serão referenciadas como versões do sistema em três pontos distintos do ciclo de desenvolvimento do mesmo. A partir dessas versões, podemos fazer a análise da qualidade do código através dos métodos de avaliação que serão descritos mais à frente. E com isso, será possível ver a evolução do sistema, podendo ver o quão maduro ou não o mesmo se tornou durante o período de desenvolvido analisado.

1.2. Objetivos da avaliação

O objetivo da avaliação é validar métricas de **Manutenibilidade**, **Eficiência** e **Usabilidade** do aplicativo proposto, "*qFashion*" através das suas sub-características **Testabilidade**: para garantir a capacidade de testar e validar o sistema após modificações, **Modularidade**: para garantir que as mudanças em algum componente tenham menor impacto em outros componentes, **Operabilidade**, para garantir a facilidade de uso da aplicação por parte dos usuários E por fim a subcaracterística **Tempo de Resposta**, para garantir que a aplicação tenha um bom comportamento diante da demanda dos usuários. Os desenvolvedores serão solicitados para realizar as medições do sistema.

Analisar	Manutenibilidade
Para o propósito de	Analisar a facilidade do software de ser modificado a fim de corrigir defeitos e realizar novas implementações e identificar quando ocorrem falhas e se elas são críticas para o sistema quando ocorrem.
Com respeito a	Testabilidade, Modularidade
Do ponto de vista	Dos desenvolvedores e QAs
No contexto de	Understand e SonarQube

Analisar	Usabilidade
Para o propósito de	Analisar o comportamento da aplicação durante o seu uso, com o intuito de verificar se está funcionando dentro da conformidade.
Com respeito a	Operabilidade
Do ponto de vista	Dos usuários da aplicação

No contexto de	Da aplicação

Analisar	Eficiência
Para o propósito de	Analisar a eficiência da aplicação em responder as ações dos usuários durante o uso da aplicação. Analisando o tempo que a aplicação demora a responder aos comandos executados.
Com respeito a	Tempo de Resposta
Do ponto de vista	Usuário
No contexto de	Da aplicação utilizando o driver Appium

2. MÉTODO

Nos tópicos abaixo, abordamos quais as tarefas que devem ser realizadas pelos participantes, qual ambiente será testado e as ferramentas utilizadas. Além disso, foi realizada uma descrição de todo esse procedimento.

2.1. Ambiente de avaliação

- As medidas a serem coletadas foram : quantidade de linhas, métodos testados, além de taxas de funções, e arquivos acima do permitido, tempo para conclusão de tarefa específica e a taxa de consistência operacional em uso;
- Os ambiente de execução das análises foi o de desenvolvimento, em um computador desktop Windows 11, que rodava o código e emula o app, além de testes de usabilidade feitos nos celulares Android dos usuários;
- As ferramentas SonarQube, Appium e Understand serão as utilizadas para coleta.

2.2. Procedimentos da Avaliação

Os participantes foram informados de que a Manutenibilidade (Testabilidade e Modularidade), a Usabilidade (Operabilidade) e Eficiência (Tempo de Resposta) do app seriam analisadas apenas por motivo de coleta de dados para que pudéssemos analisar a facilidade do software de ser modificado a fim de corrigir defeitos e realizar novas implementações, bem como para saber o grau de facilidade do uso do sistema e por fim, para testarmos seu desempenho em uso. Dessa forma, os desenvolvedores e testadores foram informados que isso não era um teste para avaliar especificamente o seu desempenho individual, mas sim, para buscar entender melhor o código e realizar melhorias que irão otimizar a continuação do desenvolvimento.

Em seguida, o avaliador explicou para os desenvolvedores e QAs como seria a avaliação, apresentando o que deveria ser coletado e as ferramentas que possibilitam esse processo. Além disso, foi feita uma pesquisa com os participantes para descobrir se os mesmos possuem experiência com as ferramentas que serão utilizadas.

2.3. Medidas de Software

Abaixo foram descritas as medidas de software que serão coletadas, que são a Testabilidade, Modularidade, Operabilidade e Tempo de resposta.

2.3.1. Testabilidade

A Testabilidade examina as diferentes probabilidades e características comportamentais que levam o código a falhar se alguma coisa estiver incorreta. Ou seja, a capacidade de testar um sistema modificado, tanto quanto as novas funcionalidades quanto as não afetadas diretamente pela modificação.

Nome	**Descrição**	**Função de Medição**	**Método**
Taxa de **linhas** de código testadas.	Qual a **taxa de linhas** de código que foram chamadas pelo menos uma vez durante os testes unitários ?	X = (LT/T) * 100 LT = Número de linhas testadas T = Número total de linhas existentes na classe	Análise de código usando a ferramenta SonarQube e driver Appium.
Taxa de **métodos** testados.	Qual a **taxa de métodos** que foram chamadas pelo menos uma vez em uma classe durante os testes unitários ?	X = (MT/T) * 100 MT = Número de métodos testados T = Número total de métodos existentes na classe	Análise de código usando a ferramenta SonarQube.

2.3.2. Modularidade

As medidas de modularidade abaixo, foram utilizadas para avaliar o grau em que as mudanças em um componente podem impactar em outros componentes, buscando garantir o menor impacto possível.

Para realizar a análise dessa subcaracterística, levamos em consideração os argumentos apresentados por Martin Fowler sobre características que levam a um código de qualidade. Dentre suas análises, destacamos seu comentário sobre a quantidade de linhas em funções e o impacto na complexidade do projeto. A partir desse estudo, desenvolvemos métricas visando a qualidade de código do projeto.

Nome	**Descrição**	**Função de Medição**	**Método**
Taxa de **funções** com tamanho acima do permitido.	Quantas **funções** passam do **limite** de **linhas** estipulado pela equipe?	X = (NF / NT) * 100 NF = Número de funções que passam do limite de linhas por função. NT = Número total de funções.	Análise de código usando a ferramenta Understand SciTools.
Taxa de **arquivos** com tamanho acima do permitido.	Quantos **arquivos** passam do **limite** de **linhas** estipulado pela equipe?	X = (NA / NT) * 100 NA = Número de arquivos que passam do limite de linhas por arquivo. NT = Número total de arquivos.	Análise de código usando a ferramenta Understand SciTools.

2.3.3. Operabilidade

A operabilidade mede o grau de facilidade com a qual um produto ou sistema é operado e/ou controlado pelo usuário. Aqui ela será utilizada para analisar se o app está de acordo com o propósito pelo qual ele foi desenvolvido através de uma análise da consistência dos componentes de interface do usuário.

Nome	Descrição	Função de Medição	Método
Taxa de consistência operacional em uso	Quão consistentes são os componentes da interface do usuário?	NOI = Número de operações que o usuário considerou inaceitavelmente inconsistentes com a expectativa do usuário	Análise de resultados com teste de usabilidade com o usuário

2.3.4. Tempo de Resposta

O tempo de resposta tem por objetivo medir o grau em que os tempos de resposta e de processamento e taxas de transferência de um produto ou sistema, no desempenho das suas funções, atende aos requisitos. Tendo em vista seu objetivo, ele servirá para que possa ser feito uma medição da eficiência do app ao executar as suas principais funcionalidades.

Nome	Descrição	Função de Medição	Método
Tempo para conclusão de uma tarefa específica	Qual é o tempo necessário para concluir uma tarefa específica?	T = Tf - Ti Tf = Tempo para obter o resultado Ti = Tempo da entrada do comando finalizada	Análise do desempenho utilizando o driver Appium para coletar o resultado nos dispositivos

2.4. Procedimentos de Interpretação

As medidas são valores reais e quantitativos, retirados de ambas as ferramentas; SonarQube e Understand e do driver Appium. Como valor hipotético será definido um limite de linhas para a medição, baseados na média total existente no código e na referência literária de Martin C. Fowler citada anteriormente. Tais valores foram decididos de acordo com a análise da equipe. Dessa forma, temos que o limite desejável de linhas por funções, que será de 12 linhas. Já o limite desejável para linhas para classes será de 100 linhas e para arquivos será considerado o limite de 120 linhas por arquivo.

Como representação gráfica serão utilizados os gráficos de barra da ferramenta Understand e os gráficos e relatórios do SonarQube.

Para Operabilidade definimos em consenso que o nível bom seria ter menos que 03 operações inconsistentes, regular entre 03 e 10 operações inconsistentes e ruim acima de 10 operações inconsistentes.

Quanto ao Tempo de Resposta em melhor caso tempos abaixo de 1 segundo, regular entre 01 e 10 segundos e ruim acima de 10 segundos.

3. REFERÊNCIAS

ISO/IEC 25000. Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. v. 2005, 2005.

ISO/IEC 9126. Software Engineering – Product Quality – Part 1. 2001.

ISO/IEC 9126-2. Software Engineering – Product Quality – Part 2: External Metrics. 2001.

FOWLER Martin. FunctionLength. martinFowler.com, 2016. Disponível em: <https://martinfowler.com/bliki/FunctionLength.html>. Acesso em: 13 de Novembro de 2021.