UNIVERSIDADE ESTADUAL DO PIAUÍ
CENTRO DE TECNOLOGIA E URBANISMO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Ada Beatriz de Oliveira Araujo

Inteligência Artificial na Documentação de Sistemas Legados: Uma Abordagem Comparativa de Ferramentas

TERESINA
2025

Ada Beatriz de Oliveira Araujo

Inteligência Artificial na Documentação de Sistemas Legados: Uma Abordagem Comparativa de Ferramentas

Monografia de Trabalho de Conclusão de Curso apresentado na Universidade Estadual do Piauí – UESPI como parte dos requisitos para conclusão do Curso de Bacharelado em Ciência da Computação.

Orientador: Prof. Dr. Thiago Carvalho de Sousa

TERESINA

2025

A658i Araújo, Ada Beatriz de Oliveira.
Inteligência artificial na documentação de sistemas legados: uma abordagem comparativa de ferramentas / Ada Beatriz de Oliveira Araújo. - 2025.
75 f.: il.

Monografia (graduação) - Universidade Estadual do Piauí - UESPI, Bacharelado em Ciência da Computação, Campus Poeta Torquato Neto, Teresina-PI, 2025.
"Orientador: Prof. Dr. Thiago Carvalho de Sousa".

1. Engenharia de Software. 2. Sistemas Legados. 3. Inteligência Artificial. 4. Documentação de Software. I. Sousa, Thiago Carvalho de . II. Título.

CDD 006.3

Ficha elaborada pelo Serviço de Catalogação da Biblioteca da UESPI
Francisca Carine Farias Costa (Bibliotecário) CRB-3ª/1637

Inteligência Artificial na Documentação de Sistemas Legados: Uma Abordagem Comparativa de Ferramentas

Ada Beatriz de Oliveira Araujo

Monografia de Trabalho de Conclusão de Curso apresentado na Universidade Estadual do Piauí – UESPI como parte dos requisitos para conclusão do Curso de Bacharelado em Ciência da Computação.

gov.br Documento assinado digitalmente
THIAGO CARVALHO DE SOUSA
Data: 09/07/2025 15:02:23-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Thiago Carvalho de Sousa, Dsc.
Orientador

Nota da Banca Examinadora: 8

Banca Examinadora:

gov.br Documento assinado digitalmente
THIAGO CARVALHO DE SOUSA
Data: 09/07/2025 15:05:40-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Thiago Carvalho de Sousa, Dsc.
Presidente

gov.br Documento assinado digitalmente
MAURICIO REGO MOTA DA ROCHA
Data: 09/07/2025 16:56:11-0300
Verifique em https://validar.iti.gov.br

Maurício Rêgo da Mota Rocha, Dsc.
Membro

gov.br Documento assinado digitalmente
LILIAM BARROSO LEAL
Data: 10/07/2025 09:29:24-0300
Verifique em https://validar.iti.gov.br

Liliam Barroso Leal, Dsc.
Membro

“Ciência da Computação está tão relacionada aos computadores quanto a Astronomia aos telescópios, Biologia aos microscópios, ou Química aos tubos de ensaio. A Ciência não estuda ferramentas. Ela estuda como nós as utilizamos, e o que descobrimos com elas.”

Edsger Dijkstra

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a Deus, pois Ele me proporcionou viver cada momento até aqui, debaixo do Seu sustento.

A minha família, em especial aos meus pais Elisângela e David, meu padrasto Edinardo, minha irmã Lara e minha avó Maria Raimunda, que proporcionou apoio incondicional e sacrifícios incansáveis para que eu pudesse alcançar essa conquista. Sem o amor e apoio de vocês, esta jornada seria impossível.

Ao meu amado marido Micaías, cujo apoio e incentivo foram fundamentais para a conclusão deste trabalho. Obrigada por sempre estar ao meu lado, me encorajando a acreditar em mim mesma. Sou eternamente grata pela sua dedicação e apoio. Este trabalho não seria possível sem você ao meu lado.

Aos meus amigos de curso em especial Arielle, Hanna e Jelson, que foram mais do que colegas, foram companheiros de jornada. Juntos, enfrentamos desafios, celebramos conquistas e criamos memórias que levarei para sempre em meu coração.

Por fim, meu profundo agradecimento ao meu orientador, Dr. Thiago Carvalho de Sousa. Sua sabedoria, orientação e apoio foram fundamentais para concluir este trabalho e para meu crescimento pessoal e acadêmico.

Este trabalho não apenas representa o fim de uma jornada, mas também o início de um novo capítulo em minha vida.

“Porque dEle, e por Ele, e para Ele, são todas as coisas;
glória, a Ele eternamente. Amém.”
(Romanos, 11:36).

RESUMO

A documentação de *Software* é uma atividade essencial na engenharia de *Software*, desempenhando um papel crucial na comunicação, manutenção, escalabilidade e na experiência do usuário, sendo um pilar fundamental para o sucesso de qualquer projeto de desenvolvimento de *Software*. No entanto, esta atividade de documentação é muitas vezes subestimada. Sistemas legados desempenham um papel crítico nas operações de muitas empresas, mas frequentemente enfrentam desafios significativos relacionados à manutenção e à documentação. Isso resulta em custos financeiros substanciais para as organizações e demanda tempo para os programadores que trabalham com esses sistemas. A automação advinda da inteligência artificial oferece uma alternativa para lidar com esses desafios, otimizando a documentação de sistemas legados, reduzindo custos e melhorando a eficiência na manutenção desses sistemas. Identificar a ferramenta de inteligência artificial mais eficaz para essa finalidade é essencial para a adoção de estratégias mais precisas e eficientes no contexto profissional, promovendo melhorias na manutenção dos sistemas e prolongando sua vida útil. Neste trabalho foi realizada uma análise comparativa das ferramentas de geração de documentação de código, com foco em sua aplicação em sistemas legados. A comparação foi realizada a partir da inserção de trechos de código-fonte selecionados nas ferramentas, com o objetivo de avaliar a documentação gerada, a partir de critérios de legibilidade, dimensão e adequação de comentários. Os resultados indicaram que a ferramenta *Gemini* apresentou o melhor desempenho na atividade de geração de documentação; no entanto, as demais ferramentas, *ChatGPT*, *DeepSeek*, *Claude*, *Meta AI*, *Le Chat Mistral* e *Grok*, também forneceram resultados relevantes, evidenciando potencial para aprimoramento tanto nas próprias ferramentas quanto no processo de avaliação adotado. A automação da documentação por meio da inteligência artificial não só melhora a compreensão do código, mas também facilita sua manutenção, resultando em maior eficiência e redução de custos para as organizações.

Palavras-chaves: Engenharia de *Software*. Sistemas Legados. Inteligência Artificial. Documentação de *Software*.

ABSTRACT

Software documentation is an essential activity in software engineering, playing a crucial role in communication, maintenance, scalability, and user experience, and serving as a fundamental pillar for the success of any software development project. However, this documentation activity is often underestimated. Legacy systems play a critical role in the operations of many companies but frequently face significant challenges related to maintenance and documentation. This results in substantial financial costs for organizations and demands considerable time from programmers working with these systems. Automation powered by artificial intelligence offers an alternative to address these challenges by optimizing the documentation of legacy systems, reducing costs, and improving maintenance efficiency. Identifying the most effective AI tool for this purpose is essential for the adoption of more precise and efficient strategies in professional contexts, promoting improvements in system maintenance and extending their lifespan. In this study, a comparative analysis was conducted on code documentation generation tools, focusing on their application to legacy systems. The comparison was based on the insertion of selected source code excerpts into the tools, with the objective of evaluating the generated documentation according to criteria such as readability, length, and adequacy of comments. The results indicated that the Gemini tool demonstrated the best performance in the documentation generation task; however, other tools ChatGPT, DeepSeek, Claude, Meta AI, Le Chat Mistral, and Grok also provided relevant results, highlighting the potential for improvements both in the tools themselves and in the evaluation process used. Automating documentation through artificial intelligence not only enhances code comprehension but also facilitates its maintenance, resulting in greater efficiency and cost reduction for organizations.

Key-words: Software Engineering. Legacy Systems. Artificial Intelligence. Software Documentation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LLMs	*Large Language Models*
APR	*Automatic Program Repair*
NMT	*Neural Machine Translation*
QLoRA	*Quantized Low-Rank Adaptation*
IA	*Inteligência Artificial*
API's	*Application Programming Interface*
Wiki	*Web Page*
TIOBE	*Programming Community Index*
IDE	*Integrated Development Environment*

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A Engenharia de *Software* é a disciplina que abrange todas as fases da criação de *Software*, desde a concepção até a manutenção (Sommerville, 2011). Com a popularização do Manifesto Ágil, que representa a combinação de uma filosofia orientada à satisfação do cliente com um conjunto de princípios de desenvolvimento, a ênfase na entrega em detrimento da análise e do projeto ganhou destaque. Um dos valores fundamentais expressos é a preferência por ”*Software* operacional acima de documentação completa”(Pressman, 2016).

A documentação desempenha um papel fundamental na compreensão e registro de cada componente do código-fonte, seja uma função, classe, trecho ou módulo. Ela representa um conjunto de manuais, abrangendo aspectos gerais e técnicos do *Software*, com flexibilidade para ser organizada de diversas maneiras e aproveitando uma variedade de ferramentas disponíveis (Coelho, 2009). Embora os métodos ágeis questionem a relevância da documentação, dentre todas as melhores práticas na área de engenharia de *Software*, esta assume uma posição particular (Oliveira, 2005).

Sistemas legados são sistemas antigos que ainda estão em uso dentro de uma organização. Geralmente são aplicações complexas, difíceis de manter e de alta importância e frequentemente responsáveis por processos críticos. Entre os principais problemas ligados a esses sistemas estão a falta de documentação, tecnologia obsoleta sem suporte, custos elevados e dificuldades na manutenção (Fritola; Santander, 2022).

O artificial é o que não é natural, feito para imitar a natureza produzido de forma artística ou industrial. Inteligência, embora não tenha uma definição exata, pode-se dizer que está associada ao entendimento, raciocínio e interpretação. Tem-se como Inteligência Artificial a confecção de máquinas com a capacidade de aprender sendo estas programadas previamente (Damaceno; Vasconcelos et al., 2018). Neste sentido, a Inteligência Artificial sistematiza e automatiza tarefas intelectuais e, portanto, é potencialmente relevante para qualquer esfera da atividade intelectual humana (Silva, 2005).

Dentro dessa perspectiva, a questão de manutenção e compreensão de código de sistemas legados, especialmente em contextos onde a documentação original é escassa ou inexistente, este trabalho propõe uma comparação entre ferramentas de Inteligência Artificial voltadas à geração automática de documentação de código-fonte. Busca-se identificar qual ferramenta apresenta melhor desempenho na tarefa de gerar comentários proveitosos, de modo a facilitar o entendimento do código e, consequen-

temente, melhorar a eficiência da manutenção e reduzir os custos associados.

1.1 Justificativa

O avanço da tecnologia traz consigo desafios significativos para as organizações que dependem de sistemas legados. Estes sistemas, embora desempenhem um papel crucial, muitas vezes enfrentam obstáculos como a falta de documentação atualizada e custos elevados de manutenção (Crotty; Horrocks, 2017).

Além disso, devido a restrições de prazos e orçamentos mais limitados, a documentação e o processo de modelagem de *Software* frequentemente são negligenciados, o que pode resultar em consequências que, embora não imediatamente perceptíveis, podem se tornar agravantes a médio e longo prazo (Rocha et al., 2008).

Nesse cenário, a ascensão da inteligência artificial com suas capacidades notáveis, incluindo a automatização de tarefas (Silva, 2005), destaca-se como uma solução para otimizar esses processos.

Assim, surge a importância de investigar o uso de ferramentas de inteligência artificial voltadas à documentação de código-fonte, com um foco em sistemas legados, bem como identificar quais ferramentas apresentam melhores resultados na documentação gerada. Este trabalho busca contribuir para a facilitação da documentação de sistemas legados por meio do uso de inteligência artificial, visando a redução de custos e do tempo necessário para a manutenção desses sistemas.

1.2 Objetivos

1.2.1 Geral

Comparar ferramentas de inteligência artificial que podem ser utilizadas para aprimorar a documentação do código-fonte de sistemas legados.

1.2.2 Específico

- Mapear o estado da arte da pesquisa em geradores de documentação baseados em IA, identificando as tecnologias e tendências predominantes nesse campo.
- Estabelecer critérios claros para a métrica de comparação, permitindo uma avaliação precisa e significativa das ferramentas selecionadas.
- Identificar a melhor ferramenta geradora de documentação de sistemas legados.

1.3 Metodologia

A metodologia seguida para o desenvolvimento desse projeto foi dividido nas seguintes etapas:

Etapa 1 Busca Sistemática e Seleção de Ferramentas: Nesta etapa, foi conduzida uma revisão bibliográfica abrangente para entender as ferramentas utilizadas. Foram identificadas e avaliadas as ferramentas disponíveis que são relevantes.

Etapa 2 Definição de Métricas de Comparação: Nesta etapa, métricas de comparação foram definidas para avaliar as documentações dos códigos-fonte coletados. A justificativa para a escolha de cada métrica foi apresentada.

Etapa 3 Coleta de Código-Fonte de Sistemas Legados no *GitHub*[1]: A terceira etapa consistiu na coleta de código-fonte de sistemas legados de código aberto disponíveis no *GitHub*.

Etapa 4 Comparação da Documentação de Códigos: Após a seleção das ferramentas e a coleta dos códigos-fonte, a análise comparativa foi conduzida de acordo com as métricas definidas na Etapa 3.

Etapa 5 Análise e Apresentação de Resultados: Na quinta etapa, os resultados das comparações foram analisados. A verificação dos resultados considerando as métricas definidas, destacou as melhorias identificadas nas documentações dos códigos-fonte analisados.

1.4 Organização do Trabalho

A estrutura deste trabalho está organizada da seguinte forma: o Capítulo 2 trata da fundamentação teórica, abordando conceitos relacionados à documentação de *Software*, sistemas legados e inteligência artificial. No Capítulo 3, são discutidas as ferramentas selecionadas, as métricas adotadas e os sistemas legados analisados, compondo a base metodológica do estudo. O Capítulo 4 descreve o processo de aplicação das ferramentas sobre os códigos-fonte. O Capítulo 5 apresenta os resultados obtidos a partir da análise dos comentários gerados e da aplicação das métricas. Por fim, o Capítulo 6 reúne as conclusões do trabalho, destacando os principais achados e possibilidades para estudos futuros.

[1] Plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o *Software* Git

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Documentação de Software

A documentação de *Software* consiste em um conjunto de manuais gerais e técnicos (Coelho, 2009). Além disso, pode ser definida como um artefato cuja finalidade seja comunicar a informação sobre o sistema de *Software* ao qual ele pertence (Souza et al., 2015).

A documentação e a modelagem de *Software* desempenham um papel central em todas as etapas que conduzem à implementação bem-sucedida de um *Software* de qualidade. Assim como o projeto arquitetônico e de engenharia orienta a construção de um edifício, a chave para o sucesso de um *Software* está profundamente ligada à sua arquitetura, ao processo de desenvolvimento e às ferramentas utilizadas (Booch, 2006).

2.1.1 Tipos de Documentação

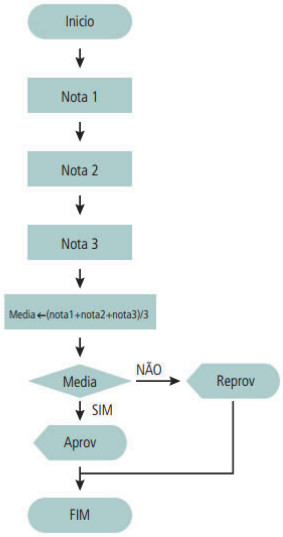
A documentação de *Software* pode ser dividida em dois grandes grupos. A parte técnica é voltada ao desenvolvedor, compreende dicionários e modelos de dados, dicionários de funções, comentários de código (Figura 1) e fluxogramas de processos e regras de negócios (Figura 2). Já a documentação de uso é voltada para o usuário final (Figura 3) e o administrador do sistema, formada por apostilas ou manuais que apresentam como deve ser usado, o que esperar dele e como receber as informações que deseja (Michelazzo, 2008).

Figura 1 – Exemplo de Documentação em Código-Fonte

```
// Verifica se o funcionario tem direito a todos os beneficios
if ((employee.flags & HOURLY_FLAG) &&
    (employee.age > 65))
```

Fonte: (Martin, 2019)

Figura 2 – Exemplo de Fluxograma



Fonte: (Batista, 2016)

Figura 3 – Exemplo de Documentação para Usuário Final

Fonte: (Qualitor, 2016)

2.1.2 Qualidade de Documentação

Qualidade é algo que sentimos e julgamos na vida diária, mas que não conseguimos medir exatamente. Envolve diferentes pontos de vista sobre algo e as características que associamos a esse algo (Wingkvist et al., 2010).

Crosby (1979) define qualidade como ”conformidade com os requisitos”, implicando que os requisitos devem ser claros e compreensíveis para evitar mal-entendidos.

McCall et al. (1977) propõem um modelo de qualidade que identifica onze fatores relacionados aos estágios de revisão, operação e transição do ciclo de vida do *Software*. Eles também desenvolvem cerca de 22 métricas para avaliar esses fatores, muitas das quais são ponderadas e aplicadas para determinar a qualidade de cada um.

Essas métricas frequentemente se baseiam em listas de verificação e são avaliadas em uma escala de 0 a 10, refletindo uma medição subjetiva. Esse modelo de qualidade é adotado como padrão na norma ISO/IEC 9126-1:2001.

McConnell (2004) distingue entre qualidade interna e externa, referindo-se à qualidade durante a produção do produto e à qualidade quando o produto está em uso, respectivamente. Esses conceitos também são abordados na norma ISO/IEC 9126. Juran et al. (1999) oferece uma perspectiva adicional sobre qualidade, definindo-a como ”aptidão para uso”. Essa abordagem leva em consideração as necessidades e expectativas dos clientes em relação ao produto e como eles o utilizam. Juran et al. (1999) também destaca que, como diferentes clientes podem usar o produto de maneiras diversas, é importante que o produto apresente várias características que atendam a esses usos variados.

A documentação de *Software* de alta qualidade desempenha um papel crucial no desenvolvimento, compreensão e manutenção do *Software*. No entanto, devido à sua natureza de linguagem natural informal, que é essencialmente ambígua, imprecisa, desestruturada e complexa tanto em sintaxe quanto em semântica, a avaliação de sua qualidade frequentemente requer métodos manuais. Atualmente, há uma lacuna significativa na disponibilidade de um quadro abrangente e ferramentas para avaliar a qualidade da documentação de *Software* (Treude; Middleton; Atapattu, 2020).

2.2 Sistemas Legados

A definição de Sistemas legados abrange sistemas de informações mais antigos que continuam a operar em uma organização. São caracterizados por sua resistência à modificação, importância crítica para os negócios e potencial impacto negativo em caso de falha. Além disso, sistemas legados podem ser baseados em tecnologias desatualizadas e dados obsoletos, porém, ainda desempenham um papel fundamental nas operações cotidianas. Em especial, nos setores financeiros, os gastos com a manutenção desses sistemas representam uma parcela significativa dos investimentos em Tecnologia da Informação (Crotty; Horrocks, 2017). Em suma, esses sistemas possuem valor para a organização e ainda resistem às modificações e à evolução, principalmente por causa de uma documentação falha (Espindola; Majdenbaum; Audy, 2004).

A manutenção é uma das atividades mais duradouras no ciclo de vida de um *Software*, e isto se deve à observação das leis de evolução. Entre essas leis, destacam-se: a) A mudança contínua: os sistemas devem ser adaptados senão eles se tornarão insatisfatórios; b) O aumento da complexidade: conforme o sistema evolui, a sua complexidade aumenta; c) O crescimento contínuo: o conteúdo funcional deve crescer para

manter a satisfação do cliente. Já no contexto de sistemas legados, alguns fatores como: a) A dificuldade de entender o programa de “outra pessoa”; b) A má documentação e algoritmos mal escritos; c) Documentação inexistente ou não compreensível e consistente com o código fonte, encarecem e dificultam a manutenção do *Software* (Ramos; Oliveira; Anquetil, 2004).

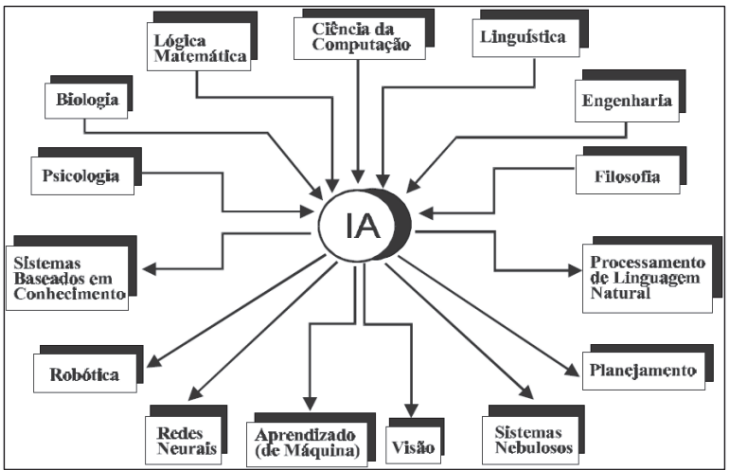
2.3 Inteligência Artificial

A inteligência artificial surgiu após a Segunda Guerra Mundial e assim representa uma ciência recente. Compreende uma ampla série de subcampos, abrangendo desde aplicações de uso geral, como aprendizado de máquina e percepção, até tarefas específicas, como jogos de xadrez. A inteligência artificial tem como objetivo a sistematização e automação de tarefas intelectuais e seu potencial impacto abrange todas as esferas da atividade intelectual humana, tornando-a um campo de estudo de alcance universal (Russel; Norvig, 2004).

Embora seja difícil defini-la, ao longo do tempo, a inteligência artificial seguiu quatro linhas de pensamentos: I. Sistemas que pensam como seres humanos: “O novo e interessante esforço para fazer os computadores pensarem... máquinas com mentes, no sentido total e literal” (Haugeland, 1989); II. Sistemas que atuam como seres humanos: “A arte de criar máquinas que executam funções que exigem inteligência quando executadas por pessoas” (Kurzweil et al., 1990); III. Sistemas que pensam racionalmente: “O estudo das faculdades mentais pelo seu uso de modelos computacionais” (Charniak, 1985); IV. Sistemas que atuam racionalmente: “A Inteligência Computacional é o estudo do projeto de agentes inteligentes” (Poole; Mackworth; Goebel, 1998).

Em termos gerais, as linhas de pensamento I e III estão relacionadas ao processo de pensamento e raciocínio, enquanto as II e IV se concentram no comportamento. Além disso, as abordagens I e II avaliam o sucesso com base na capacidade de se assemelhar ao desempenho humano, enquanto as abordagens III e IV medem o sucesso comparando-o a um conceito ideal de inteligência, conhecido como racionalidade. Um sistema racional é aquele que realiza suas ações corretamente dadas as informações disponíveis (Russel; Norvig, 2004). À medida que a inteligência artificial avançou e demonstrou sua capacidade de automatizar tanto tarefas simples quanto complexas, um vasto conjunto de aplicações surgiu (Figura 4).

Figura 4 – Áreas Relacionadas à Inteligência Artificial



Fonte: (Monard; Baranauskas, 2000)

2.3.1 Modelos de Linguagem de Grande Escala

Modelos de grande linguagem (LLMs) são modelos de inteligência artificial geralmente baseados na arquitetura *Transformer*, projetados para compreender e gerar linguagem humana, códigos, entre outros. Esses modelos são treinados com grandes volumes de dados textuais, o que lhes permite capturar complexidades e nuances da linguagem natural. Os LLMs são capazes de realizar uma ampla variedade de tarefas, desde a classificação de textos até a geração de conteúdo, com elevado grau de precisão, fluência e adequação estilística (Ozdemir, 2023).

2.3.1.1 Aplicações na Engenharia de *Software*

Zhang et al. (2023) apresenta uma revisão abrangente sobre os avanços recentes da comunidade de engenharia de *Software* baseados em LLMs, organizando-os em cinco fases fundamentais do ciclo de vida da engenharia de *Software*: requisitos e design, desenvolvimento, testes, manutenção e gerenciamento.

Na fase de requisitos e design, Xie et al. (2023) realizaram o primeiro estudo empírico avaliando a capacidade dos LLMs em gerar especificações de *Software* a partir de comentários e documentação, utilizando técnicas de *few-shot learning* e estratégias variadas de construção rápida. O estudo também realizou uma comparação entre falhas dos LLMs e métodos tradicionais, identificando forças e limitações de cada abordagem.

No desenvolvimento de *Software*, a pesquisa de Wang et al. (2023) buscou identificar erros semânticos em códigos traduzidos por modelos como Codex e ChatGPT.

Na área de testes, Zhu et al. (2021) propuseram o TroBo, uma técnica de locali-

zação de *bugs* entre projetos baseada no modelo CodeBERT, que utiliza relatórios de erro e código-fonte como entrada.

Já em relação à manutenção, Jiang et al. (2021) propuseram o CURE, uma abordagem de reparo automático de programas (APR) baseada em NMT, com suporte do modelo GPT. O CURE realiza pré-treinamento e ajuste fino em grandes conjuntos de métodos Java, aplica tokenização de subpalavras e uma estratégia avançada de busca de feixe, combinando-se ao CoCoNuT para formar um *pipeline* de geração de *patches* mais eficiente.

Por fim, no contexto de gerenciamento de *Software*, Alhamed et al. (2022) exploraram o uso do modelo BERT com recursos especializados para estimar o esforço necessário em tarefas de manutenção.

3 FERRAMENTAS, MÉTRICAS E SISTEMAS LEGADOS

Este capítulo apresenta explanações relacionadas à busca sistemática de trabalhos relacionados e ferramentas, definição de métricas e coleta de código-fonte de sistemas legados utilizados para o desenvolvimento deste trabalho, descrevendo os procedimentos adotados nestas três etapas da metodologia de pesquisa.

3.1 Busca Sistemática

A busca sistemática é uma etapa fundamental da revisão sistemática, cujo objetivo é identificar e incluir, de forma rigorosa e reprodutível, os estudos relevantes sobre uma determinada questão de pesquisa (Pereira; Galvão, 2014). Dessa forma, buscou-se identificar trabalhos relacionados ao presente estudo, além de analisar e selecionar as ferramentas existentes e relevantes para o seu desenvolvimento.

3.1.1 Trabalhos Relacionados

Neste tópico, são apresentados os principais trabalhos que se relacionam com o tema deste estudo.

3.1.1.1 Metodologia

Neste trabalho foi conduzida uma busca sistemática para identificar os estudos fundamentais sobre as ferramentas de Inteligência Artificial na documentação de código-fonte. Inicialmente, foram identificados 94 resultados no *Google* Acadêmico. Posteriormente, utilizando critérios de seleção e exclusão, foram obtidos 79 estudos. Em seguida, com a leitura dos resumos, foram obtidos 7 estudos. Os acessos foram realizados nos meses fevereiro e março de 2025.

3.1.1.2 Processo de Seleção

1. Título
2. Resumo
3. Leitura Completa

3.1.1.3 Critérios de Seleção

Neste ponto, foram definidos critérios de seleção com o objetivo de filtrar os estudos mais relevantes para a proposta deste trabalho. Foram considerados critérios de inclusão e exclusão.

3.1.1.3.1 Critérios de Inclusão

Foram adotados os seguintes critérios de inclusão:

1. Artigos
2. Trabalhos em Inglês
3. Trabalhos publicados no período de 2020-2025

3.1.1.3.2 Critérios de Exclusão

Foram adotados os seguintes critérios de exclusão:

1. Publicações anteriores aos últimos cinco anos
2. Idioma diferente
3. Trabalhos duplicados
4. Acesso Restrito

3.1.1.4 Estratégias de Busca

A estratégia de busca adotada nesta pesquisa consistiu na elaboração de uma *string* com palavras-chave relacionadas ao tema da pesquisa. Essa *string*, apresentada no Quadro 1, foi aplicada no *Google* Acadêmico com o objetivo de recuperar estudos potencialmente significativos.

Quadro 1: *String* de Busca dos Trabalhos Relacionados

”automatic code documentation generation”AND software engineering AND (Natural language processing OR LLM)

Fonte: (Próprio Autor, 2025)

3.1.1.5 Resultados da Busca

Concluído o processo de aplicação da estratégia de busca e dos critérios de seleção, foram identificados os trabalhos mais importantes para esta pesquisa. O processo resultou em um conjunto de estudos que abordam diferentes perspectivas sobre o tema investigado. Os principais trabalhos selecionados estão organizados no Quadro 2.

Quadro 2: Resultados da Busca

Título	Autor	Ano
Prompt Engineering or Fine-Tuning: An Empirical Assessment of LLMs for Code	Jiho Shin, Clark Tang, Tahmineh Mohati, Maleknaz Nayebi, Song Wang, Hadi Hemmati	2025
Can Large Language Models Serve as Evaluators for Code Summarization?	Yang Wu, Yao Wan, Zhaoyang Chu, Wenting Zhao, Ye Liu, Hongyu Zhang, Xuanhua Shi	2024
Prompting and Fine-tuning Large Language Models for Automated Code Review Comment Generation	Md. Asif Haider, Ayesha Binte Mostofa, Sk. Sabit Bin Mosaddek, Anindya Iqbal, Toufique Ahmed	2024
COMCAT: Leveraging Human Judgment to Improve Automatic Documentation and Summarization	Skyler Grandel, Scott Thomas Andersen, Yu Huang, and Kevin Leac	2024
Leveraging LLMs for Legacy Code Modernization: Challenges and Opportunities for LLM-Generated Documentation	Colin Diggs, Michael Doyle, Amit Madan, Siggy Scott, Emily Escamilla, Jacob Zimmer, Naveed Nekoo, Paul Ursino, Michael Bartholf, Zachary Robin, Anand Patel, Chris Glasz, William Macke, Paul Kirk, Jasper Phillips, Arun Sridharan, Doug Wendt, Scott Rosen, Nitin Naik, Justin F. Brunelle, Samruddhi Thaker	2024
LLMofBabel: An analysis of the behavior of large language models when performing Java code summarization in Dutch	Gopal-Raj G. S. Panch	2024
MMultilingual Code Co-evolution using Large Language Models	Jiyang Zhang, Pengyu Nie, Junyi Jessy Li, Milos Gligoric	2023

Fonte: (Próprio Autor, 2025)

Diferente do artigo *Prompt Engineering or Fine-Tuning: An Empirical Assessment of LLMs for Code*, que avalia o desempenho do GPT-4 e de modelos ajustados em tarefas como sumarização, geração e tradução de código, a presente pesquisa ressalta a geração automática de comentários para trechos de código-fonte legado em diferentes linguagens de programação. Em vez de comparar estratégias de engenharia de prompt ou *fine-tuning*, este trabalho compara ferramentas de geração de texto quanto à qualidade dos comentários produzidos.

Enquanto o artigo *Can Large Language Models Serve as Evaluators for Code Summarization?* investiga o uso de LLMs como avaliadores da qualidade de resumos de código por meio do método CODERPE, o presente trabalho tem como objetivo a comparação entre diferentes ferramentas de geração automática de comentários em código-fonte legados de várias linguagens de programação. O artigo analisa a atuação dos LLMs em papéis como revisor e analista, avaliando critérios como coerência e relevância. Já este trabalho avalia diretamente os comentários gerados por ferramentas.

Enquanto o artigo *Prompting and Fine-tuning Large Language Models for Automated Code Review Comment Generation* busca melhorar a geração de comentários por meio do fine-tuning de um LLM com QLoRA, focando na automação da revisão de código, o presente trabalho tem uma abordagem comparativa: avalia comentários gerados por diferentes ferramentas de geração de texto aplicadas a código-fonte legado em várias linguagens sem ajustar modelos, mas analisando a qualidade das saídas produzidas.

Este trabalho difere do artigo *COMCAT: Leveraging Human Judgment to Improve Automatic Documentation and Summarization* por adotar uma visão de comparação entre ferramentas de geração automática de comentários aplicadas a código-fonte legados em diversas linguagens, enquanto o artigo propõe uma única abordagem voltada para C/C++, com comentários gerados com base em contexto guiado por especialistas. Além disso, o presente trabalho avalia a qualidade dos comentários gerados, enquanto o artigo desenvolve um pipeline específico para identificar onde e que tipo de comentário inserir.

Esta pesquisa e o artigo *Leveraging LLMs for Legacy Code Modernization* investigam o uso de LLMs para gerar comentários em código legado, mas sob perspectivas distintas. Enquanto o artigo propõe uma abordagem linha a linha com uma rubrica própria de avaliação e analisa correlações com métricas automatizadas, o presente trabalho compara diferentes ferramentas de geração de texto aplicadas a códigos legados em várias linguagens, avaliando a qualidade dos comentários gerados de forma comparativa.

Diferente do artigo *LLMofBabel: An analysis of the behavior of large language models when performing Java code summarization in Dutch*, que analisa os erros de

um único modelo (CodeQwen 1.5-7) ao gerar comentários em holandês para código Java. O presente trabalho compara diferentes ferramentas de geração de texto aplicadas a trechos de código legado em várias linguagens de programação. Enquanto o artigo aborda a categorização de erros linguísticos e semânticos em um idioma específico, este trabalho avalia a qualidade dos comentários gerados por diferentes ferramentas.

O artigo *Multilingual Code Co-evolution using Large Language Models* propõe o uso de LLMs para traduzir alterações de código entre diferentes linguagens, mantendo múltiplas versões de um software sincronizadas. Este trabalho utiliza LLMs para gerar e comparar comentários automáticos em códigos legados de diferentes linguagens de programação, com foco na documentação e compreensão do código-fonte de sistemas legados. Ambos exploram LLMs em contextos multilíngues, mas com objetivos distintos: tradução funcional no artigo e documentação automática no referido trabalho.

3.1.2 Seleção das Ferramentas

Nesta etapa, foi realizado o processo de seleção das ferramentas com o objetivo de escolher aquelas que eram relevantes para o momento da pesquisa. Para isto, foram aplicados critérios de exclusão.

3.1.2.1 Critérios de Exclusão

Foram adotados os seguintes critérios de exclusão:

1. Ferramenta paga.
2. Ferramenta que não gera texto.

3.1.3 Estratégias de Busca

A estratégia de busca adotada para este estudo envolveu a elaboração de uma *string* composta por palavras-chave relacionadas aos exemplos das ferramentas mais utilizadas. Essa *string*, apresentada no Quadro 3, foi aplicada na barra de pesquisa do *Google* para a recuperação de resultados relevantes.

Quadro 3: *String* de Busca das Ferramentas

“Best AI chatbot tools OR ”example AI chatbot tools” OR ”popular chatbot tools” OR ”most used AI chatbots””

Fonte: (Próprio Autor, 2025)

3.1.3.1 Resultado da Busca

A aplicação da *string* de busca no mecanismo de pesquisa *Google* (acessado em 6 de abril de 2025) resultou na exibição da página principal do site *Zapier*, que apresenta uma lista atualizada dos principais *chatbots* de inteligência artificial do ano de 2025, posicionando-se logo após os resultados gerados por IA e anúncios patrocinados. A relação das principais ferramentas identificadas encontra-se no Quadro 4.

Com base na lista obtida apresentada no Quadro 4, a seleção das ferramentas selecionadas, organizadas no Quadro 5, foi realizada considerando a distinção entre os modelos utilizados pelas ferramentas, conforme descrito no próprio site consultado.

Quadro 4: Principais *chatbots* de 2025

Nome Ferramenta	Modelo
ChatGPT	*OpenAI GPT models, o1 and o3 models, and DALL-E 3*
DeepSeek	*DeepSeek V3 and R1*
Claude	*Claude Haiku, Sonnet, and Opus series*
Meta AI	*Llama series*
Google Gemini	*Gemini and Imagen series*
Microsoft Copilot	*OpenAI's models*
Zapier Agents	*OpenAI's models*
Poe	*Includes OpenAI's models, StableDiffusionXL, Claude, Gemini, Llama, Mistral, and more*
Perplexity	*OpenAI, Claude, and DeepSeek models*
Le Chat Mistral	*Mistral model series*
Zapier Chatbots	*OpenAI's models*
HuggingChat	*Lots of open models*
Grok	*Grok series*
Pi	*Inflection-2.5*
you.com	*OpenAI models, Claude, Llama, DBRX, and others*
Personal AI	*Proprietary GGT-P, OpenAI's models, Claude*
Merlin AI	*OpenAI models, Claude, Gemini, Mistral*
ZenoChat	*Sophos 2, OpenAI's models, Claude, Llama*
Khan Academy's Khanmigo	*Proprietary*

Fonte: Rebelo (2025)

Quadro 5: Ferramentas Escolhidas

Nome Ferramenta	Versão Disponível na Data do Trabalho
ChatGPT	*GPT-4o mini*
DeepSeek	*DeepSeek-V3*
Claude	*Claude Sonnet 4*
Meta AI	*Llama-4*
Google Gemini	*2.5 flash*
Le Chat Mistral	*Pixtral Large*
Grok	*Grok 3*

Fonte: (Próprio Autor, 2025)

3.2 Métricas de Comparação

Arthur e Stevens (1989) identificaram quatro indicadores essenciais de Qualidade de Documento (IQD) para uma documentação adequada: Precisão, Completude, Usabilidade e Expansibilidade.

Plosch et al. (2014) e Wingkvist et al. (2010) propuseram os seguintes atributos adicionais de qualidade da documentação: Precisão, Clareza, Consistência, Legibilidade, Estruturação e Compreensibilidade.

As métricas propostas por Aversano et al. (2017) destacam-se por sua abrangência e objetividade. Elas contemplam diferentes tipos de documentação, como documentos técnicos, APIs, páginas Wiki e comentários no código-fonte, permitindo uma análise mais completa da qualidade documental de um sistema. Além disso, são definidas de forma clara e mensurável, com fórmulas e critérios bem estabelecidos, o que possibilita uma avaliação sistemática e reduz a subjetividade no processo de análise. Por essas razões, adotou-se utilizar neste trabalho as métricas citadas como base para sua análise, visando garantir uma abordagem sólida, comparável e fundamentada.

O Quadro 6 apresenta métricas para indicadores de qualidade. Entre elas, destaca-se a Q3 - Legibilidade, que verifica se as sentenças expressam conceitos de maneira clara e compreensível. A métrica Q4 - Dimensão avalia se as sentenças são excessivamente longas ou curtas, o que pode comprometer a clareza ou a completude da informação. Por fim, a métrica de Q9 - Adequação dos Comentários analisa a densidade de comentários presentes no código-fonte. Uma alta densidade tende a favorecer a compreensão por parte dos desenvolvedores, facilitando a manutenção e evolução do sistema. As métricas escolhidas para este estudo foram analisadas quanto à sua compatibilidade com a documentação do tipo comentários, que é o foco deste trabalho.

Quadro 6: Métricas de Qualidade da Documentação

QUESTION		METRIC			
Id	**Name**	**Id**	**Formulas**	**Values metrics**	**Acceptance Threshold**
Q1	What is the completeness level of documentation with reference to the source code?	M1.1	$Completeness = \frac{\text{Total classes and Packages described in } I}{\text{Total Classes and Packages in source Code}}$ With I ε {API, Wiki, Comments, Documentation}	[0..1]	0.6
Q2	Is the available documentation updated with the considered release?	M2.1	Updating, with I ε {API, Wiki, Comments, Documentation}	{yes, not}	Yes
Q3	What is the level of readability of the documentation?	M3.1	$FleschReadabilityIndex$, with I ε {API, Wiki, Comments, Documentation}	[0..100]	50.0
Q4	What is the medium sentences dimension?	M4.1	$MediumSentenceLength$, with I ε {API, Wiki, Comments, Documentation}	Natural number	[15..20]
Q5	If present, the Figures and the Tables are appropriately indexed? Have they a Legend?	M5.1.E	$Number_{D,E}$, with E ε {Figure, Table}	Natural number	
		M5.2.E	$Indexing_{D,E} = \frac{\text{\#indexed figures}}{\text{total figures}}$, with E ε {Figure, Table}	[0..1]	0.6
		M5.3.E	$Legend_{D,E} = \frac{\text{NumberCaptions}_{I,E}}{\text{Total}_{I,E}}$, with E ε {Figure, Table}	[0..1]	0.6
Q6	Is the documentation organized in accordance to the standards?	M6.1	$Consistency_{StD} = \frac{\#documents}{\#documents\ of\ standards}$	[0..1]	0.6
Q7	Is the documentation well structured?	M7.1	$DocumentSize_D$ = Number pages of Doc_1	Natural number	
		M7.2	$AverageSize_{CP_D} = \frac{\#chapters}{\#pages}$	[0..1]	0.6
		M7.3	$ChapterTreeDepth_D$	Natural number	3
		M7.4.E	$Density_{D,E} = \frac{Number_{D,E}}{TotalPages_{D,E}}$, with E ε {Figure, Table}	[0..1]	0.6
Q8	Is the documentation easy to use?	M8.1	$InfoFragmentation_w = \frac{NumberInfo_w}{NumberPages_w}$	[0..1]	0.6
		M8.2	$JavaDocDensity = \frac{\#classes}{JavaDoc\ MB}$	{0..1}	0.8
		M8.3	$WikiSize = \#Wiki\ pages$	Natural number	
		M8.4	$WikiPagesTreeDept$	Natural number	
Q9	Is the code appropriately commented?	M9.1	$CommentDensity = \frac{\#CommentLines}{Lines\ of\ Code}$	[0..1]	0.15

Fonte: (Aversano; Guardabascio; Tortorella, 2017)

3.3 Coleta de Código-Fonte

Nesta etapa, foi realizada a coleta de trechos de código-fonte. Essa coleta serviu para embasar a escolha das linguagens de programação a serem utilizadas no

trabalho, para a aplicação e comparação de ferramentas de geração automática de documentação, conforme os objetivos propostos.

3.3.1 Escolha das Linguagens de Programação

Durante esta fase, foram definidas as linguagens de programação mais adequadas para o desenvolvimento deste trabalho, considerando a relevância para sistemas legados.

3.3.1.1 Estratégia de Busca

A estratégia de busca adotada neste estudo consistiu na elaboração de uma *string*, exposta no Quadro 7, com o objetivo de identificar as linguagens de programação mais populares. Essa *string* foi utilizada na barra de pesquisa do *Google* para recuperar resultados significativos.

Quadro 7: *String* de Busca das Linguagens de Programação

”most popular programming languages”

Fonte: (Próprio Autor, 2025)

3.3.1.2 Resultado da Busca

A aplicação da *string* de busca no *Google* (acessado em 10 de maio de 2025) resultou no acesso à página principal do site *TIOBE*, exibida logo após os resultados gerados por inteligência artificial e os anúncios patrocinados. Nessa página, encontra-se disponível o *ranking* de popularidade das linguagens de programação referente ao ano de 2025, apresentado no Quadro 8.

A escolha das linguagens para este trabalho, exibida no Quadro 9, se baseia no fato de que todas são consideradas linguagens de software legado. Segundo Sneed (2006), à medida que as mudanças nas tecnologias de software ocorrem com maior rapidez, cresce também a proporção de software considerado legado. Agora, essa expressão não se limita mais às linguagens das décadas de 1970, como Fortran, COBOL, PLI e C, mas inclui também linguagens de Quarta Geração dos anos 1980 e linguagens orientadas a objetos da década de 1990. Inclusive, os primeiros programas desenvolvidos em Java são considerados como legados.

Quadro 8: *Ranking* de Popularidade *TIOBE* 2025

Posição	Linguagem
1	*Python*
2	*C++*
3	*C*
4	*Java*
5	*C#*
6	*JavaScript*
7	*Go*
8	*Visual Basic*
9	*Delphi/Object Pascal*
10	*SQL*
11	*Fortran*
12	*R*
13	*Ada*
14	*Scratch*
15	*PHP*
16	*Perl*
17	*MATLAB*
18	*Assembly Language*
19	*Rust*
20	*COBOL*

Fonte: Jansen (2025)

Quadro 9: Linguagens de Programação Escolhidas

COBOL
Fortran
C
Java

Fonte: (Próprio Autor, 2025)

3.3.2 Escolha da Plataforma *GitHub*

O *GitHub* foi escolhido como fonte de dados para este trabalho por ser uma plataforma de projetos de código aberto ampla e popular, que hospeda diversos projetos de código aberto em diferentes domínios. (Han et al., 2019). A plataforma se torna ainda mais popular, não apenas como um serviço de hospedagem de projetos de software, mas também como alvo de pesquisas em engenharia de software. (Hu et al., 2016)

3.3.3 Busca do Código-Fonte das Linguagens Escolhidas

Para a seleção dos repositórios, foram realizadas buscas individuais para cada linguagem de programação na plataforma *GitHub*. As *strings* de busca seguiram o padrão *language*:<Nome Linguagem> *stars*:>N, onde ”N”representa o número mínimo de estrelas, utilizado como critério para filtrar projetos com maior relevância.As expressões de busca específicas para cada linguagem estão detalhadas nos Quadros 10, 11, 12 e 13.

Quadro 10: *String* de Busca do Código-Fonte Para COBOL

language:COBOL *stars*:>10

Fonte: (Próprio Autor, 2025)

Quadro 11: *String* de Busca do Código-Fonte Para FORTRAN

language:FORTRAN *stars*:>100

Fonte: (Próprio Autor, 2025)

Quadro 12: *String* de Busca do Código-Fonte Para C

language:C *stars*:>100

Fonte: (Próprio Autor, 2025)

Quadro 13: *String* de Busca do Código-Fonte Para JAVA

language:JAVA *stars*:>100

Fonte: (Próprio Autor, 2025)

Além disso, os repositórios foram ordenados pela data de atualização mais antiga, limitando-se a publicações anteriores ao ano de 2015. A exceção no caso de *COBOL* se deve à escassez de repositórios com mais de 100 estrelas nas condições estabelecidas. Por esse motivo, foi necessário reduzir o limite para 10 estrelas para garantir a obtenção de resultados.

O repositório *COBOL* escolhido contém programas didáticos escritos em *COBOL* por estudantes, entre 2003 e 2004. Ele reúne exemplos clássicos usados no ensino da linguagem *COBOL*, focando em lógica de programação e manipulação de arquivos (Eric, 2015). O código-fonte utilizado neste experimento pode ser consultado no Anexo A.

O repositório *Fortran* escolhido disponibiliza o código-fonte do *NASTRAN-93*, uma versão do *software* de análise estrutural por elementos finitos desenvolvido pela

NASA. Escrito principalmente em *Fortran*, o sistema é usado para simulações estruturais complexas em engenharia aeroespacial (Nasa, 2015). Para mais detalhes, o código-fonte utilizado encontra-se no Anexo B.

O repositório *C* escolhido contém o código-fonte do *Unix* Versão 6, lançado em 1975 pelos laboratórios *Bell*. O material é uma reprodução do sistema operacional que influenciou significativamente o desenvolvimento de *kernels* modernos (Krishnamurthy, 2009). O Anexo C apresenta o código-fonte empregado neste procedimento.

O repositório *Java* escolhido abriga o *plugin* nbgit, criado para integrar o sistema de controle de versão Git ao *NetBeans*, utilizando a biblioteca JGit. Foi desenvolvido antes do suporte oficial ao *Git* no IDE, oferecendo funcionalidades como *commit, log e diff* diretamente no ambiente (Brandon, 2010). O código-fonte usado está incluído no Anexo D.

Estes foram os repositórios selecionados. A escolha do código-fonte considerou as limitações de tamanho dos *prompts* aceitos pelas ferramentas utilizadas, o que dificultou a análise de projetos muito extensos. Por esse motivo, optou-se por selecionar apenas um projeto por linguagem, de forma a garantir uma análise viável dentro das restrições.

4 ANÁLISE DA GERAÇÃO DE CÓDIGO

Neste capítulo, descreve-se todo o processo de geração de documentação utilizando as ferramentas selecionadas, aplicado ao arquivo de código-fonte escolhido para a análise.

4.1 Definição do *Prompt* de Entrada

Para a definição do *prompt*, optou-se por utilizar a formulação do Quadro 14.

Quadro 14: *Prompt* de Entrada

”Documente em português brasileiro todo o código-fonte abaixo, adicionando comentários no próprio código-fonte.”

Fonte: (Próprio Autor, 2025)

Essa escolha tende a garantir que a documentação gerada seja da forma planejada e esteja de acordo com o idioma desejado para a saída.

4.2 Preparação do Código-Fonte

A partir do código-fonte selecionado, foi essencial remover todos os comentários previamente existentes, a fim de evitar qualquer interferência nos resultados. Além disso, os espaços em branco entre as linhas também foram eliminados, com o objetivo de garantir que todo o conteúdo pudesse ser inserido no *prompt* de entrada da ferramenta.

A inserção do código na ferramenta foi realizada por meio do recurso de copiar e colar, em razão das limitações das ferramentas utilizadas quanto à forma de envio de dados para o *prompt*. O script contendo o resultado completo de cada ferramenta se encontra em repositório do *GitHub*[1].

[1] https://github.com/adabeatriz/resultadoTCC

4.3 Aplicação da Métrica na Documentação Gerada

Todas as três métricas escolhidas foram aplicadas aos comentários gerados pelas ferramentas expostas no Quadro 5. Foram selecionados apenas comentários que possuíam texto. A análise considerou a versão mais recente disponível, no momento do estudo, de cada uma das ferramentas. A comparação das ferramentas foi realizada com base nas métricas de Legibilidade, Dimensão e Adequação dos Comentários, conforme detalhado a seguir.

A métrica Q3, correspondente ao teste de legibilidade *Flesch Reading Ease*, foi utilizada para mensurar o grau de legibilidade dos comentários presentes no código. Para isso, foi adotada a fórmula adaptada para a língua portuguesar (Figura 5).

Figura 5 – Fórmula da Legibilidade de *Flesch Reading Ease*

$$226 - 1{,}04 \times \left(\frac{\text{Qnt. de palavras}}{\text{Qnt. de frases}}\right) - 72 \times \left(\frac{\text{Qnt. de sílabas}}{\text{Qnt. de palavras}}\right)$$

Fonte: (Souza Gleice C. de L. Moreno, 2021)

A métrica Q4, referente ao tamanho médio das sentenças, foi obtida por meio da divisão do número total de palavras pelo número total de sentenças nos trechos de comentários extraídos do código, baseada nas orientações de Moreno (2021).

Por fim, a métrica Q9, que avalia a adequação de comentários no código, foi calculada a partir da razão entre o número de linhas de comentários e o número total de linhas de código.

4.4 Resultados

Nesse tópico, descreve-se a análise dos resultados obtidos a partir das métricas.

4.4.1 Análise da Métrica Legibilidade

A métrica de legibilidade Q3, avalia o quão compreensível é o comentário gerado, atribuindo uma pontuação entre 0 e 100. Valores iguais ou superiores a 50 são considerados adequados, indicando maior facilidade de leitura, enquanto pontuações abaixo desse limite sugerem maior dificuldade de interpretação.

Dentre as ferramentas avaliadas, a *Gemini* apresentou o melhor desempenho geral, sendo a única a atingir pontuações superiores a 50 em todas as linguagens

analisadas: *COBOL*, *FORTRAN*, C e JAVA. O destaque foi o resultado em *FORTRAN*, no qual a ferramenta obteve 78,2 pontos, a maior pontuação entre todos os modelos avaliados.

Na linguagem *COBOL*, todas as ferramentas, com exceção de *Claude*, que alcançou 48,1 pontos, superaram o limite mínimo. A *Gemini* obteve 65,0 pontos, enquanto a *GPT* atingiu 62,5, ambos considerados bons resultados.

Para a linguagem C, os valores foram, em geral, mais baixos. Apenas *Le Chat Mistral* com pontuação de 53,8 e *Claude* com valor de 52,7 superaram o valor mínimo, enquanto *Meta* ficou ligeiramente abaixo, com 48,8 pontos.

Na linguagem *JAVA*, apenas *Meta* com pontuação de 53,5 e *Gemini* obtendo um valor de 52,1 superaram a pontuação de referência. Dessa forma, evidencia-se que a ferramenta *Gemini* manteve uma consistência elevada na legibilidade em todas as linguagens avaliadas.

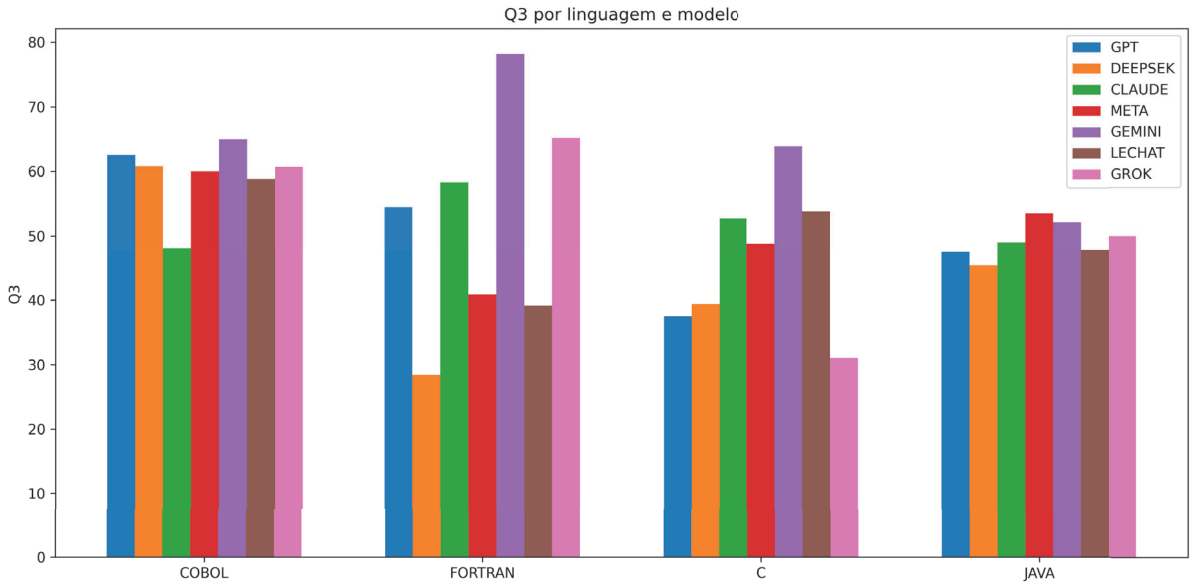
A Tabela 1 apresenta todos os valores obtidos, enquanto a Figura 6 apresenta uma representação gráfica comparativa dos resultados obtidos.

Tabela 1 – Resultado da Legibilidade por Linguagem x Modelo

	COBOL	***FORTRAN***	***C***	***JAVA***
GPT	62.5	54.4	37.5	47.5
DEEPSEEK	60.8	28.4	39.4	45.4
CLAUDE	48.1	58.3	52.7	49.0
META	60.0	40.9	48.8	53.5
GEMINI	65.0	78.2	63.9	52.1
LE CHAT MISTRAL	58.8	39.1	53.8	47.8
GROK	60.7	65.2	31.0	49.9

Fonte: Próprio Autor, 2025

Figura 6 – Gráfico Comparativo de Legibilidade por Linguagem x Modelo



Fonte: Próprio Autor, 2025

4.4.2 Análise da Métrica Dimensão

A métrica de dimensão Q4, avalia o comprimento médio dos comentários gerados por cada ferramenta, considerando como faixa ideal de referência valores entre 15 e 20 palavras, de modo a garantir explicações suficientemente detalhadas sobre o código. Os resultados obtidos, apresentados na Tabela 2, indicam que nenhuma das ferramentas atingiu a faixa considerada ideal, o que sugere que os comentários gerados foram, em geral, curtos demais para proporcionar uma descrição mais completa e precisa das funcionalidades do código.

Apesar disso, a ferramenta *Gemini* apresentou o melhor desempenho relativo, com os maiores comprimentos médios em todas as linguagens analisadas, alcançando 9,9 palavras em *JAVA* e 9,5 em C, demonstrando uma tendência a gerar comentários mais informativos e desenvolvidos, ainda que abaixo do ideal.

As demais ferramentas, como *GPT*, *Claude*, *Meta*, *Le Chat Mistral* e Grok, apresentaram comprimentos médios variando entre 4,1 e 11,8 palavras, o que comprova limitações no detalhamento das explicações geradas. A ferramenta *Claude* em *JAVA*, com 11,8 palavras em média, e *Meta*, com 11,5, se aproximam do desempenho de *Gemini*, mas de forma incoerente entre as demais linguagens.

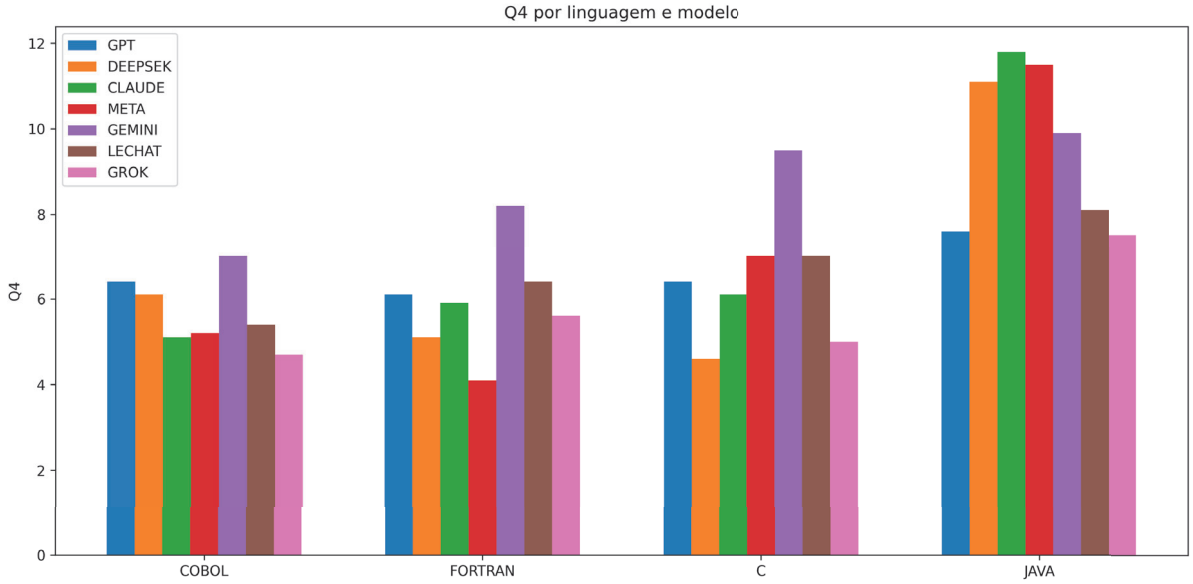
A Figura 7 apresenta uma comparação gráfica entre as ferramentas analisadas.

Tabela 2 – Resultado do Comprimento Médio por Linguagem x Modelo

	COBOL	***FORTRAN***	***C***	***JAVA***
GPT	6.4	6.1	6.4	7.6
DEEPSEK	6.1	5.1	4.6	11.1
CLAUDE	5.1	5.9	6.1	11.8
META	5.2	4.1	7.0	11.5
GEMINI	7.0	8.2	9.5	9.9
LE CHAT MISTRAL	5.4	6.4	7.0	8.1
GROK	4.7	5.6	5.0	7.5

Fonte: Próprio Autor, 2025

Figura 7 – Gráfico Comparativo do Comprimento Médio por Linguagem x Modelo



Fonte: Próprio Autor, 2025

4.4.3 Análise da Métrica Adequação de Comentários

A métrica adequação de comentários Q9, avalia a densidade de comentários no código gerado, com valores ideais entre 0 e 1 e um mínimo recomendado de 0,15.

De modo geral, todas as ferramentas superam esse limite em quase todas as linguagens analisadas. As exceções foram as ferramentas *DeepSeek* e *Meta* em *FORTRAN*, que apresentaram proporções de comentários consideravelmente abaixo do esperado, com 0,086 e 0,057, respectivamente.

A ferramenta *Gemini* se destacou por alcançar os melhores resultados, registrando as maiores proporções de comentários em todas as linguagens testadas, especialmente em *FORTRAN* obtendo um valor de 1,464 e C alcançando um valor de

0,935. Além dela, os modelos *Claude* e *Grok* também apresentaram pontuações altas, especificamente em C e *FORTRAN*, com valores acima de 0,7 nessas linguagens. Esses resultados indicam que *Gemini*, *Claude* e *Grok* geram códigos mais bem documentados, o que pode facilitar a compreensão do código-fonte.

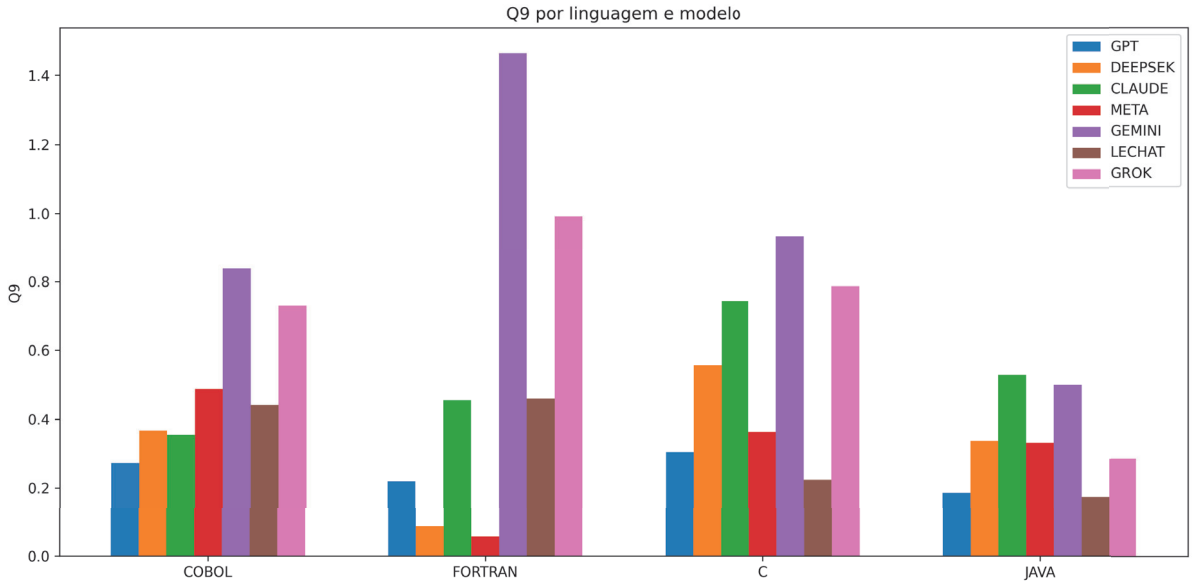
A Tabela 3 concentra os resultados obtidos nas análises, e a Figura 8 apresenta esses mesmos dados em formato gráfico para uma interpretação comparativa.

Tabela 3 – Resultados da Adequação de Comentários por Linguagem x Modelo

	COBOL	***FORTRAN***	***C***	***JAVA***
GPT	0.273	0.220	0.304	0.186
DEEPSEK	0.367	0.086	0.556	0.337
CLAUDE	0.355	0.456	0.743	0.529
META	0.488	0.057	0.363	0.331
GEMINI	0.838	1.464	0.935	0.5
LE CHAT MISTRAL	0.441	0.459	0.224	0.174
GROK	0.729	0.992	0.786	0.284

Fonte: Próprio Autor, 2025

Figura 8 – Gráfico Comparativo de Adequação de Comentários por Linguagem x Modelo



Fonte: Próprio Autor, 2025

5 CONCLUSÃO

Este trabalho teve como objetivo comparar ferramentas de inteligência artificial que podem ser utilizadas para aprimorar a documentação do código-fonte de sistemas legados, com foco nas linguagens *COBOL*, *FORTRAN*, *C* e *JAVA*. A pesquisa partiu da constatação de que a ausência de documentação adequada em sistemas legados representa um obstáculo significativo à sua manutenção e evolução, especialmente em ambientes corporativos onde essas linguagens ainda são amplamente utilizadas.

A metodologia adotada consistiu na utilização de sete ferramentas de geração automática de texto: *GPT*, *Deepseek*, *Claude*, *Meta*, *Gemini*, *Le Chat Mistral* e *Grok*. As ferramentas foram avaliadas por meio de três métricas principais: legibilidade dos comentários gerados, comprimento médio dos textos e proporção de comentários no código.

Em geral, a ferramenta *Gemini* apresentou o melhor desempenho total, destacando-se positivamente nas três métricas avaliadas. Foi a única capaz de atingir níveis desejáveis de legibilidade em todas as linguagens, produzir textos relativamente mais longos mesmo que abaixo do aceitável e manter a maior proporção de comentários no código gerado. As demais ferramentas apresentaram desempenhos mais irregulares, com declínio significativo em uma ou mais métricas dependendo da linguagem avaliada, principalmente referente à legibilidade e comprimento dos textos. Essa análise evidencia o potencial da ferramenta *Gemini* como uma ótima solução para a geração de documentação legível, informativa e bem comentada para códigos-fonte legados.

Durante o processo de avaliação, foram identificadas limitações importantes relacionadas à entrada do *prompt*, tanto em relação à quantidade de *caracteres* permitidos quanto à forma de envio dos dados. Além disso, embora as métricas adotadas neste estudo sejam relevantes, há espaço para aprimoramento em investigações posteriores com o intuito de possibilitar uma análise mais precisa.

Assim, para trabalhos futuros, propõe-se a ampliação do termo utilizado no *prompt*, permitindo um aproveitamento mais aprofundado de diferentes maneiras de entrada. Além disso, a análise de uma maior variedade de linguagens de programação se faz importante para avaliar o comportamento das ferramentas em diversos ambientes de desenvolvimento. Também se destaca a possibilidade de geração de documentação em outros idiomas, o que amplia o alcance e a aplicabilidade dos resultados. Por fim, a utilização de diferentes estratégias de envio de entrada pode melhorar a interação com as ferramentas. Essas linhas de pesquisa não apenas enriquecem a compreensão das capacidades das ferramentas, mas também contribuirão para um

campo de estudo mais diversificado e aplicado.

REFERÊNCIAS

ALHAMED, M.; STORER, T. Evaluation of context-aware language models and experts for effort estimation of software maintenance issues. In: IEEE. *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. [S.l.], 2022. p. 129–138. Citado na página 24.

ARTHUR, J. D.; STEVENS, K. T. Assessing the adequacy of documentation through document quality indicators. In: IEEE. *Proceedings. Conference on Software Maintenance-1989*. [S.l.], 1989. p. 40–49. Citado na página 31.

AVERSANO, L.; GUARDABASCIO, D.; TORTORELLA, M. Evaluating the quality of the documentation of open source software. In: SCITEPRESS. *International Conference on Evaluation of Novel Approaches to Software Engineering*. [S.l.], 2017. v. 2, p. 308–313. Citado 2 vezes nas páginas 31 e 32.

BATISTA, R. d. S. *Lógica de Programação*. [S.l.: s.n.], 2016. Citado na página 20.

BOOCH, G. *UML: guia do usuário*. [S.l.]: Elsevier Brasil, 2006. Citado na página 19.

BRANDON, A. *nbgit*. 2010. Disponível em: <https://github.com/myabc/nbgit>. Citado na página 36.

CHARNIAK, E. *Introduction to artificial intelligence*. [S.l.]: Pearson Education India, 1985. Citado na página 22.

COELHO, H. S. Documentação de software: uma necessidade. *Texto Livre: linguagem e tecnologia*, Universidade Federal de Minas Gerais, v. 2, n. 1, p. 17–21, 2009. Citado 2 vezes nas páginas 16 e 19.

CROSBY, P. B. Quality is free: The art of making quality certain. 1979. Citado na página 20.

CROTTY, J.; HORROCKS, I. Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company. *Applied computing and informatics*, Elsevier, v. 13, n. 2, p. 175–183, 2017. Citado 2 vezes nas páginas 17 e 21.

DAMACENO, S. S.; VASCONCELOS, R. O. et al. Inteligência artificial: uma breve abordagem sobre seu conceito real e o conhecimento popular. *Caderno de Graduação-Ciências Exatas e Tecnológicas-UNIT-SERGIPE*, v. 5, n. 1, p. 11–11, 2018. Citado na página 16.

ERIC. *COBOL*. 2015. Disponível em: <https://github.com/EstesE/COBOL>. Citado na página 35.

ESPINDOLA, R. S. de; MAJDENBAUM, A.; AUDY, J. L. N. Uma análise crítica dos desafios para engenharia de requisitos em manutenção de software. In: *WER*. [S.l.: s.n.], 2004. p. 226–238. Citado na página 21.

FRITOLA, R. G.; SANTANDER, V. F. Documentação de requisitos de sistemas legados uma proposta baseada na engenharia de requisitos orientada a objetivos. In: *WER 2022*. [S.l.: s.n.], 2022. Citado na página 16.

HAN, J. et al. Characterization and prediction of popular projects on github. In: IEEE. *2019 IEEE 43rd annual computer software and applications conference (COMPSAC)*. [S.l.], 2019. v. 1, p. 21–26. Citado na página 34.

HAUGELAND, J. *Artificial intelligence: The very idea*. [S.l.]: MIT press, 1989. Citado na página 22.

HU, Y. et al. Influence analysis of github repositories. *SpringerPlus*, Springer, v. 5, p. 1–19, 2016. Citado na página 34.

JANSEN, P. *TIOBE Index for May 2025*. 2025. <https://www.tiobe.com/tiobe-index/>. Acessado em: 10 Maio 2025. Citado na página 34.

JIANG, N.; LUTELLIER, T.; TAN, L. Cure: Code-aware neural machine translation for automatic program repair. In: IEEE. *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. [S.l.], 2021. p. 1161–1173. Citado na página 24.

JURAN, J. M.; GODFREY, A. B. *The quality control process*. [S.l.]: McGraw-Hill, 1999. Citado na página 21.

KRISHNAMURTHY, R. *UnixV6*. 2009. Disponível em: <https://github.com/Rajmohan/UnixV6>. Citado na página 36.

KURZWEIL, R. et al. *The age of intelligent machines*. [S.l.]: MIT press Cambridge, 1990. v. 580. Citado na página 22.

MARTIN, R. C. *Código limpo: habilidades práticas do Agile software*. [S.l.]: Alta Books Grupo Editorial, 2019. Citado na página 19.

MCCALL, J. A.; RICHARDS, P. K.; WALTERS, G. F. Factors in software quality. volume i. concepts and definitions of software quality. *General Electric CO Sunnyvale CA*, 1977. Citado na página 20.

MCCONNELL, S. *Code complete*. [S.l.]: Pearson Education, 2004. Citado na página 21.

MICHELAZZO, P. *Documentação de software, 2006*. 2008. Citado na página 19.

MONARD, M. C.; BARANAUSKAS, J. A. Aplicações de inteligência artificial: uma visão geral. *Anais*, 2000. Citado na página 23.

NASA. *NASTRAN-93*. 2015. Disponível em: <https://github.com/nasa/NASTRAN-93>. Citado na página 36.

OLIVEIRA, K. M. de. A study of the documentation essential to software maintenance. 2005. Citado na página 16.

OZDEMIR, S. *Quick Start Guide to Large Language Models: Strategies and Best Practices for Using ChatGPT and Other LLMs*. [S.l.]: Addison-Wesley Professional, 2023. Citado na página 23.

PEREIRA, M. G.; GALVÃO, T. F. Etapas de busca e seleção de artigos em revisões sistemáticas da literatura. *Epidemiologia e Serviços de Saúde*, SciELO Brasil, v. 23, n. 2, p. 369–371, 2014. Citado na página 25.

PLÖSCH, R.; DAUTOVIC, A.; SAFT, M. The value of software documentation quality. In: IEEE. *2014 14th International Conference on Quality Software*. [S.l.], 2014. p. 333–342. Citado na página 31.

POOLE, D.; MACKWORTH, A.; GOEBEL, R. Computational intelligence and knowledge. *Computational intelligence: a logical approach*, Oxford University Press New York, NY, USA, v. 1, n. 1, p. 1–22, 1998. Citado na página 22.

PRESSMAN, B. M. R. *Engenharia de Software*. [S.l.]: AMGH Editora LTDA, 2016. Citado na página 16.

QUALITOR. *Manual do USUÁRIO FINAL*. 2016. Disponível em: <https://www.qualitor.com.br/docs/8.10.04.20170127/novainterusufinal/index.html?uf_interface_portal.htm>. Citado na página 20.

RAMOS, C. S.; OLIVEIRA, K. M. de; ANQUETIL, N. Conhecendo sistemas legados através de métricas de software. In: SBC. *Anais do III Simpósio Brasileiro de Qualidade de Software*. [S.l.], 2004. p. 261–275. Citado na página 22.

REBELO, M. *The best AI chatbots in 2025*. 2025. <https://zapier.com/blog/best-ai-chatbot/>. Acessado em: 6 Abril 2025. Citado na página 30.

ROCHA, N. A. d. S. et al. Documentação de software: integração de ferramentas de modelação e processamento de texto. 2008. Citado na página 17.

RUSSEL, S.; NORVIG, P. Inteligência artificial. 2ª. *Edição. São Paulo: Campus*, 2004. Citado na página 22.

SILVA, R. A. C. Inteligência artificial aplicada a ambientes de engenharia de software: Uma visão geral. *INFOCOMP Journal of Computer Science*, v. 4, n. 4, p. 27–37, 2005. Citado 2 vezes nas páginas 16 e 17.

SNEED, H. M. Integrating legacy software into a service oriented architecture. In: IEEE. *Conference on Software Maintenance and Reengineering (CSMR'06)*. [S.l.], 2006. p. 11–pp. Citado na página 33.

SOMMERVILLE, I. *Engenharia de Software*. [S.l.]: Pearson Education do Brasil, 2011. Citado na página 16.

SOUZA GLEICE C. DE L. MORENO, N. H. A. K. Marco P. M. de. *ALT - Análise de Legibilidade Textual*. 2021. <https://legibilidade.com/>. Acessado em: 16 Maio 2025. Citado na página 38.

SOUZA, S. C. B. de et al. Investigação da documentação de maior importância para manutenção de software. 2015. Citado na página 19.

TREUDE, C.; MIDDLETON, J.; ATAPATTU, T. Beyond accuracy: Assessing software documentation quality. In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. [S.l.: s.n.], 2020. p. 1509–1512. Citado na página 21.

WANG, B. et al. Transmap: Pinpointing mistakes in neural code translation. In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. [S.l.: s.n.], 2023. p. 999–1011. Citado na página 23.

WINGKVIST, A. et al. A metrics-based approach to technical documentation quality. In: IEEE. *2010 Seventh International Conference on the Quality of Information and Communications Technology*. [S.l.], 2010. p. 476–481. Citado 2 vezes nas páginas 20 e 31.

XIE, D. et al. Impact of large language models on generating software specifications. *arXiv preprint arXiv:2306.03324*, 2023. Citado na página 23.

ZHANG, Q. et al. A survey on large language models for software engineering. *arXiv preprint arXiv:2312.15223*, 2023. Citado na página 23.

ZHU, Z.; WANG, Y.; LI, Y. Trobo: A novel deep transfer model for enhancing cross-project bug localization. In: SPRINGER. *International Conference on Knowledge Science, Engineering and Management*. [S.l.], 2021. p. 529–541. Citado na página 23.

ANEXO A – CÓDIGO COBOL

```
IDENTIFICATION DIVISION.
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT PAYROLL-FILE-IN
         ASSIGN TO "TW12C.DAT".
     SELECT INCOME-TAX-REPORT-OUT
         ASSIGN TO "PROG6_OUT.DAT".
 DATA DIVISION.
 FILE SECTION.
 FD  PAYROLL-FILE-IN
         RECORD CONTAINS 80 CHARACTERS
         LABEL RECORDS ARE OMITTED
         DATA RECORD IS PR-PAYROLL-REC-IN.
 01  PR-PAYROLL-REC-IN               PIC X(80).
 FD  INCOME-TAX-REPORT-OUT
         RECORD CONTAINS 132 CHARACTERS
         LABEL RECORDS ARE OMITTED
         DATA RECORD IS IT-REPORT-OUT.
 01  IT-REPORT-OUT                   PIC X(132).
 WORKING-STORAGE SECTION.
 01  WS-SWITCHES.
     05  WS-EOF-SWITCH               PIC X(1).
 01  WS-PAGE-LAYOUT.
     05  WS-LINES-USED               PIC 9(2).
     05  WS-PAGE-COUNT               PIC 9(2).
 01  WS-CALC-WORK-AREA.
     05  WS-EXEMPT-SUB-TOT           PIC 9(5)V99.
     05  WS-ANN-EARN                 PIC 9(6)V99.
     05  WS-EARNINGS                 PIC 9(6)V99.
     05  WS-ANN-TAX-AMT              PIC 9(6)V99.
     05  WS-PER-TAX-AMT              PIC 9(6)V99.
 01  WS-TABLE-CONTROLS.
     05  WS-ROW-FOUND-SWITCH         PIC X(1).
         88  WS-ROW-FOUND            VALUE "Y".
         88  WS-END-OF-TABLE         VALUE "E".
 01  WS-TOTALS.
     05  WS-EARN-THIS-PER-TOT        PIC 9(6)V99.
     05  WS-FED-TAX-AMT-TOT         PIC 9(6)V99.
     05  WS-FED-TAX-THIS-PER-TOT     PIC 9(6)V99.
 01  WS-MARR-TAX-DATA.
     05  FILLER              PIC X(16) VALUE "0000002400000000".
     05  FILLER              PIC X(16) VALUE "0240006600000015".
```

```
44      05  FILLER                 PIC X(16) VALUE "0660010900063018".
45      05  FILLER                 PIC X(16) VALUE "1090015000140421".
46      05  FILLER                 PIC X(16) VALUE "1500019200226524".
47      05  FILLER                 PIC X(16) VALUE "1920023600327328".
48      05  FILLER                 PIC X(16) VALUE "2360028900450532".
49      05  FILLER                 PIC X(16) VALUE "2890099999620137".
50  01  WS-MARR-TAX-TABLE REDEFINES WS-MARR-TAX-DATA.
51      05  WS-MARR-TAX-ROW        OCCURS 8 TIMES
52                                 INDEXED BY M-INDEX.
53          10  WS-MARR-LOW        PIC 9(5).
54          10  WS-MARR-HIGH       PIC 9(5).
55          10  WS-MARR-BASE-AMT   PIC 9(4).
56          10  WS-MARR-PERCENT    PIC V99.
57  01  WS-SING-TAX-DATA.
58      05  FILLER                 PIC X(16) VALUE "0000001420000000".
59      05  FILLER                 PIC X(16) VALUE "0142003300000015".
60      05  FILLER                 PIC X(16) VALUE "0330006800028218".
61      05  FILLER                 PIC X(16) VALUE "0680010200091221".
62      05  FILLER                 PIC X(16) VALUE "1020014200162626".
63      05  FILLER                 PIC X(16) VALUE "1420017200266630".
64      05  FILLER                 PIC X(16) VALUE "1720022500356634".
65      05  FILLER                 PIC X(16) VALUE "2250099999536839".
66  01  WS-SING-TAX-TABLE REDEFINES WS-SING-TAX-DATA.
67      05  WS-SING-TAX-ROW        OCCURS 8 TIMES
68                                 INDEXED BY S-INDEX.
69          10  WS-SING-LOW        PIC 9(5).
70          10  WS-SING-HIGH       PIC 9(5).
71          10  WS-SING-BASE-AMT   PIC 9(4).
72          10  WS-SING-PERCENT    PIC V99.
73  01  WS-PR-PAYROLL-REC-IN.
74      05  WS-PR-REC-CODE-IN          PIC 9(2).
75      05  WS-PR-SSN-IN.
76          10  WS-PR-SSN-3-IN         PIC X(3).
77          10  WS-PR-SSN-2-IN         PIC X(2).
78          10  WS-PR-SSN-4-IN         PIC X(4).
79      05  WS-PR-EMPL-NAME-IN         PIC X(18).
80      05  FILLER                     PIC X(20) VALUE SPACES.
81      05  WS-PR-EARN-THIS-PER-IN     PIC 9(5)V99.
82      05  FILLER                     PIC X(14) VALUE SPACES.
83      05  WS-PR-MARITAL-STATUS-IN    PIC 9(1).
84      05  WS-PR-EXEMPTIONS-IN        PIC 9(1).
85      05  FILLER                     PIC X(8) VALUE SPACES.
86  01  WS-HEADING.
87      05  FILLER                     PIC X(7) VALUE "FEDERAL".
88      05  FILLER                     PIC X(1) VALUE SPACES.
89      05  FILLER                     PIC X(6) VALUE "INCOME".
90      05  FILLER                     PIC X(1) VALUE SPACES.
```

```
91      05  FILLER                      PIC X(3) VALUE "TAX".
92      05  FILLER                      PIC X(1) VALUE SPACES.
93      05  FILLER                      PIC X(8) VALUE "REGISTER".
94      05  FILLER                      PIC X(50) VALUE SPACES.
95      05  FILLER                      PIC X(4) VALUE "PAGE".
96      05  FILLER                      PIC X(1) VALUE SPACES.
97      05  WS-PAGE-OUT                 PIC ZZ9.
98      05  FILLER                      PIC X(47) VALUE SPACES.
99  01  WS-COLUMN-HEADING-01.
100     05  FILLER                      PIC X(6) VALUE "SOCIAL".
101     05  FILLER                      PIC X(1) VALUE SPACES.
102     05  FILLER                      PIC X(4) VALUE "SEC.".
103     05  FILLER                      PIC X(22) VALUE SPACES.
104     05  FILLER                      PIC X(1) VALUE "M".
105     05  FILLER                      PIC X(2) VALUE SPACES.
106     05  FILLER                      PIC X(2) VALUE "WH".
107     05  FILLER                      PIC X(3) VALUE SPACES.
108     05  FILLER                      PIC X(8) VALUE "EARNINGS".
109     05  FILLER                      PIC X(2) VALUE SPACES.
110     05  FILLER                      PIC X(10) VALUE "ANNUALIZED".
111     05  FILLER                      PIC X(4) VALUE SPACES.
112     05  FILLER                      PIC X(8) VALUE "ADJUSTED".
113     05  FILLER                      PIC X(4) VALUE SPACES.
114     05  FILLER                      PIC X(7) VALUE "FEDERAL".
115     05  FILLER                      PIC X(5) VALUE SPACES.
116     05  FILLER                      PIC X(3) VALUE "FED".
117     05  FILLER                      PIC X(1) VALUE SPACES.
118     05  FILLER                      PIC X(3) VALUE "TAX".
119     05  FILLER                      PIC X(36) VALUE SPACES.
120 01  WS-COLUMN-HEADING-02.
121     05  FILLER                      PIC X(2) VALUE SPACES.
122     05  FILLER                      PIC X(6) VALUE "NUMBER".
123     05  FILLER                      PIC X(7) VALUE SPACES.
124     05  FILLER                      PIC X(8) VALUE "EMPLOYEE".
125     05  FILLER                      PIC X(1) VALUE SPACES.
126     05  FILLER                      PIC X(4) VALUE "NAME".
127     05  FILLER                      PIC X(5) VALUE SPACES.
128     05  FILLER                      PIC X(1) VALUE "S".
129     05  FILLER                      PIC X(2) VALUE SPACES.
130     05  FILLER                      PIC X(2) VALUE "EX".
131     05  FILLER                      PIC X(2) VALUE SPACES.
132     05  FILLER                      PIC X(4) VALUE "THIS".
133     05  FILLER                      PIC X(1) VALUE SPACES.
134     05  FILLER                      PIC X(4) VALUE "PER.".
135     05  FILLER                      PIC X(4) VALUE SPACES.
136     05  FILLER                      PIC X(8) VALUE "EARNINGS".
137     05  FILLER                      PIC X(4) VALUE SPACES.
```

```
138      05  FILLER                      PIC X(8) VALUE "EARNINGS".
139      05  FILLER                      PIC X(4) VALUE SPACES.
140      05  FILLER                      PIC X(3) VALUE "TAX".
141      05  FILLER                      PIC X(1) VALUE SPACES.
142      05  FILLER                      PIC X(4) VALUE "AMT.".
143      05  FILLER                      PIC X(2) VALUE SPACES.
144      05  FILLER                      PIC X(4) VALUE "THIS".
145      05  FILLER                      PIC X(1) VALUE SPACES.
146      05  FILLER                      PIC X(6) VALUE "PERIOD".
147      05  FILLER                      PIC X(34) VALUE SPACES.
148  01  WS-DETAIL-LINE.
149      05  WS-SSN-OUT.
150          10  WS-SSN-3-OUT            PIC X(3).
151          10  FILLER                  PIC X(1) VALUE "-".
152          10  WS-SSN-2-OUT            PIC X(2).
153          10  FILLER                  PIC X(1) VALUE "-".
154          10  WS-SSN-4-OUT            PIC X(4).
155      05  FILLER                      PIC X(2) VALUE SPACES.
156      05  WS-EMPL-NAME-OUT            PIC X(18).
157      05  FILLER                      PIC X(2) VALUE SPACES.
158      05  WS-MARITAL-STATUS-OUT       PIC 9.
159      05  FILLER                      PIC X(2) VALUE SPACES.
160      05  WS-WITHHOLD-EXEMPT-OUT      PIC 99.
161      05  FILLER                      PIC X(2) VALUE SPACES.
162      05  WS-EARN-THIS-PER-OUT        PIC ZZ,ZZZ.99.
163      05  FILLER                      PIC X(2) VALUE SPACES.
164      05  WS-ANN-EARN-OUT             PIC ZZZ,ZZZ.99.
165      05  FILLER                      PIC X(2) VALUE SPACES.
166      05  WS-ADJ-EARN-OUT             PIC ZZZ,ZZZ.99.
167      05  FILLER                      PIC X(2) VALUE SPACES.
168      05  WS-FED-TAX-AMT-OUT          PIC ZZ,ZZZ.99.
169      05  FILLER                      PIC X(4) VALUE SPACES.
170      05  WS-FED-TAX-THIS-PER-OUT     PIC Z,ZZZ.99.
171      05  FILLER                      PIC X(36) VALUE SPACES.
172  01  WS-TAX-EXEMPT-LINE.
173      05  WS-TE-SSN-OUT.
174          10  WS-TE-SSN-3-OUT         PIC X(3).
175          10  FILLER                  PIC X(1) VALUE "-".
176          10  WS-TE-SSN-2-OUT         PIC X(2).
177          10  FILLER                  PIC X(1) VALUE "-".
178          10  WS-TE-SSN-4-OUT         PIC X(4).
179      05  FILLER                      PIC X(2) VALUE SPACES.
180      05  WS-TE-EMPL-NAME-OUT         PIC X(18).
181      05  FILLER                      PIC X(2) VALUE SPACES.
182      05  WS-TE-MARITAL-STATUS-OUT    PIC 9.
183      05  FILLER                      PIC X(2) VALUE SPACES.
184      05  WS-TE-WITHHOLD-EXEMPT-OUT   PIC 99.
```

```
185      05  FILLER                      PIC X(2) VALUE SPACES.
186      05  WS-TE-EARN-THIS-PER-OUT     PIC ZZ,ZZZ.99.
187      05  FILLER                      PIC X(3) VALUE SPACES.
188      05  FILLER                      PIC X(3) VALUE "-  ".
189      05  FILLER                      PIC X(7) VALUE "  T A X".
190      05  FILLER                      PIC X(6) VALUE SPACES.
191      05  FILLER                  PIC X(13) VALUE "E X E M P T  ".
192      05  FILLER                      PIC X(3) VALUE "  -".
193  01  WS-TOTAL-LINE.
194      05  FILLER                      PIC X(20) VALUE SPACES.
195      05  FILLER                  PIC X(13) VALUE "T O T A L S :".
196      05  FILLER                      PIC X(6) VALUE  SPACES.
197      05  WS-TOT-EARN-THIS-PER-OUT    PIC ZZZ,ZZZ.99.
198      05  FILLER                      PIC X(25) VALUE SPACES.
199      05  WS-TOT-FED-TAX-AMT-OUT      PIC ZZZ,ZZZ.99.
200      05  FILLER                      PIC X(3) VALUE SPACES.
201      05  WS-TOT-FED-TAX-THIS-PER-OUT PIC ZZ,ZZZ.99.
202      05  FILLER                      PIC X(1) VALUE SPACES.
203      05  FILLER                      PIC X(1) VALUE "*".
204      05  FILLER                      PIC X(34) VALUE SPACES.
205  PROCEDURE DIVISION.
206  A00-MAINLINE-PARA.
207      OPEN INPUT PAYROLL-FILE-IN
208           OUTPUT INCOME-TAX-REPORT-OUT.
209      PERFORM B10-INIT-PARA.
210      READ PAYROLL-FILE-IN INTO WS-PR-PAYROLL-REC-IN
211          AT END MOVE "Y" TO WS-EOF-SWITCH.
212      PERFORM B20-PROCESS-PARA
213          UNTIL WS-EOF-SWITCH = "Y".
214      PERFORM C20-TOTAL-PARA.
215      CLOSE PAYROLL-FILE-IN
216         INCOME-TAX-REPORT-OUT.
217      STOP RUN.
218  B10-INIT-PARA.
219      MOVE ZEROS TO WS-LINES-USED
220                    WS-EARN-THIS-PER-TOT
221                    WS-FED-TAX-AMT-TOT
222                    WS-FED-TAX-THIS-PER-TOT.
223      MOVE 1 TO WS-PAGE-COUNT.
224      PERFORM C10-HEADINGS-PARA.
225  B20-PROCESS-PARA.
226      MULTIPLY WS-PR-EXEMPTIONS-IN BY 1000 GIVING
227          WS-EXEMPT-SUB-TOT ROUNDED.
228      MULTIPLY WS-PR-EARN-THIS-PER-IN BY 26 GIVING
229          WS-ANN-EARN.
230      SUBTRACT WS-EXEMPT-SUB-TOT FROM WS-ANN-EARN GIVING
231          WS-EARNINGS ROUNDED.
```

```
232
233     IF WS-PR-MARITAL-STATUS-IN = 1 THEN
234         PERFORM C30-SINGLE-TAX-PARA
235     ELSE
236         MOVE "E" TO WS-ROW-FOUND-SWITCH
237     END-IF.
238
239     IF WS-PR-MARITAL-STATUS-IN = 2 THEN
240         PERFORM C40-MARRIED-TAX-PARA
241     ELSE
242         MOVE "E" TO WS-ROW-FOUND-SWITCH
243     END-IF.
244
245     IF WS-PR-MARITAL-STATUS-IN = 3 THEN
246         PERFORM C60-TAX-EXEMPT-PARA
247     END-IF.
248
249
250     ADD 2 TO WS-LINES-USED.
251     ADD WS-PR-EARN-THIS-PER-IN TO WS-EARN-THIS-PER-TOT.
252     READ PAYROLL-FILE-IN INTO WS-PR-PAYROLL-REC-IN
253         AT END MOVE "Y" TO WS-EOF-SWITCH.
254 C10-HEADINGS-PARA.
255     MOVE WS-PAGE-COUNT TO WS-PAGE-OUT.
256     WRITE IT-REPORT-OUT FROM WS-HEADING
257         AFTER ADVANCING PAGE.
258     MOVE SPACES TO IT-REPORT-OUT.
259     WRITE IT-REPORT-OUT
260         AFTER ADVANCING 1 LINE.
261     WRITE IT-REPORT-OUT FROM WS-COLUMN-HEADING-01
262         AFTER ADVANCING 1 LINES.
263     WRITE IT-REPORT-OUT FROM WS-COLUMN-HEADING-02
264         AFTER ADVANCING 1 LINE.
265     ADD 3 TO WS-LINES-USED.
266     ADD 1 TO WS-PAGE-COUNT.
267 C20-TOTAL-PARA.
268     IF WS-LINES-USED >= 57 THEN
269         PERFORM C10-HEADINGS-PARA
270         MOVE ZEROS TO WS-LINES-USED
271     END-IF.
272     MOVE WS-EARN-THIS-PER-TOT TO WS-TOT-EARN-THIS-PER-OUT.
273     MOVE WS-FED-TAX-AMT-TOT TO WS-TOT-FED-TAX-AMT-OUT.
274     MOVE WS-FED-TAX-THIS-PER-TOT TO WS-TOT-FED-TAX-THIS-PER-OUT.
275     MOVE SPACES TO IT-REPORT-OUT.
276     WRITE IT-REPORT-OUT
277         AFTER ADVANCING 1 LINE.
278     WRITE IT-REPORT-OUT FROM WS-TOTAL-LINE
```

```
279         AFTER ADVANCING 2 LINE.
280 C30-SINGLE-TAX-PARA.
281     SET S-INDEX TO 1.
282     SEARCH WS-SING-TAX-ROW
283         AT END MOVE "E" TO WS-ROW-FOUND-SWITCH
284         WHEN WS-EARNINGS IS >= WS-SING-LOW  (S-INDEX) AND
285                              IS <= WS-SING-HIGH (S-INDEX)
286         MOVE "Y" TO WS-ROW-FOUND-SWITCH.
287     IF WS-ROW-FOUND THEN
288         COMPUTE WS-ANN-TAX-AMT =
289             WS-SING-BASE-AMT (S-INDEX) +
290             WS-SING-PERCENT (S-INDEX) *
291             (WS-EARNINGS - WS-SING-LOW (S-INDEX)).
292     PERFORM C50-LINE-OUTPUT-PARA.
293 C40-MARRIED-TAX-PARA.
294     SET M-INDEX TO 1.
295     SEARCH WS-MARR-TAX-ROW
296         AT END MOVE "E" TO WS-ROW-FOUND-SWITCH
297         WHEN WS-EARNINGS IS >= WS-MARR-LOW (M-INDEX) AND
298                              IS <= WS-MARR-HIGH (M-INDEX)
299         MOVE "Y" TO WS-ROW-FOUND-SWITCH.
300     IF WS-ROW-FOUND THEN
301         COMPUTE WS-ANN-TAX-AMT =
302             WS-MARR-BASE-AMT (M-INDEX) +
303             WS-MARR-PERCENT (M-INDEX) *
304             (WS-EARNINGS - WS-MARR-LOW (M-INDEX)).
305     PERFORM C50-LINE-OUTPUT-PARA.
306 C50-LINE-OUTPUT-PARA.
307     DIVIDE WS-ANN-TAX-AMT BY 26 GIVING WS-PER-TAX-AMT ROUNDED.
308     ADD WS-ANN-TAX-AMT TO WS-FED-TAX-AMT-TOT.
309     ADD WS-PER-TAX-AMT TO WS-FED-TAX-THIS-PER-TOT.
310     MOVE WS-PR-SSN-3-IN TO WS-SSN-3-OUT.
311     MOVE WS-PR-SSN-2-IN TO WS-SSN-2-OUT.
312     MOVE WS-PR-SSN-4-IN TO WS-SSN-4-OUT.
313     MOVE WS-PR-EMPL-NAME-IN TO WS-EMPL-NAME-OUT.
314     MOVE WS-PR-MARITAL-STATUS-IN TO WS-MARITAL-STATUS-OUT.
315     MOVE WS-PR-EXEMPTIONS-IN TO WS-WITHHOLD-EXEMPT-OUT.
316     MOVE WS-PR-EARN-THIS-PER-IN TO WS-EARN-THIS-PER-OUT.
317     MOVE WS-ANN-EARN TO WS-ANN-EARN-OUT.
318     MOVE WS-EARNINGS TO WS-ADJ-EARN-OUT.
319     MOVE WS-PER-TAX-AMT TO WS-FED-TAX-THIS-PER-OUT.
320     MOVE WS-ANN-TAX-AMT TO WS-FED-TAX-AMT-OUT.
321     IF WS-LINES-USED >= 57 THEN
322         PERFORM C10-HEADINGS-PARA
323         MOVE ZEROS TO WS-LINES-USED
324     END-IF.
325     MOVE SPACES TO IT-REPORT-OUT.
```

```
326         WRITE IT-REPORT-OUT
327             AFTER ADVANCING 1 LINE.
328         WRITE IT-REPORT-OUT FROM WS-DETAIL-LINE
329             AFTER ADVANCING 1 LINE.
330     C60-TAX-EXEMPT-PARA.
331         ADD 2 TO WS-LINES-USED.
332         MOVE WS-PR-SSN-3-IN TO WS-TE-SSN-3-OUT.
333         MOVE WS-PR-SSN-2-IN TO WS-TE-SSN-2-OUT.
334         MOVE WS-PR-SSN-4-IN TO WS-TE-SSN-4-OUT.
335         MOVE WS-PR-EMPL-NAME-IN TO WS-TE-EMPL-NAME-OUT.
336         MOVE WS-PR-MARITAL-STATUS-IN TO WS-TE-MARITAL-STATUS-OUT.
337         MOVE WS-PR-EXEMPTIONS-IN TO WS-TE-WITHHOLD-EXEMPT-OUT.
338         MOVE WS-PR-EARN-THIS-PER-IN TO WS-TE-EARN-THIS-PER-OUT.
339         MOVE "Y" TO WS-ROW-FOUND-SWITCH.
340         IF WS-ROW-FOUND THEN
341             MOVE SPACES TO IT-REPORT-OUT.
342             WRITE IT-REPORT-OUT
343                 AFTER ADVANCING 1 LINE.
344             WRITE IT-REPORT-OUT FROM WS-TAX-EXEMPT-LINE
345                 AFTER ADVANCING 1 LINE.
```

ANEXO B – CÓDIGO FORTRAN

```
      PROGRAM FF
CDC   PROGRAM FF (TAPE3)
      LOGICAL         STAR,     PCT,      NOTYET,   PUNCH,    UPFLAG
      INTEGER         SCREEN,   PROM,     FFFLAG,   FACSF,    NC(7),
     1                IBM,      UNIVAC,   CDC,      VAX,      PC,
     2                UNIX
      CHARACTER*1     FN(1),    BK1,      N1,       Y1,       X1,
     1                S1,       W1,       A1,       H1,       CARD1(5),
     2                FX,       MARKQ,    QMARK,    TMP,      RPRN,
     3                SPL(8),   DOT,      D1,       T1
      CHARACTER*2     CARD2,    REPL
      CHARACTER*3     NEW,      OLD,      ODNW
      CHARACTER*4     CARD(20), SAVE(20), BEGN,     HELP,     STOP,
     1                LIST,     BLANK,    END1,     END2,     END3,
     2                CANC1,    CANC2,    ALTER,    CARD4,    FN4,
     3                BBK(2),   KSMB(9)
      CHARACTER*6     FN6,      BEGN6,    HELP6,    STOP6,    CARD6,
     1                MTYPE(7), CNTLWD(3)
      CHARACTER*8     FNAME(4), MYFILE,   BLNK8,    SITE,     CNTRL,
     1                USE(2),   SYM(4),   TAPE03,   TPF,      FOR003,
     2                DORK,     KEEP,     SPILL
      CHARACTER*11    F
      CHARACTER*32    FN32
      COMMON /SYSTEM/ IBUF,     NOUT,     NOGO,     IN,       ISYS(15)
      COMMON /MACHIN/ MACH
      COMMON /XXREAD/ INFLAG,   INSAVE,   LOOP4,    IBMCDC
      COMMON /XECHOX/ FFFLAG,   IECHO(3), ISORT(5)
      COMMON /XREADX/ SCREEN,   LOOP,     KOUNT,    PROM,     NOTYET,
     1                STAR,     PCT,      ICONT(36)
      COMMON /QMARKQ/ MARKQ,    TMP(16),  SPILL,    SAVE
      COMMON /UPCASX/ UPFLAG,   UPID(3)
      EQUIVALENCE     (FN(1),FN4,FN6,FN32,FNAME(1)),(XXI,LLI),
     1                (CARD(1),CARD6,CARD1(1),CARD2,CARD4),
     2                (SPL(1),SPILL)
      DATA            BEGN,     HELP,     STOP,     BLANK,    BK1    /
     1                'BEGI',   'HELP',   'STOP',   '    ',   ' '    /
      DATA            BEGN6,    HELP6,    STOP6,    BLNK8,    KEEP   /
     1                'BEGIN ', 'HELP  ', 'STOP  ', '    ',   'KEEP' /
      DATA            END1,     END2,     END3,     N1,       ALTER  /
     1                'ENDD',   'END ',   'ENDA',   'N',      'LTER' /
      DATA            CANC1,    CANC2,    DORK,     LLI,      LLJ    /
     1                'CANC',   'EL  ',   'DELETE', 4H  I ,   4H  J  /
      DATA            CNTRL,    MYFILE,   NEW,      OLD,      Y1     /
```

```
     1               'CENTRAL','MIFYLE', 'NEW',    'OLD',    'Y'    /
      DATA           NC/ 12,   7,   28,  7,   28,  0,  28          /
      DATA           UNIVAC,   IBM, CDC, VAX, PC,      UNIX        /
     1               3,        2,   4,   5,   1,       7           /
      DATA           MTYPE                                         /
     1               'IBM PC', '  IBM', 'UNIVAC', '  CDC', '  VAX',
     2               ' ***  ', '  UNIX'                           /
      DATA           TAPE03,   TPF,      FOR003,   F               /
     1               'TAPE3.', 'FF$$.', 'FT03.', 'FF       FF'    /
      DATA           S1,       A1,       BBK,                LIST  /
     1               'S',      'A',      '(BLA', 'NK) ',     'LIST' /
      DATA           W1,       H1,       FX,       RPRN,     ICNTL /
     1               '0',      'H',      'X',      ')',      0     /
      DATA           DOT,      D1,       T1                        /
     1               '.',      'D',      'T'                       /
      DATA           CNTLWD,                         NCNTL,    REPL  /
     1               'CANCEL', 'PROMPT', 'LIST  ', 3,        'R:'  /
      DATA           KSMB  /   '+C0N',   '+C1N',   '+C2N',   '+C3N' ,
     1               '+C4N',   '+C5N',   '+C6N',   '+C7N',   '+C8N' /
      DATA      USE / '@USE 3.,', 'FF$$ .  '/       PUNCH /   .FALSE./
      DATA      SYM / '@SYM    ', 'PUNCH$,,', '****', ' . '          /
      DATA            QMARK / '?' /
      J = LLJ - LLI
      MACH = IBM
      IF (J .EQ.   256) MACH = PC
      IF (J .EQ.   512) MACH = UNIVAC
      IF (J .GT. 65535) MACH = VAX
      IF (J .GT. 2**30) MACH = CDC
      IF (MACH.EQ.VAX .AND. (XXI.GT.1.60E-19 .OR. XXI.LT.1.8E-19))
     1    MACH = UNIX
      IF (MACH.EQ.VAX .AND. (XXI.GT.3.40E-20 .OR. XXI.LT.3.39E-20))
     1    MACH = PC
      LU      = 2
      LOUT    = 3
      IN      = 5
      NOUT    = 6
      IPUN    = 7
      SCREEN  = 6
      IF (MACH .EQ. VAX) SCREEN = 5
      LOOP    =-1
      KOUNT   = 0
      KONTN   = 10000000
      IKI     = 1
      PROM    =+1
      STAR    =.FALSE.
      PCT     =.FALSE.
      NOTYET  =.FALSE.
```

```
      MARKQ   = QMARK
      IECHO(2)=-2
      INSAVE  = IN
      INFLAG  = 0
      FFFLAG  = 0
      UPFLAG  =.FALSE.
      ISYS(15)= 0
      J       = NC(MACH)
      CARD4   = BLANK
      CARD6   = STOP6
      FN4     = STOP
      FN6     = STOP6
      LOOP4   = LOOP - 4
      DO 5 I = 1,20
5     SAVE(I) = BLANK
      IBMCDC = UNIVAC + VAX + PC
      IF (MACH.EQ.IBM .OR. MACH.EQ.CDC) IBMCDC = 0
      IF (MACH .NE. CDC) GO TO 20
      OPEN (UNIT=IN  ,FILE='INPUT' ,STATUS='UNKNOWN')
      OPEN (UNIT=NOUT,FILE='OUTPUT',STATUS='UNKNOWN')
      OPEN (UNIT=IPUN,FILE='PUNCH' ,STATUS='UNKNOWN')
      OPEN (UNIT=LOUT,FILE='TAPE3' ,STATUS='UNKNOWN',ERR=15)
      GO TO 20
15    STOP 'SCRATCH FILE ERROR, UNIT 3'
20    WRITE  (NOUT,25) (F,I=1,7),MTYPE(MACH),(F,I=1,4)
25    FORMAT (/////15X,A11, /14X,A11, 3(/13X,A11), /8X,2(3X,'FFFFFF'),
     1       2(/13X,A11),7X,A6,' VERSION / APRIL 93', /13X,A11,
     2       /12X,A11,10X,'COSMIC, (706) 542-3265', /11X,A11,11X,
     3       'UNIVERSITY OF  GEORGIA', /10X,A11,12X,'ATHENS, GEORGIA',
     4       '  30602')
30    WRITE  (NOUT,35) J
35    FORMAT (//,' *** ENTER A BLANK, ''HELP'', OR A FILE NAME (UP TO'
     1,       I3,' CHARACTERS)', /5X,'IF OUTPUT IS TO BE SAVED')
      READ (IN,40,ERR=330,END=330) FNAME
40    FORMAT (4A8)
      IF (FNAME(1) .EQ. MYFILE) CALL FFHELP (*30,*600,4)
      IF (FNAME(1) .EQ. BLNK8) FNAME(1) = MYFILE
      IF (FN(J+1).NE.BK1 .OR. FN(J+2).NE.BK1 .OR. FN(J+3).NE.BK1)
     1   GO TO 330
      IF (FNAME(2) .NE. BLNK8) GO TO 50
      CALL UPCASE (FNAME,J)
      IF (FN6 .EQ. BEGN6) GO TO 330
      IF (FN6 .EQ. STOP6) GO TO 600
      IF (FN6 .EQ. HELP6) CALL FFHELP (*30,*600,1)
50    ODNW = OLD
      IF (MACH .NE. VAX) GO TO 60
      DO 52 I = 2,J
```

```
138       IF (FN(I) .EQ. DOT) GO TO 60
139       IF (FN(I) .EQ. BK1) GO TO 55
140   52  CONTINUE
141       I = J + 1
142   55  FN(I  ) = DOT
143       FN(I+1) = D1
144       FN(I+2) = A1
145       FN(I+3) = T1
146   60  IF (MACH .NE. IBM) GO TO 65
147       I = IQZDDN(FNAME(1))
148       ODNW = OLD
149       IF (I .EQ. 0) ODNW = NEW
150       IF (ODNW .EQ. NEW) CALL QQDCBF (FNAME(1),0,'F  ',80,80,DA)
151       CALL QQGETF (LU,FNAME(1),IERR)
152       IF (IERR .NE. 0) GO TO 130
153   65  IF (IBMCDC.EQ.0) OPEN (UNIT=LU,FILE=FNAME(1),STATUS=ODNW,ERR=130)
154       IF (IBMCDC.NE.0) OPEN (UNIT=LU,FILE=FN32    ,STATUS=ODNW,ERR=130)
155       IF (ODNW .EQ. NEW) GO TO 140
156   70  WRITE  (NOUT,80)
157   80  FORMAT (/,' FILE ALREADY EXISTS, ENTER ''STOP'', ''OVERWRITE'',',
158      1        ' OR ''APPEND'' -')
159       IF (MACH.EQ.CDC .AND. IN.EQ.5) REWIND IN
160       READ (IN,90,END=70) X1
161   90  FORMAT (A1)
162       CALL UPCASE (X1,1)
163       IF (X1 .EQ. S1) GO TO 480
164       IF (X1 .EQ. W1) GO TO 140
165       IF (X1 .NE. A1) GO TO 70
166       SAVE(2) = BBK(1)
167       SAVE(3) = BBK(2)
168   110 READ (LU,180,END=115) SAVE
169       IF (SAVE(1).EQ.BEGN .AND. SAVE(4).EQ.BLANK) FFFLAG = 1234
170       IF (SAVE(19) .EQ. KSMB(IKI)) IKI = IKI + 1
171       GO TO 110
172   115 BACKSPACE LU
173       IF (FFFLAG .EQ. 1234) WRITE (NOUT,120)
174   120 FORMAT (/,' IF EXISTING FILE CONTAINS FREE-FIELD INPUT CARDS, ',
175      1        ' THIS PROGRAM WILL NOT', /5X,'EXPAND THEM TO FIXED-',
176      2        ' FIELD FORMATS',/)
177       WRITE (NOUT,255) SAVE
178       IF (FFFLAG.EQ.1234 .AND. INFLAG.EQ.0) CALL FFHELP (*125,*125,5)
179   125 CARD(1) = SAVE(1)
180       CARD(2) = SAVE(2)
181       GO TO 140
182   130 IF (ODNW .EQ. NEW) GO TO 310
183       ODNW = NEW
184       GO TO 60
```

```
140  IF (MACH .EQ. UNIVAC) J = FACSF(USE)
     IF (FNAME(1) .EQ. MYFILE) FNAME(1) = BLNK8
     IF (FNAME(1) .EQ.  BLNK8) WRITE (NOUT,150)
150  FORMAT (/5X,'*** OUTPUT NOT SAVED ***',//)
     WRITE  (NOUT,160)
160  FORMAT (//,' *** NASTRAN FREE-FIELD INPUT PROGRAM ***',
    1 /5X,'(THERE WILL BE NO INPUT ECHO UNTIL ''BEGIN BULK'' IS TYPED',
    2 /5X,' TO TERMINATE JOB:  ENTER ''ENDDATA'' OR ''STOP'')',
    3 //5X,'PLEASE BEGIN -',/)
170  CALL FFREAD (*320,CARD)
     IF (CARD4.EQ.CANC1 .AND. CARD(2).EQ.CANC2) GO TO 230
     IF (CARD4.EQ. LIST .AND. CARD(2).EQ.BLANK) GO TO 230
     IF (CARD2.EQ. REPL .AND. CARD(4).EQ.BLANK) GO TO 350
     IF (CARD4.EQ. STOP .AND. CARD(2).EQ.BLANK) GO TO 400
     IF (CARD4.EQ.BLANK .AND. CARD(2).EQ.BLANK) GO TO 370
     IF (CARD4.EQ. HELP .AND. CARD(2).EQ.BLANK)
    1   CALL FFHELP (*280,*400,2)
     IF (LU .EQ. 2) WRITE (LU,180) CARD
180  FORMAT (20A4)
     IF (FFFLAG.NE.1234 .AND. INFLAG.EQ.4) WRITE (NOUT,190) CARD
190  FORMAT (1X,20A4)
     IF (CARD4.EQ.BEGN .AND. CARD(5).EQ.BLANK) GO TO 340
     IF (CARD4.NE.END1 .AND. CARD4.NE.END2 .AND. CARD4.NE.END3)
    1   GO TO 170
     IF (CARD(2).EQ.BLANK .OR. CARD(2).EQ.ALTER) GO TO 170
     GO TO 410
230  IF (LU .NE. 2) GO TO 290
     J = 1
     IF (CARD(5).EQ.CANC1 .AND. ICONT(1).GT.0) J = ICONT(1) + 1
     IF (CARD(5).EQ. LIST .AND. ICONT(1).GT.0) J = ICONT(1)
     DO 240 I = 1,J
240  BACKSPACE LU
     ICONT(1) = 0
     IF (CARD(5) .EQ. LIST) GO TO 260
     READ (LU,180) SAVE
     J = J - 1
     WRITE  (NOUT,250) J
250  FORMAT (1X,I4,' PREVIOUSLY GENERATED CARDS CANCELLED ***')
     IF (J .GT. 0) WRITE (NOUT,255) SAVE
255  FORMAT (/,' *** LAST CARD WAS:', /1X,20A4)
     GO TO 280
260  WRITE  (NOUT,265) J
265  FORMAT (//,' *** PREVIOUS',I4,' CARDS WERE (COLS. 1-79) -',/)
     DO 270 I = 1,J
     READ (LU,180,END=285) SAVE
270  WRITE  (NOUT,275) SAVE
275  FORMAT (1X,20A4)
```

```
280  CARD(1) = SAVE(1)
     CARD(2) = SAVE(2)
     GO TO 170
285  BACKSPACE LU
     SAVE(1) = CARD(1)
     SAVE(2) = CARD(2)
     GO TO 170
290  WRITE  (NOUT,300) CARD4,CARD(2)
300  FORMAT (' *** ',A4,A3,'OPTION NOT ACTIVE.  NO SAVE FILE ',
    1        'REQUESTED')
     GO TO 170
310  WRITE  (NOUT,315) FNAME
315  FORMAT (' *** CAN NOT ASSIGN FILE - ',4A8)
     GO TO 20
320  WRITE  (NOUT,325)
325  FORMAT (' *INPUT ERROR/FF*')
     GO TO 170
330  WRITE  (NOUT,335)
335  FORMAT (' *NOT A VALID FILE NAME*')
     IF (MACH.EQ.CDC .AND. IN.EQ.5) REWIND IN
     GO TO 20
340  FFFLAG = 1234
     IF (INFLAG .EQ. 0) CALL FFHELP (*170,*170,5)
     GO TO 170
350  I = 3
     IF (CARD1(5) .EQ. RPRN) I = 4
     READ (CARD1(I),355,ERR=170) II
355  FORMAT (I1)
     IF (II .EQ. 0) II = 10
     I = I + 2
     DO 360 J = 1,8
     SPL(J) = CARD1(I)
360  I = I + 1
     SAVE(II) = SPILL
     DO 365 I = 1,10
365  CARD(I) = SAVE(I)
     BACKSPACE LU
     WRITE (LU,  180) SAVE
     WRITE (NOUT,190) SAVE
     GO TO 170
370  IF (LU .NE. 2) GO TO 385
     DO 375 J = 3,18
     IF (CARD(J) .NE. BLANK) GO TO 380
375  CONTINUE
     GO TO 170
380  BACKSPACE LU
     READ (LU,180) SAVE
```

```
279       IF (SAVE(19) .EQ. BLANK) GO TO 390
280       WRITE  (NOUT,382)
281  382  FORMAT (/,' BAD INPUT - FIRST FIELD BLANK. TRY AGAIN')
282       WRITE  (NOUT,383)
283  383  FORMAT (13X,'NOT ALLOW. PREVIOUS CARD HAS CONTINUATION FIELD ',
284      1        'DEFINED')
285       GO TO 170
286  385  WRITE  (NOUT,382)
287       WRITE  (NOUT,387)
288  387  FORMAT (13X,'NOT ALLOW WITHOUT SAVE FILE')
289       GO TO 170
290  390  KONTN = KONTN + 1
291       IF (MOD(KONTN,10000) .EQ. 0) IKI = IKI + 1
292       CALL INT2K8 (*385,KONTN,SAVE(19))
293       SAVE(19) = KSMB(IKI)
294       CARD(1 ) = KSMB(IKI)
295       CARD(2 ) = SAVE(20)
296       WRITE  (NOUT,395)
297  395  FORMAT (' ...PREVIOUS CARDS REPLACED BY:')
298       WRITE (NOUT,190) SAVE
299       WRITE (NOUT,190) CARD
300       BACKSPACE LU
301       WRITE (LU,180) SAVE
302       WRITE (LU,180) CARD
303       GO TO 170
304  400  BACKSPACE LOUT
305  410  ENDFILE LOUT
306       IF (LU .NE. 6) ENDFILE LU
307       PUNCH =.FALSE.
308       SITE  = BLNK8
309  420  WRITE  (NOUT,430) QMARK
310  430  FORMAT (/,' *** DO YOU WANT TO PUNCH OUT THE NASTRAN DECK',A1,
311      1        ' (Y,N,X,HELP) ')
312       IF (MACH.EQ.CDC .AND. IN.EQ.5) REWIND IN
313       READ (IN,90,END=420) X1
314       CALL UPCASE (X1,1)
315       IF (X1 .EQ. H1) CALL FFHELP (*420,*480,3)
316       IF (X1 .EQ. N1) GO TO 500
317       IF (X1.NE.Y1 .AND. X1.NE.FX) GO TO 420
318       PUNCH =.TRUE.
319       LX = LOUT
320       IF (X1 .EQ. FX) LX = LU
321       IF (MACH .NE. UNIVAC) GO TO 460
322       IPUN = 1
323       WRITE  (NOUT,440)
324  440  FORMAT (/,' *** ENTER SITE-ID, OR ''CENTRAL'', WHERE CARDS ARE',
325      1        ' TO BE PUNCHED ')
```

```
450  READ (IN,40,ERR=450,END=450) SITE
     IF (SITE .EQ. BLNK8) GO TO 490
     CALL UPCASE (SITE,8)
     IF (SITE .EQ. CNTRL) GO TO 460

     SYM(3) = SITE
     J = FACSF(SYM)
460  REWIND LX
470  READ (LX,180,END=500) CARD
     DO 475 J = 2,NCNTL
     IF (CARD6 .NE. CNTLWD(J)) GO TO 475
     IF (CARD(4) .EQ. BLANK) GO TO 470
475  CONTINUE
     IF ((CARD6.EQ.HELP6 .OR. CARD6.EQ.STOP6) .AND. CARD(3).EQ.BLANK)
    1   GO TO 470
     IF (CARD6.EQ.CNTLWD(1) .AND. CARD(4).EQ.BLANK) ICNTL = ICNTL + 1
     WRITE (IPUN,180) CARD
     GO TO 470
480  FNAME(1) = BLNK8
490  PUNCH    =.FALSE.
500  WRITE  (NOUT,510)
510  FORMAT (//10X,'ADIEU MY FRIEND.  IT IS A PLEASURE TO SERVE YOU')
     IF (FNAME(1) .NE. BLNK8) WRITE (NOUT,520) FNAME
520  FORMAT (10X,'DON''T FORGET - YOUR NASTRAN DECK IS IN FILE -',
    1    /25X,4A8, /10X,'WHICH IS ACCESSIBLE BY THE SYSTEM EDITOR')
     IF (.NOT.PUNCH) GO TO 550
     WRITE  (NOUT,530)
530  FORMAT (/10X,'AND DON''T FORGET TO PICK UP YOUR PUNCHED CARDS')
     IF (SITE .NE. BLNK8) WRITE (NOUT,535) SITE
     IF (SITE.EQ.BLNK8 .AND. MACH.NE.VAX) WRITE (NOUT,540)
     IF (SITE.EQ.BLNK8 .AND. MACH.EQ.VAX) WRITE (NOUT,545)
535  FORMAT (10X,'WHEN YOU SIGN OFF',22X,'SITE-ID: ',A8)
540  FORMAT (10X,'AT THE CENTRAL-SITE')
545  FORMAT (10X,'IN FORTRAN FILE FOR007.DAT')
550  IF (FNAME(1) .EQ. BLNK8) GO TO 570
     IF (MACH .EQ. UNIVAC) FOR003 = TPF
     IF (MACH .EQ.    CDC) FOR003 = TAPE03
     WRITE  (NOUT,555)
555  FORMAT (//10X,'A COPY OF YOUR ACTUAL INPUT CARDS WAS SAVED IN')
     IF (MACH .NE. VAX) WRITE (NOUT,560) FOR003
     IF (MACH .EQ. VAX) WRITE (NOUT,565)
560  FORMAT (1H+,56X,'FORTRAN FILE - ',A8)
565  FORMAT (10X,'FORTRAN FILE FOR003.DAT')
570  IF (ICNTL .NE. 0) WRITE (NOUT,575)
575  FORMAT (/4X,'*** WARNING - CANCELLED CARDS IN PUNCHED DECK NEED ',
    1       'TO BE REMOVED', /19X,'OR MODIFIED BEFORE USE')
     IF (FNAME(1) .NE. BLNK8) DORK = KEEP
```

```
373       CLOSE (UNIT=LU  ,STATUS=DORK)
374       CLOSE (UNIT=LOUT,STATUS=DORK)
375       IF (DORK.EQ.KEEP .OR. (PUNCH .AND. MACH.EQ.VAX)) WRITE (NOUT,585)
376  585  FORMAT (/4X,'*** DON''T FORGET TO DELETE YOUR FILES GENERATED BY',
377      1        ' THIS RUN ***')
378       WRITE  (NOUT,590)
379  590  FORMAT (/26X,'*** JOB DONE ***',/)
380  600  CONTINUE
381       END
```

ANEXO C – CÓDIGO C

```
1
2  #include "../param.h"
3  #include "../user.h"
4  #include "../systm.h"
5  #include "../proc.h"
6  #include "../text.h"
7  #include "../inode.h"
8  #include "../seg.h"
9  #define CLOCK1  0177546
10 #define CLOCK2  0172540
11 int icode[]
12 {
13   0104413,
14   0000014,
15   0000010,
16   0000777,
17   0000014,
18   0000000,
19   0062457,
20   0061564,
21   0064457,
22   0064556,
23   0000164,
24 };
25
26 main()
27 {
28   extern schar;
29   register i, *p;
30
31   updlock = 0;
32   i = *ka6 + USIZE;
33   UISD->r[0] = 077406;
34   for(;;) {
35     UISA->r[0] = i;
36     if(fuibyte(0) < 0)
37       break;
38     clearseg(i);
39     maxmem++;
40     mfree(coremap, 1, i);
41     i++;
42   }
43   if(cputype == 70)
```

```
44    for(i=0; i<62; i=+2) {
45      UBMAP->r[i] = i<<12;
46      UBMAP->r[i+1] = 0;
47    }
48    printf("mem = %l\n", maxmem*5/16);
49    maxmem = min(maxmem, MAXMEM);
50    mfree(swapmap, nswap, swplo);
51
52    UISA->r[7] = ka6[1];
53    UISD->r[7] = 077406;
54    lks = CLOCK1;
55    if(fuiword(lks) == -1) {
56      lks = CLOCK2;
57      if(fuiword(lks) == -1)
58        panic("no clock");
59    }
60    proc[0].p_addr = *ka6;
61    proc[0].p_size = USIZE;
62    proc[0].p_stat = SRUN;
63    proc[0].p_flag =| SLOAD|SSYS;
64    u.u_procp = &proc[0];
65    *lks = 0115;
66    cinit();
67    binit();
68    iinit();
69    rootdir = iget(rootdev, ROOTINO);
70    rootdir->i_flag =& ~ILOCK;
71    u.u_cdir = iget(rootdev, ROOTINO);
72    u.u_cdir->i_flag =& ~ILOCK;
73
74    if(newproc()) {
75      expand(USIZE+1);
76      estabur(0, 1, 0, 0);
77      copyout(icode, 0, sizeof icode);
78      return;
79    }
80    sched();
81  }
82
83  sureg()
84  {
85    register *up, *rp, a;
86
87    a = u.u_procp->p_addr;
88    up = &u.u_uisa[16];
89    rp = &UISA->r[16];
90    if(cputype == 40) {
```

```
91      up =- 8;
92      rp =- 8;
93    }
94    while(rp > &UISA->r[0])
95      *--rp = *--up + a;
96    if((up=u.u_procp->p_textp) != NULL)
97      a =- up->x_caddr;
98    up = &u.u_uisd[16];
99    rp = &UISD->r[16];
100   if(cputype == 40) {
101     up =- 8;
102     rp =- 8;
103   }
104   while(rp > &UISD->r[0]) {
105     *--rp = *--up;
106     if((*rp & WO) == 0)
107       rp[(UISA-UISD)/2] =- a;
108   }
109 }
110
111 estabur(nt, nd, ns, sep)
112 {
113   register a, *ap, *dp;
114
115   if(sep) {
116     if(cputype == 40)
117       goto err;
118     if(nseg(nt) > 8 || nseg(nd)+nseg(ns) > 8)
119       goto err;
120   } else
121     if(nseg(nt)+nseg(nd)+nseg(ns) > 8)
122       goto err;
123   if(nt+nd+ns+USIZE > maxmem)
124     goto err;
125   a = 0;
126   ap = &u.u_uisa[0];
127   dp = &u.u_uisd[0];
128   while(nt >= 128) {
129     *dp++ = (127<<8) | RO;
130     *ap++ = a;
131     a =+ 128;
132     nt =- 128;
133   }
134   if(nt) {
135     *dp++ = ((nt-1)<<8) | RO;
136     *ap++ = a;
137   }
```

```
138   if(sep)
139   while(ap < &u.u_uisa[8]) {
140     *ap++ = 0;
141     *dp++ = 0;
142   }
143   a = USIZE;
144   while(nd >= 128) {
145     *dp++ = (127<<8) | RW;
146     *ap++ = a;
147     a =+ 128;
148     nd =- 128;
149   }
150   if(nd) {
151     *dp++ = ((nd-1)<<8) | RW;
152     *ap++ = a;
153     a =+ nd;
154   }
155   while(ap < &u.u_uisa[8]) {
156     *dp++ = 0;
157     *ap++ = 0;
158   }
159   if(sep)
160   while(ap < &u.u_uisa[16]) {
161     *dp++ = 0;
162     *ap++ = 0;
163   }
164   a =+ ns;
165   while(ns >= 128) {
166     a =- 128;
167     ns =- 128;
168     *--dp = (127<<8) | RW;
169     *--ap = a;
170   }
171   if(ns) {
172     *--dp = ((128-ns)<<8) | RW | ED;
173     *--ap = a-128;
174   }
175   if(!sep) {
176     ap = &u.u_uisa[0];
177     dp = &u.u_uisa[8];
178     while(ap < &u.u_uisa[8])
179       *dp++ = *ap++;
180     ap = &u.u_uisd[0];
181     dp = &u.u_uisd[8];
182     while(ap < &u.u_uisd[8])
183       *dp++ = *ap++;
184   }
```

```
185   sureg();
186   return(0);
187
188 err:
189   u.u_error = ENOMEM;
190   return(-1);
191 }
192
193 nseg(n)
194 {
195
196   return((n+127)>>7);
197 }
```

ANEXO D – CÓDIGO JAVA

```
1  package org.nbgit;
2
3  import java.beans.PropertyChangeListener;
4  import java.beans.PropertyChangeSupport;
5  import java.io.File;
6  import java.io.FileInputStream;
7  import java.io.FileOutputStream;
8  import java.io.IOException;
9  import java.util.HashMap;
10 import java.util.Map;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13 import org.nbgit.util.GitUtils;
14 import org.netbeans.modules.versioning.spi.VersioningSupport;
15 import org.openide.filesystems.FileObject;
16 import org.openide.filesystems.FileUtil;
17 import org.openide.util.RequestProcessor;
18 import org.eclipse.jgit.lib.Constants;
19 import org.eclipse.jgit.lib.Repository;
20
21 public class Git {
22
23   public static final String GIT_OUTPUT_TAB_TITLE = org.openide.util.
     NbBundle.getMessage(Git.class, "CTL_Git_DisplayName");
24   public static final String PROP_ANNOTATIONS_CHANGED = "
     annotationsChanged";
25   public static final String PROP_VERSIONED_FILES_CHANGED = "
     versionedFilesChanged";
26   public static final String PROP_CHANGESET_CHANGED = "changesetChanged";
27   public static final Logger LOG = Logger.getLogger("org.nbgit");
28   private static final int STATUS_DIFFABLE =
29       StatusInfo.STATUS_VERSIONED_UPTODATE |
30       StatusInfo.STATUS_VERSIONED_MODIFIEDLOCALLY |
31       StatusInfo.STATUS_VERSIONED_MODIFIEDINREPOSITORY |
32       StatusInfo.STATUS_VERSIONED_CONFLICT |
33       StatusInfo.STATUS_VERSIONED_MERGE |
34       StatusInfo.STATUS_VERSIONED_REMOVEDINREPOSITORY |
35       StatusInfo.STATUS_VERSIONED_MODIFIEDINREPOSITORY |
36       StatusInfo.STATUS_VERSIONED_MODIFIEDINREPOSITORY;
37   private final PropertyChangeSupport support = new PropertyChangeSupport(
     this);
38   private final StatusCache statusCache = new StatusCache(this);
39   private HashMap<String, RequestProcessor> processorsToUrl;
```

```
40   private final Map<File, Repository> repos = new HashMap<File, Repository
     >();
41   private static Git instance;
42
43   private Git() {
44   }
45
46   public static synchronized Git getInstance() {
47     if (instance == null) {
48       instance = new Git();
49     }
50     return instance;
51   }
52
53   public Repository getRepository(File root) {
54     Repository repo = repos.get(root);
55
56     if (repo == null) {
57       final File gitDir = new File(root, Constants.DOT_GIT);
58       try {
59         repo = new Repository(gitDir);
60         repos.put(root, repo);
61       } catch (IOException ex) {
62       }
63     }
64
65     return repo;
66   }
67
68   public StatusCache getStatusCache() {
69     return statusCache;
70   }
71
72   public boolean isAdministrative(File file) {
73     String name = file.getName();
74     return isAdministrative(name) && file.isDirectory();
75   }
76
77   public boolean isAdministrative(String fileName) {
78     return fileName.equals(".git");
79   }
80
81   public boolean isManaged(File file) {
82     return VersioningSupport.getOwner(file) instanceof GitVCS && !GitUtils
     .isPartOfGitMetadata(file);
83   }
84
```

```
85   public File getTopmostManagedParent(File file) {
86     if (GitUtils.isPartOfGitMetadata(file)) {
87       for (; file != null; file = file.getParentFile()) {
88         if (isAdministrative(file)) {
89           file = file.getParentFile();
90           break;
91         }
92       }
93     }
94     File topmost = null;
95     for (; file != null; file = file.getParentFile()) {
96       if (org.netbeans.modules.versioning.util.Utils.isScanForbidden(file)
     ) {
97         break;
98       }
99       if (new File(file, ".git").canWrite()) {
100        topmost = file;
101        break;
102      }
103    }
104    return topmost;
105  }
106
107  public String getMimeType(File file) {
108    FileObject fo = FileUtil.toFileObject(file);
109    String foMime;
110    if (fo == null) {
111      foMime = "content/unknown";
112    } else {
113      foMime = fo.getMIMEType();
114      if ("content/unknown".equals(foMime))
115      {
116        foMime = "text/plain";
117      }
118    }
119    if ((statusCache.getStatus(file).getStatus() & StatusInfo.
     STATUS_VERSIONED) == 0) {
120      return GitUtils.isFileContentBinary(file) ? "application/octet-
     stream" : foMime;
121    } else {
122      return foMime;
123    }
124  }
125
126  public void versionedFilesChanged() {
127    support.firePropertyChange(PROP_VERSIONED_FILES_CHANGED, null, null);
128  }
```

```
129
130   public void refreshAllAnnotations() {
131     support.firePropertyChange(PROP_ANNOTATIONS_CHANGED, null, null);
132   }
133
134   public void changesetChanged(File repository) {
135     support.firePropertyChange(PROP_CHANGESET_CHANGED, repository, null);
136   }
137
138   public void addPropertyChangeListener(PropertyChangeListener listener) {
139     support.addPropertyChangeListener(listener);
140   }
141
142   public void removePropertyChangeListener(PropertyChangeListener listener
      ) {
143     support.removePropertyChangeListener(listener);
144   }
145
146   public void getOriginalFile(File workingCopy, File originalFile) {
147     StatusInfo info = statusCache.getStatus(workingCopy);
148     LOG.log(Level.FINE, "getOriginalFile: {0} {1}", new Object[]{
     workingCopy, info});
149     if ((info.getStatus() & STATUS_DIFFABLE) == 0) {
150       return;
151     }
152     try {
153       File original = GitUtils.getFileRevision(workingCopy, GitRepository.
     REVISION_BASE);
154       if (original == null) {
155         return;
156       }
157       org.netbeans.modules.versioning.util.Utils.copyStreamsCloseAll(new
     FileOutputStream(originalFile), new FileInputStream(original));
158       original.delete();
159     } catch (IOException e) {
160       Logger.getLogger(Git.class.getName()).log(Level.INFO, "Unable to get
      original file", e);
161     }
162
163   }
164
165   public RequestProcessor getRequestProcessor() {
166     return getRequestProcessor((String) null);
167   }
168
169   public RequestProcessor getRequestProcessor(File file) {
170     return getRequestProcessor(file.getAbsolutePath());
```

```
171   }
172
173   public RequestProcessor getRequestProcessor(String url) {
174     if (processorsToUrl == null) {
175       processorsToUrl = new HashMap<String, RequestProcessor>();
176     }
177     String key;
178     if (url != null) {
179       key = url;
180     } else {
181       key = "ANY_URL";
182     }
183     RequestProcessor rp = processorsToUrl.get(key);
184     if (rp == null) {
185       rp = new RequestProcessor("Git - " + key, 1, true);
186       processorsToUrl.put(key, rp);
187     }
188     return rp;
189   }
190
191   public void clearRequestProcessor(String url) {
192     if (processorsToUrl != null & url != null) {
193       processorsToUrl.remove(url);
194     }
195   }
196 }
```