**Engenharia de Software II**

O processo de qualidade de software foi amplamente abordado. Esse tema engloba duas principais questões: A qualidade de produto (o software em si) e a qualidade do processo (como o software é desenvolvido). Para este aplicativo, a qualidade de produto se traduz em garantir que ele seja eficaz, seguro, eficiente e fácil de ser utilizado pelos usuários. Por outro lado, a qualidade do processo, está conectada a práticas e métodos utilizados na criação e manutenção do aplicativo. Com a utilização de metodologias ágeis (Scrum, Kanban Extreme Programming), alinhamento de métricas como o desempenho, funcionalidade, e responsividade, tudo para otimizar e qualificar o processo de desenvolvimento do software.

A diante, a verificação e validação de software é de extrema importância para a produção do aplicativo, já que visa reconhecer o ciclo de vida do software e entender os requisitos especificados, funcionando assim como esperado. O objetivo principal é reduzir erros e deixar o sistema de forma mais confiável. Nele ocorre revisões de código, inspeções, testes unitários e testes de integração, sem contar com o papel fundamental dos ambientes de desenvolvimento e de testes que ajudam a identificar erros e falhas de maneiras inicial.

O MVP (Produto Mínimo Viável), busca implementar um sistema simples, com um conjunto de requisitos mínimos, porém eles necessitam ser suficientes para testar a viabilidade de continuar investindo no seu desenvolvimento. O MVP, tem 3 passos extremamente importantes: Construir (onde se tem a ideia do produto, e então implementa-se um MVP para testá-la), medir (nesse segundo passo, o MVP é disponibilizado para o uso por clientes reais com o intuito de coletar dados sobre a sua viabilidade) e aprender (o terceiro e último passo onde as métricas coletadas são analisadas e geram o que se denomina de “aprendizado validado”). Portanto, o MVP não precisa ser um software real, implementado em uma linguagem de programação, integrado a outros sistemas ou com seu próprio banco de dados.

O Teste Unitário desempenha um papel crucial no desenvolvimento de software. Pois ajudam a encontrar e corrigir erros de forma precoce, garantindo qualidade e confiabilidade do software. Também, permitem que sejam feitas alterações e adições no código de forma segura, evitando efeitos indesejados. É uma técnica de teste de software que visa verificar o funcionamento de unidades individuais de um código. Assim, isola a unidade a ser testada e retorna entradas específicas para confirmar se a saída é a que espera. São executados de forma rápida, sem depender de outras partes do sistema, focando assim na manutenção e legibilidade do código. Outro ponto extremamente importante, é que os testes unitários são objetivados na redução dos custos, pois ao identificar problemas o mais cedo possível, a correção poderá ser feita de forma mais rápida, evitando problemas maiores e custosos no futuro.

A manutenção de software é o conjunto de atividades realizadas após a entrega de um produto, tendo seu objetivo em corrigir defeitos e adaptar o sistema a novos ambientes, melhorando sua performance e evitando problemas futuros. Tradicionalmente os ajustes são feitos, sempre após uma implantação do software, em intervalos regulares, quando um artefato ou produto é feito. Com tudo, a manutenção tem seus tipos: Emergencial (Coloca o sistema em seu estado de uso; feito de forma superficial e não programada; vai necessitar de uma manutenção corretiva posteriormente), Corretiva (Acontece após a detecção de um defeito que gerou um erro; faz o requisito funcionar como o esperado), Preventiva (Antes de ocorrer um erro, com um problema latente identificado, é proativa), Adaptativa (Adapta o software ao ambiente no qual ele está inserido), Perfectiva (Introduz um novo requisito no software, funcional ou não-funcional). Não deixando de lado os conceitos importantes da manutenção, a Refatoração é muito importante, visando mudar a estrutura interna do software sem necessariamente mudar o comportamento externo, deixando mais fácil de entender ou modificar. A Documentação é outro fator importante, em que seu objetivo é ser simplificada, estimulando a eficiência, criar e manter a documentação atualizada, facilita a manutenção. Já a Engenharia Reversa, visa a partir do código extrair a documentação. Por fim a Reengenharia que é objetivada pela migração de Hardware e Software, como a troca de linguagem de programação.

“A manutenção representa mais de 60% do custo total do software”. (Roger S. Pressman, em “Engenharia de Software”). Em uma frase dita por um dos autores mais respeitados na área de Engenharia de Software, fica explícito como os custos sobrepõem toda a produção e aparecem comumente após a entrega do produto. De certa forma, quebra um paradigma sobre dizer que o desenvolvimento acaba sendo o centro dos gastos. Pode-se concluir que é de extrema importância pensar incialmente na qualidade de código, documentação, testes, modularidade e arquitetura, pois eles influenciam diretamente na facilidade e no custo da manutenção futura.

teste

**Criar chamados**

**Sumário:**  Funcionário cria chamados para obter sua assistência técnica solicitada.

**Ator Primário:** Funcionário.   
   
**Precondições:** Funcionário estar registrado e autenticado no aplicativo da empresa.

**Fluxo Principal:**

1. O funcionário acessa a página de criar chamado.

2. O funcionário define o nível de prioridade do chamado.

3. O funcionário descreve seu chamado.

4. O funcionário seleciona o tipo do seu chamado.

5. O funcionário cria um chamado.

**Extensão do fluxo:**

2b. O sistema apresenta um “\*”, indicando que o campo é obrigatório.

3b. O sistema apresenta um “\*”, indicando que o campo é obrigatório.

4b. O sistema apresenta um “\*”, indicando que o campo é obrigatório.

5b. O sistema apresenta um “\*”, em todos os campos que são obrigatórios.

**Atender Chamados**

Sumário: Técnico atende chamado feito por um funcionário

Ator Primário: Técnico.

Precondições: O Técnico estar autenticado no sistema e um chamado ter sido realizado.

Fluxo Principal:   
1. O Técnico acessa a aba “Atender”.

2. O Técnico seleciona o chamado clicando no botão “atender”.

3. O chamado é atendido e tem seu status alterado.

4. O sistema retorna ao menu.

Extensão do fluxo:   
2b. O sistema não apresenta chamados disponíveis.

**Criar conta**

Sumário: Usuário cria sua conta para utilização do aplicativo.   
   
Ator Primário: Usuário.

Precondições: Usuário necessita ter seu registro empresarial.   
   
Fluxo principal:   
1. O usuário acessa a página de “Criar conta”.

2. O usuário define se é Técnico ou Funcionário.

3. O usuário preenche o campo de “Nome Completo”.

4. O usuário preenche o campo de “Registro empresarial”.

5. O usuário preenche o campo de “e-mail”.

6. O usuário cria sua senha.

7. O usuário cria sua conta.

Fluxo alternativo:

2b. Sistema informa para selecionar uma das opções.   
3b. O sistema apresenta um “\*”, indicando que o campo é obrigatório.

4b. O sistema apresenta um “\*”, indicando que o campo é obrigatório.

5b. O sistema apresenta um “\*”, indicando que o campo é obrigatório.

6b. O sistema apresenta um “\*”, indicando que o campo é obrigatório.

**Bibliografia:**

[1] Vention. Software maintenance costs: tipos for keeping costs low and quality high. Acesso em 2025. Disponível em: [https://ventionteams.com/enterprise/software-maintenance-costs](https://ventionteams.com/enterprise/software-maintenance-costs" \t "https://brc-word-edit.officeapps.live.com/we/_blank)

[2] Appinventiv. How Much does it cost to maintain a software? All you need to know. Acesso em: 2025. Disponível em:

[https://appinventiv.com/blog/software-maintenance-cost/?utm\_source=chatgpt.com](https://appinventiv.com/blog/software-maintenance-cost/?utm_source=chatgpt.com" \t "https://brc-word-edit.officeapps.live.com/we/_blank)

[3] AddWeb Solutions. Software maintenance costs a comprehensive guide. Acesso em: 2025. Disponível em: [https://www.addwebsolution.com/blog/software-maintenance-costs?utm\_source=chatgpt.com](https://www.addwebsolution.com/blog/software-maintenance-costs?utm_source=chatgpt.com" \t "https://brc-word-edit.officeapps.live.com/we/_blank)

[4] Bennett, K. K. and V. T. Rajlich (2000). Softwre maintenance and Evolution: a roadmap. International Conference on Software Engineering, The Future of Software Engineering, Limerick, Ireland, ACM Press.

Prof. Leonardo Murta. Aula de Manutenção de Software. UFF. 2023.

Lyncas. Manutenção de software: garanta o funcionamento e a eficiência contínua. Acesso em: 2025. Disponível em:

https://lyncas.net/blog/manutencao-de-software-eficiencia-continua/ 