**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS**

**ESCOLA POLITÉCNICA**

**GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO**

Logotipo, nome da empresa

Descrição gerada automaticamente

**OTIMIZAÇÃO DO DESENVOLVIMENTO DE SOFTWARE: NORMAS E PRÁTICAS PARA FACILITAR MANUTENÇÕES FUTURAS DE CÓDIGO**

**ISAAC MONTEIRO SÉRVIO BEN RIBEIRO**

**E**

**HIGOR CAVALCANTE DOS ANJOS**

**GOIÂNIA,**

**2023**

**OTIMIZAÇÃO DO DESENVOLVIMENTO DE SOFTWARE: NORMAS E PRÁTICAS PARA FACILITAR MANUTENÇÕES FUTURAS DE CÓDIGO**

Projeto de Pesquisa apresentado à disciplina Metodologia, da Escola Politécnica, Curso de Engenharia da computação, da Pontifícia Universidade Católica de Goiás (PUC GOIÁS). Profa. Orientadora – Dra. Solange da Silva.

**GOIÂNIA,**

**2023**

1. **Introdução**

A área de desenvolvimento de software está em constante evolução, buscando aprimorar a gerenciabilidade e manutenabilidade dos sistemas. Lint code é um termo que se refere a ferramentas de análise de código estático que ajudam a identificar erros de programação, bugs, erros estilísticos e construções suspeitas. O termo se origina de um utilitário Unix que examinava código-fonte em linguagem C (Ian F. Darwin). As ferramentas de lint code podem ser usadas para ajudar a identificar problemas de código que podem ser candidatos à refatoração (Kvam, K.; Lie, R; Bakkelund, 2005). A otimização do desenvolvimento de software é um objetivo crucial, visando simplificar a manutenção do código (Pressman, 2019). A complexidade inerente à manutenção de software frequentemente encontra obstáculos decorrentes de mudanças na equipe de documentação, durante o processo de desenvolvimento e diante da evolução constante das tecnologias. Tais desafios contribuem, muitas vezes, para custos adicionais no desenvolvimento e na atualização de projetos. (Sommerville, I)

A área de lint code é relativamente antiga, com o primeiro utilitário lint sendo desenvolvido na década de 1970. No entanto, a área tem experimentado um crescimento significativo nos últimos anos, com o desenvolvimento de novas ferramentas, técnicas e pesquisas. Por exemplo, o ESLint é uma ferramenta de lint para JavaScript e TypeScript que é altamente personalizável e pode ser usada para verificar uma ampla gama de erros e padrões de estilo. Existe tambem técnicas de aprendizado de máquina estão sendo usadas para desenvolver ferramentas de lint que podem aprender com os dados de treinamento para melhorar a precisão da identificação de erros, e previsão de código aumentando a produtividade, o mais famoso sendo o Copilot da Microsoft.

E relevante estudar este tema porque as ferramentas lint code ajuda a identificar erros de programação que potencialmente resultam em questões de segurança, desempenho, funcionalidade e no aumento da complexidade. Essas ferramentas desempenham um papel fundamental na prevenção de problemas sérios que poderiam comprometer a qualidade do software. Além disso, ao automatizarem a análise de código, as ferramentas de lint code possibilitam uma diminuição de complexidade de leitura do projeto sendo uma curva mais suave de aprendizado uma economia significativa de tempo e esforço para os desenvolvedores. Esse ganho de eficiência, por sua vez, permite que os desenvolvedores concentrem seus esforços em outras atividades relevantes.

Diante do contexto, este projeto visa respondar a questão de pesquisa: **"Como aprimorar o desenvolvimento de software implementando padrões e práticas que facilitem a manutenção futura do código?"**

1. **Objetivos**
   1. **Geral**

* Desenvolver conjunto de diretrizes e métodos para otimizar a manutenção futura do código no processo de desenvolvimento de software.
  1. **Objetivos específicos**
* Realizar uma revisão bibliográfica abrangente sobre as melhores práticas de desenvolvimento de software.
* Identificar os principais desafios e obstáculos encontrados no desenvolvimento de software que dificultam as futuras manutenções de código.
* Propor diretrizes específicas que possam ser incorporadas ao ciclo de desenvolvimento de software para melhorar a manutenibilidade do código.
* Validar as normas e práticas propostas por meio de estudos de caso em projetos reais de desenvolvimento de software.

1. **Métodos**

Esta pesquisa, por sua natureza, configura-se como um resumo temático, concentrando-se na organização de uma área específica de conhecimento ao apontar sua evolução histórica e o estado atual da arte (VIESSLI, Kaliane; SILVA, Arielyn; SANTOS, Gustavo, 2020).

De acordo com seus objetivos, a pesquisa é classificada como exploratória, caracterizada pelo fato de o autor não necessariamente possuir uma hipótese ou objetivo claramente definido. Frequentemente, essa abordagem é vista como o estágio inicial de um processo de pesquisa mais extenso.

Quanto aos procedimentos técnicos, o estudo adota uma abordagem de pesquisa bibliográfica e documental. A revisão bibliográfica implica na análise de materiais previamente publicados, como livros, teses, recursos online e revistas, entre outros. Sua principal vantagem reside na capacidade de oferecer uma visão abrangente de diversos fenômenos, ultrapassando as limitações que poderiam surgir ao investigar diretamente.

* Escolha do tema:
  + Otimização Do Desenvolvimento De Software Normas E Praticas Para Facilitar Manutenções Futuras De Código
* Pesquisa Bibliográfica:
  + Identificar as principais fontes de pesquisa, incluindo livros, artigos acadêmicos e relatórios técnicos.
  + Realizar uma revisão sistemática da literatura para coletar informações relevantes sobre as melhores práticas de desenvolvimento de software.
  + Sintetizar as descobertas e identificar lacunas no conhecimento existente.
* Análise do conteúdo:
  + Selecionar projetos de desenvolvimento de software de diferentes domínios.
  + Analisar a situação atual de cada projeto, identificando problemas de manutenibilidade de código.
  + Fichamento: O fichamento inclui a análise de documentos com contribuições médias ou baixas, proporcionando um aprimoramento nas reflexões do autor.
* Pesquisa de Campo:
  + Coletar dados de empresas de desenvolvimento de software por meio de questionários e entrevistas.
  + Analisar as respostas para identificar os principais desafios enfrentados pelas empresas em relação à manutenção de código.
  + Validar as normas e práticas propostas com base nas necessidades reais das empresas.

1. **Resultados esperados**

Espera-se que os resultados deste trabalho possam AUXILIAR:

* Desenvolvedores de software na aplicação de normas e práticas que facilitarão a manutenção futura de código.
* Empresas de tecnologia da informação na melhoria da eficiência e sustentabilidade de seus produtos de software.
* A comunidade acadêmica, fornecendo diretrizes claras e conhecimento atualizado sobre otimização de desenvolvimento de software.
* Contribuir para a redução de custos e tempo de manutenção de software em projetos reais.
* Estimular o uso de práticas de desenvolvimento mais sustentáveis e eficazes em toda a indústria de software.

1. **Lista de atividades**

* Revisão bibliográfica
* Identificação de projetos
* Análise do conteúdos
* Coleta de dados de empresas de desenvolvimento de software
* Descrever normas e práticas
* Análise dos resultados
* Escrita do TCC 1
* Defender TCC 1
* Correções necessária
* Escrever TCC2
* Defender TCC2
* Corrigir e entregar TCC2

1. **CRONOGRAMA DE EXECUÇÃO DO PLANO DE TRABALHO**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Ano** | **2023** | | | | | **2024** | | | | | | | |
| **Atividade/Mês** | **AGO** | **SET** | **OUT** | **NOV** | **DEZ** | **JAN** | **FEV** | **MAR** | **ABR** | **MAI** | **JUN** | **JUL** |
| **Revisão bibliográfica** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Identificação de projetos** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Análise do conteúdos** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Coleta de dados de empresas de desenvolvimento de software** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Descrever normas e práticas** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Análise dos resultados** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Escrita do TCC 1** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Defender TCC 1** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Correções necessária** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Escrever TCC2** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Defender TCC2** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Corrigir e entregar TCC2** |  |  |  |  |  |  |  |  |  |  |  |  |

1. **REFERÊNCIAS**

Pressman, R. S. (2019). **Engenharia de Software: Uma Abordagem Profissional.** AMGH Editora.

Ian F. Darwin (1991), **Checking C programs with lint.** O'Reilly & Associates

KVAM, K.; LIE, R; BAKKELUND, **D. Legacy system exorcism by Pareto's**

**principle**. OOPSLA’05, 2005.

Sommerville, I. (2018). **Engenharia de Software.** Pearson.

VIESSLI, Kaliane; SILVA, Arielyn; SANTOS, Gustavo. (2020) **Estimativa de Esforço em Atividades de Manutenção de Software: Um Mapeamento Sistemático.**

Melluzzi Neto, Guilherme; França, Victor José A. T. de Melo; Cavalcante, Bruno Henrique; Balancieri, Renato; Leal, Gislaine Camila Lapasini. (2018) **Resultados da implantação de CMMI e MPS-BR em empresas de desenvolvimento e manutenção de software: a visão da alta gestão.**