

Explorando a Evolução da Inteligência Artificial: Ferramentas e Aplicações no Desenvolvimento de Software

Raiany de Souza¹

¹Faculdade de Computação – Universidade Federal do Pará (UFPA) – Campus Castanhal
Castanhal – PA – Brazil

`raiany.souza@castanhal.ufpa.br`

Abstract. *This paper explores the evolution and application of Artificial Intelligence (AI) with a specific focus on software development. Beginning with the central question of autonomous machine use, we delve into the intricacies of software engineering, highlighting AI as a crucial support. We investigate the history of AI, from early attempts to contemporary applications. Within the context of AI, we discuss foundational concepts and diverse application areas, emphasizing their impact on daily life. The article then shifts focus to the convergence of AI with software development, analyzing innovative tools such as TabNine, GitHub Copilot, Amazon CodeWhisperer, Sourcery e Visual Studio IntelliCode.*

Resumo. *Este artigo explora a evolução e aplicação da Inteligência Artificial (IA) com foco no desenvolvimento de software. Iniciando com a questão central da utilização autônoma de máquinas, abordamos a complexidade da engenharia de software, destacando a IA como suporte crucial. Investigamos a história da IA, desde tentativas pioneiras até suas aplicações contemporâneas. No contexto das IA's, discutimos conceitos fundamentais e diversas áreas de aplicação, enfatizando seu impacto na vida cotidiana. O artigo, então, direciona o foco para a convergência da IA com o desenvolvimento de software, analisando ferramentas inovadoras como TabNine, GitHub Copilot, Amazon CodeWhisperer, Sourcery e Visual Studio IntelliCode.*

1. INTRODUÇÃO

A sociedade sempre se questionou sobre como utilizar as máquinas como ferramentas autônomas no cotidiano, uma indagação que continua a nortear a utilização das Inteligências Artificiais (IA) na atualidade. O conjunto de métodos e conhecimentos sobre as máquinas têm evoluído ao longo do tempo, representando um desafio na execução de tarefas simples (Goodfellow, 2016). A engenharia de software, aliada à programação, tem se revelado um desafio considerável para os desenvolvedores ao longo do tempo. A interação com grandes volumes de informações apresenta dificuldades, pois vai além da simples escrita de códigos (Zawinski, 2009). Frederick P. Brooks, Jr. (1975), em "The Mythical Man Month," descreve a engenharia de software como uma etapa gigante e de difícil transcrição para a máquina, desmistificando a ideia de que é um trabalho fácil.

Neste contexto, este artigo não se limita à discussão; concentra-se, especificamente, em aprofundar a compreensão dos mecanismos e metodologias envolvidos no uso das Inteligências Artificiais para o desenvolvimento de software. A complexidade e as dificuldades inerentes à engenharia de software, como obstáculos à criatividade e consolidação de ideias, destacam as IA's como ferramentas essenciais de suporte. Diante dos desafios enfrentados pelos desenvolvedores na criação e execução de softwares, a investigação dos padrões das Inteligências Artificiais torna-se vital para seu eficaz emprego nesse cenário.

Este artigo tem como objetivo explorar e analisar as Inteligências Artificiais (IAs) direcionadas para a programação, investigando como essas tecnologias inovadoras estão transformando o cenário do desenvolvimento de software. Abordaremos avanços recentes e casos de uso notáveis, examinando como as IAs estão sendo aplicadas para otimizar o processo de codificação, identificar bugs, sugerir melhorias de código e até mesmo gerar automaticamente trechos de programa. Além disso, pretendemos avaliar o impacto dessas soluções na eficiência, produtividade e criatividade dos desenvolvedores, contribuindo assim para uma compreensão mais abrangente do papel crescente das IAs no campo da programação.

Para abordar essas questões, a pesquisa envolveu a catalogação de informações relacionadas a diferentes instrumentos de inteligência artificial. Através da execução de códigos, avaliamos a utilidade e benefícios dessas ferramentas em diversas demandas, analisando os resultados e filtrando informações pertinentes. Ao final, apresentaremos de forma prática e compreensível as informações mais relevantes, justificando o estudo e fornecendo insights sobre a aplicação eficaz de cada ferramenta.

2. INTELIGÊNCIA ARTIFICIAL

2.1. História das IA

A Inteligência Artificial é um instrumento inovador na evolução tecnológica para o ser humano, formulando um avanço no que tange a forma de como se executa uma atividade que seria difícil para uma pessoa e fácil dentro de uma arquitetura de códigos pré-estabelecidos na máquina.

Para compreender como ocorreu a evolução da inteligência artificial (IA), é fundamental começar pelo ponto de partida primordial das inteligências artificiais. Isso se deu através de pesquisadores que tentaram simular processos de evolução biológica utilizando computadores para criar máquinas inteligentes. Em seus experimentos, propunha-se a interação entre um humano e uma máquina, buscando determinar quem era quem, com o objetivo principal de as máquinas convencerem de que eram humanos. Isso resultou em uma explosão de pesquisas e discussões na época, com métodos e estudos para mecanizar a inteligência em diversas áreas, como redes neurais e métodos estatísticos, por exemplo (Negnevitsky, 2002).

O matemático e cientista da computação, Alan Turing, notabilizou-se na história da inteligência artificial por sua contribuição seminal na formulação da relação entre a inteligência humana e as máquinas. Em sua obra fundamental "Computing Machinery and Intelligence" (1950, revista *Mind*), Turing propôs o célebre "Teste de Turing" como um método para avaliar se uma máquina pode exibir comportamento inteligente indistinguível do humano. Seu objetivo central era explorar a viabilidade de as máquinas adotarem comportamentos semelhantes aos seres humanos, marcando um marco significativo no desenvolvimento e compreensão da inteligência artificial.

As primeiras tentativas para simular essa evolução em um computador foram realizadas pelo Instituto de Estudos Avançados de Princeton junto do geneticista Nils Aall Barricelli, descrevendo assim em seu artigo no ano de 1954, em que serviu como incentivo para atenção de outros cientistas na área. Nilsson (2009).

R. N. Friedberg e seus colegas da IBM, foram outros pesquisadores que tentaram formular o uso das máquinas como um facilitador de tarefas. No resumo de seu artigo de 1958 Friedberg escreveu que as máquinas seriam mais úteis se aprendessem a executar tarefas a partir de uma experiência com resultados bem sucedidos. No entanto, as tentativas de Friedberg de desenvolver esses programas não foram bem-sucedidas.

Outros trabalhos posteriores sobre a evolução artificial tiveram mais sucesso, como de Lawrence Fogel (1928-2007), podendo fazer previsões em sequência com as máquinas Woodrow W. Bledsoe (1921–1995), e Hans J. Bremermann (1926–1969), que utilizaram a evolução para resolver problemas matemáticos e de otimização. Em

geral, o motor dessa época, era utilizar a máquina como força intelectual na sua evolução em frente a tarefas “fáceis e repetitivas”. Nilsson (2009).

Puxando para a atualidade, fica bem claro o quanto essa evolução buscada pelos pesquisadores se aperfeiçoou. As máquinas além de terem uma agilidade e melhora tecnológica gigantesca, o uso de respostas simples e inteligentes de programas, adicionam fundamentos de utilidade incríveis para o cotidiano das pessoas, sendo obrigatório no dia a dia do ser humano o uso de inteligência artificial para resolver problemas.

2.2. Conceito das IA's

Desde os primórdios, os filósofos se perguntam como funciona a mente humana e se os não-humanos podem ter mentes. Ao longo da engenharia da computação, a criação de máquinas inteligentes foi se aplicando na descoberta e criatividade. A criação de máquinas já foi concebida para ser chamada de “inteligente”, chegando ao ponto de nos questionarmos sobre o significado de “inteligência”. Segundo a definição de um dicionário em inglês, surgem dois conceitos de inteligência, a saber:

- 1 A inteligência de alguém é a sua capacidade de compreender e aprender coisas.
- 2 Inteligência é a capacidade de pensar e compreender em vez de fazer coisas por instinto ou automaticamente.

(Dicionário Essencial de Inglês, Collins, Londres, 1990)

A definição de Inteligência Artificial (IA) tem sido objeto de debate ao longo do tempo, refletindo a evolução tanto do pensamento filosófico quanto dos avanços tecnológicos. Diversas abordagens conceituais emergiram, cada uma proporcionando uma lente única para compreender a natureza e o propósito da IA.

Em visão principal, o cérebro da máquina, tende a formular que a inteligência em geral deve ser usada como tomador de decisões inteligentes, nós lidamos com questões matemáticas, problemas de engenharia e tudo mais, como humano, temos a capacidade de lidar e resolver certas coisas, portanto, a máquina entra como um aspecto para essas inúmeras alçadas.

A visão clássica da IA considera a criação de sistemas que mimetizam a inteligência humana em suas várias facetas, como raciocínio lógico, aprendizado, e resolução de problemas. Essa abordagem, frequentemente associada à inteligência

artificial forte, busca criar máquinas capazes de realizar qualquer tarefa que um ser humano possa fazer.

Além das abordagens técnicas, os debates éticos sobre IA têm dado forma a diferentes concepções. Questões de autonomia, consciência e responsabilidade têm levado a uma reflexão crítica sobre até que ponto as máquinas podem ser verdadeiramente inteligentes e em que medida podem ser responsabilizadas por suas ações.

2.3. Aplicação das IA's

As aplicações das I.A são inúmeras, sendo formuladas no próprio dia a dia social, nas interações básicas que compõem a habitualidade do ser humano. Voltando ao objetivo principal da criação e resolução de problemas das máquinas, fica fácil entender as aplicações atualmente. Inúmeras áreas como a robótica, sistemas especialistas, sistemas visuais e etc, são tangíveis no uso da IA, sendo suas aplicações dentro delas muito impactantes.

As áreas são inúmeras, como uso do Google Maps, aplicativos de celular, programas de resposta rápida, widgets como a Alexa e muito mais. Dentro dessas existem outras sub áreas extremamente importantes para de fato se aplicar o uso das I.A. Segundo Association for the Advancement of Artificial Intelligence(AAAI), são essas subáreas:

a) “Pesquisa; b) Machine Learning; c) Data Mining e Big Data; d) Planejamento Automatizado; e) Representação de Conhecimento; f) Raciocínio (Probabilístico ou não); g) Processamento de Linguagem Natural; h) Robótica; i) Sistema de Agente e j) Multi-Agente e Aplicações” (ALVES, 2020)

3. IA PARA DESENVOLVIMENTO DE SOFTWARE

A convergência entre inteligência artificial (IA) e desenvolvimento de software tem resultado na criação de ferramentas inovadoras destinadas a aprimorar a eficiência e a produtividade no campo da programação. Este artigo visa realizar uma análise aprofundada de algumas dessas ferramentas, considerando seus contextos de lançamento, status de gratuidade, desempenho, características distintivas e teste das ferramentas.

3.1. TabNine

Lançada em 2019, a TabNine incorpora um avançado modelo de linguagem para fornecer sugestões de código precisas e eficientes. Além da versão gratuita, a TabNine Pro, uma opção paga, oferece funcionalidades adicionais para atender às demandas mais avançadas dos desenvolvedores. A ferramenta é reconhecida por sua notável rapidez e precisão nas sugestões de código (TabNine, 2019).

Para verificar a eficácia das funcionalidades da ferramenta, elaborou-se um código específico para a validação de CEP (Código de Endereçamento Postal). Após a realização de testes abrangentes, a ferramenta revelou diversas funcionalidades, como destacado nas Figuras 1 a 6:

Figura 1. A figura exibe algumas das funcionalidades existentes.

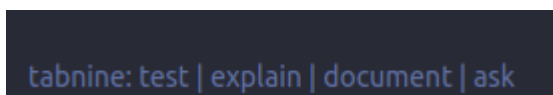


Figura 2. A figura exibe uma sugestão do TabNine para a função "run(cep)" no código-fonte, destacando a assistência inteligente fornecida pela ferramenta durante o desenvolvimento de software.

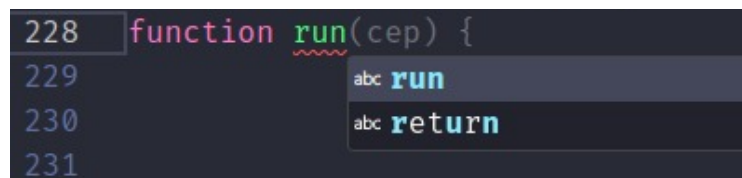


Figura 3. A figura apresenta o uso da ferramenta na funcionalidade "Ask", destacando claramente a pergunta feita e a resposta fornecida pela ferramenta. Essa visualização proporciona uma compreensão direta da interação entre o usuário e a capacidade de resposta da ferramenta diante de perguntas específicas.

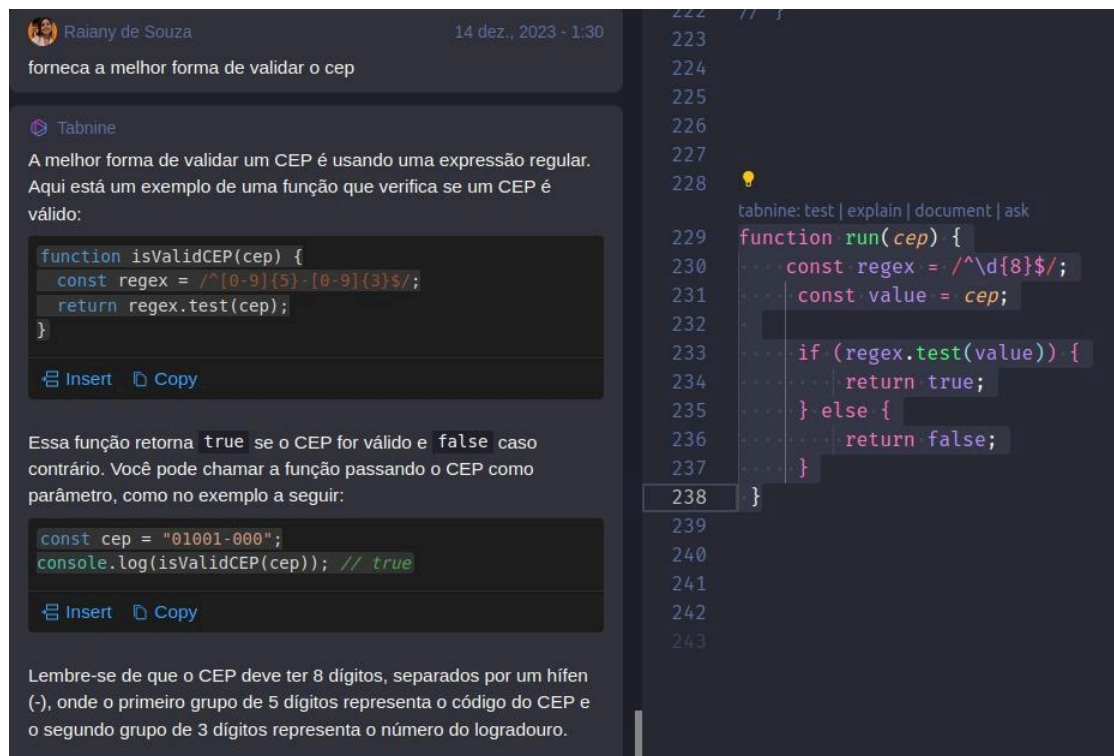


Figura 4. A figura ilustra o emprego da ferramenta na funcionalidade "Document", na qual é gerada uma documentação dinâmica para a função `run`. Essa documentação inclui detalhes sobre o parâmetro `cep`, seu tipo esperado e uma descrição do propósito da função. No caso de possíveis melhorias na implementação da função `run`, o TabNine Document tem a capacidade de sugerir correções automáticas, oferecendo uma contribuição valiosa para aprimorar a qualidade e robustez do código.

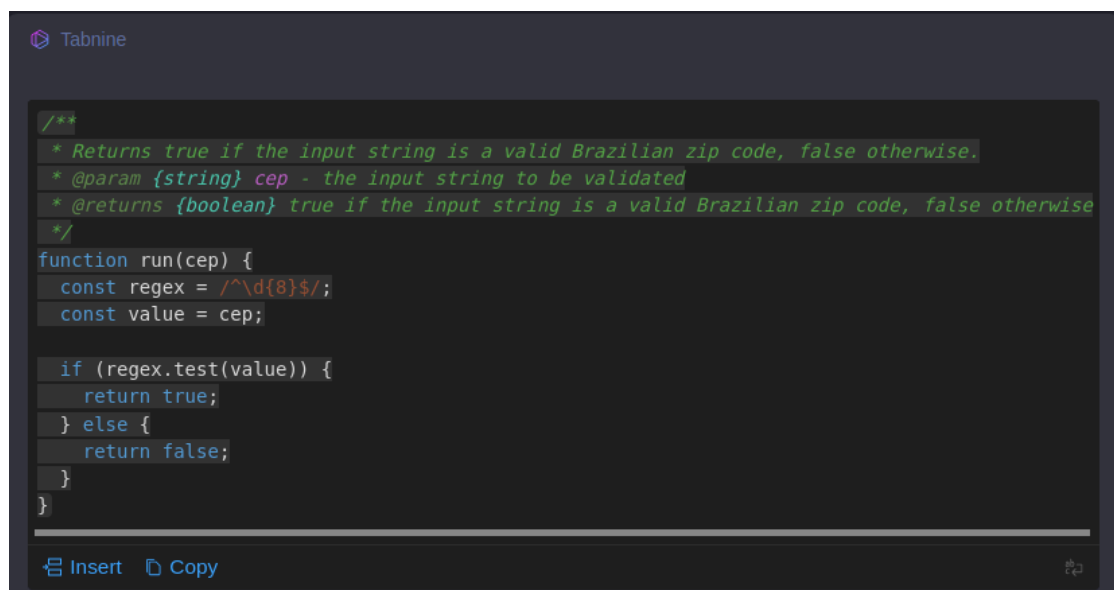


Figura 5. A figura destaca a utilização da funcionalidade "Explain" da ferramenta, na qual é possível visualizar a explicação do código em questão. Notavelmente, a ferramenta fornece a resposta em inglês..

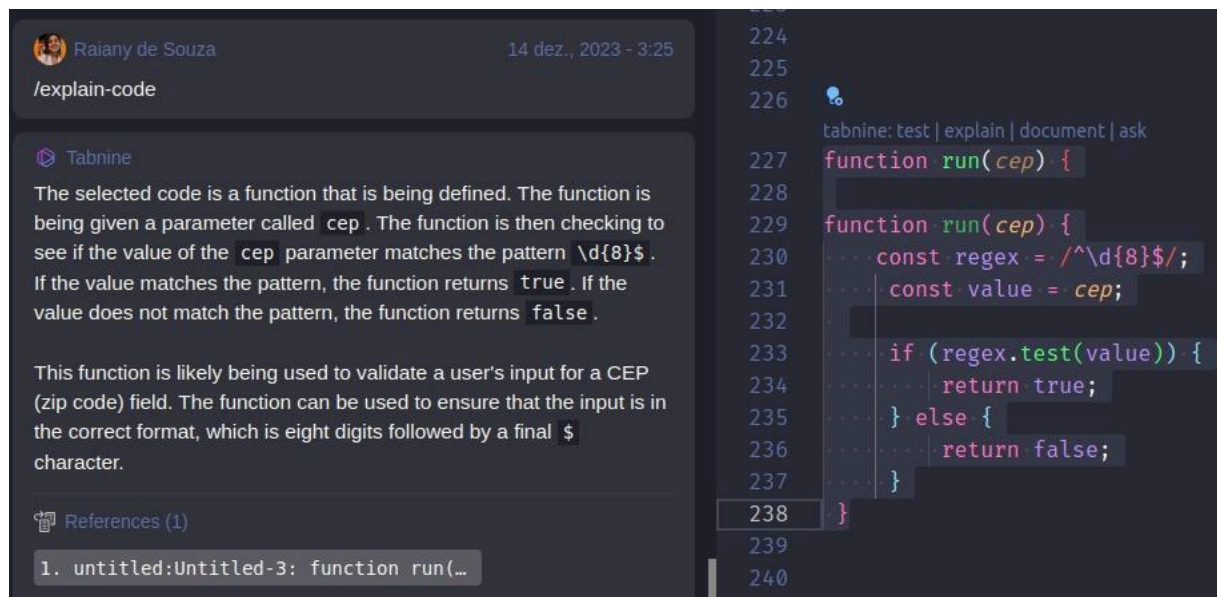
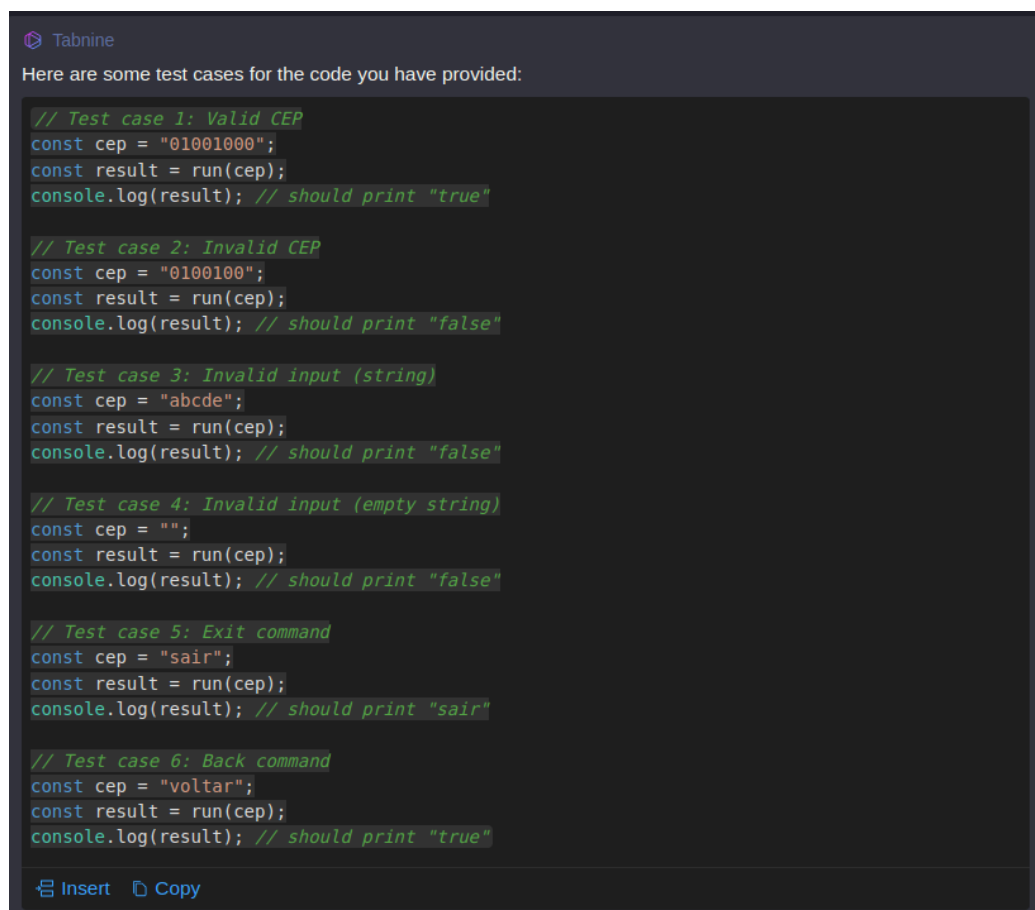


Figura 6. A figura apresenta uma sequência de códigos que podem ser empregados para testar a função em questão por meio da funcionalidade "Test".



3.2. GitHub Copilot

O GitHub Copilot foi anunciado em 2021. Utilizando o poderoso modelo GPT-3, o Copilot gera automaticamente trechos de código com base em descrições fornecidas. Incluído como parte do GitHub Codespaces, o Copilot possui diferentes planos de preços para atender às necessidades dos desenvolvedores. A integração do GPT-3 confere ao Copilot a capacidade única de gerar código automaticamente, com base em instruções em linguagem natural (GitHub, 2021).

Ao integrar-se diretamente aos ambientes de desenvolvimento, como o Visual Studio Code, GitHub Copilot transforma a maneira como os programadores concebem e implementam soluções. A partir de comentários, descrições de funções e até mesmo de parciais trechos de código, Copilot antecipa as intenções do desenvolvedor, gerando sugestões precisas e contextuais. Essa capacidade não apenas acelera o processo de codificação, mas também serve como uma ferramenta educacional valiosa, proporcionando insights instantâneos sobre práticas recomendadas e padrões de codificação.

Para validar as funcionalidades da ferramenta, foi desenvolvido um código destinado à validação de CPF, e foram realizados testes abrangentes, conforme ilustrado nas Figuras 1 a 6. Esses testes comprovaram a eficácia e a confiabilidade do GitHub Copilot, reforçando sua posição como uma ferramenta inovadora e indispensável para a comunidade de desenvolvedores.

Figura 1. A figura ilustra alguns comandos fornecidos à ferramenta na forma de comentários, indicando o que será necessário durante a escrita do código.

```
//faca um codigo em js para validar um cpf
//cpf deve ter 11 digitos e nenhuma formatacao
//exemplo 01234567891
//cpf deve ser um numero
//cpf nao pode ter numeros repetidos
```

Figura 2. Na figura, é possível observar a sugestão da ferramenta para o código solicitado pelos comandos na Figura 1.

```
function run(cpf) {
  const cpfNumeros = cpf.replace(/\D/g, '');
  const regex = /^\d{11}$/;
  const value = cpfNumeros;

  if (regex.test(value)) {
    return true;
  } else {
    return false;
  }
}
```

Figura 3. A figura apresenta o ambiente de trabalho da ferramenta, onde são visíveis atalhos para algumas funcionalidades.

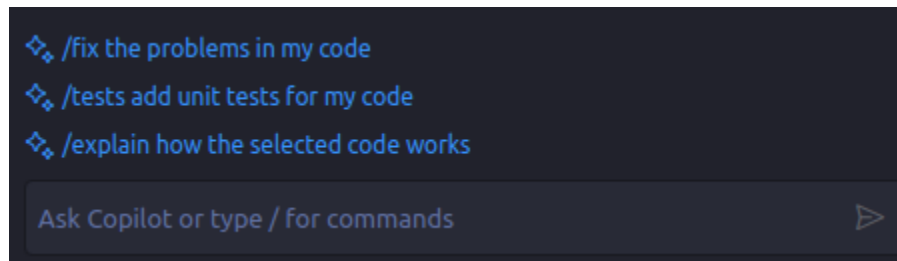


Figura 4. A figura ilustra o teste da funcionalidade "fix the problems in my code", onde a ferramenta examina o código em questão e identifica possíveis problemas que podem ser resolvidos pela solução já apresentada.



Figura 5. A figura apresenta o teste da funcionalidade "tests add unit tests for my code", revelando imediatamente todos os testes possíveis juntamente com seus respectivos resultados.

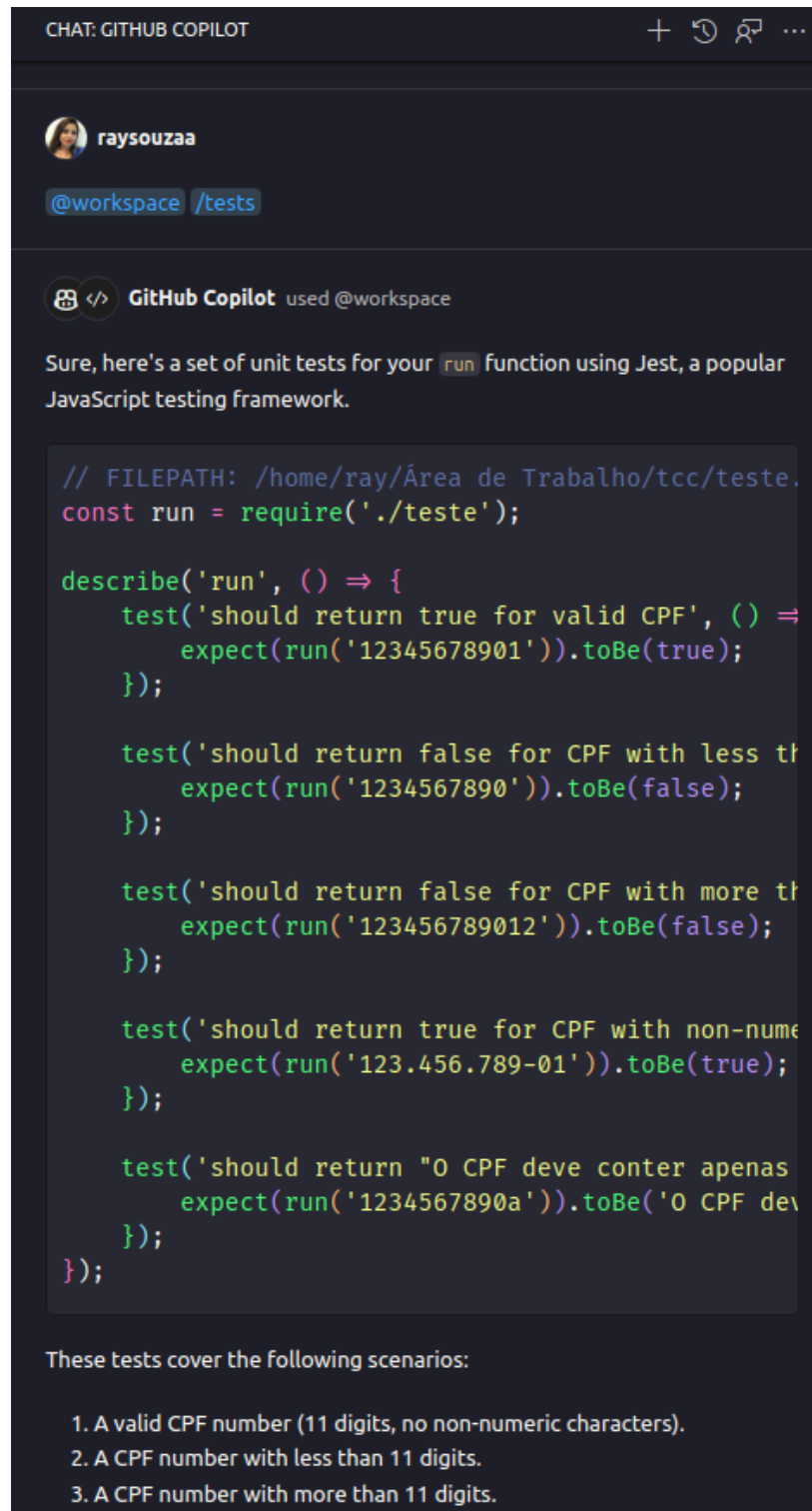
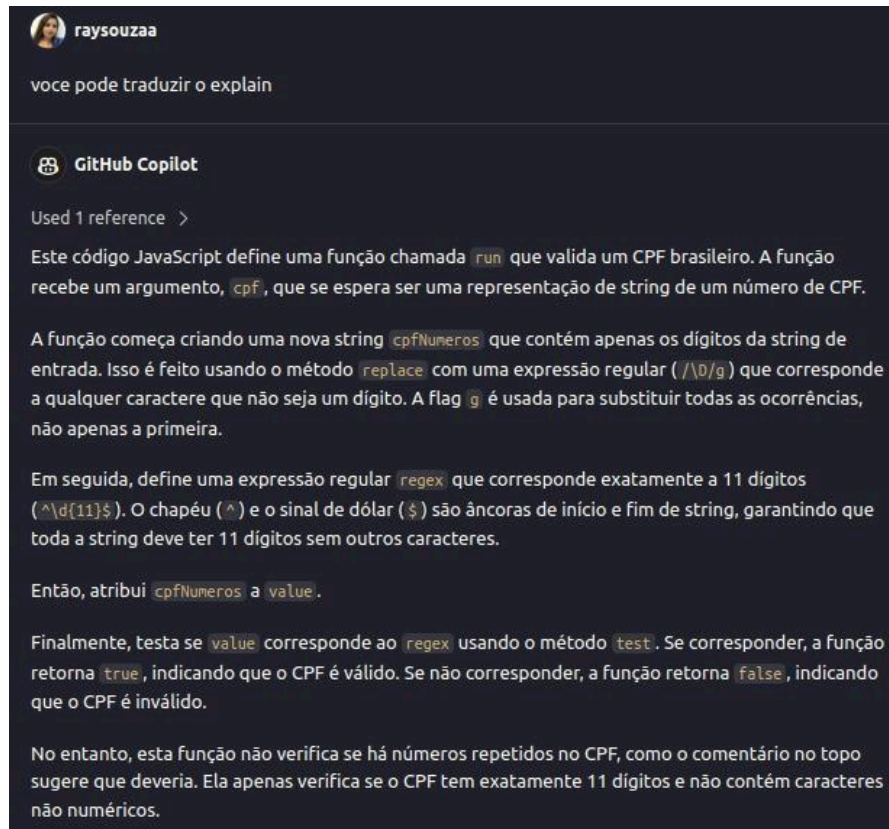


Figura 6. Na figura, ao utilizar a ferramenta "explain how the selected code works", foi possível requisitar a tradução do texto para o português brasileiro.



3.3. Amazon CodeWhisperer

O Amazon CodeWhisperer, lançado em 2022, produz sugestões de código em tempo real no ambiente de desenvolvimento integrado (IDE), que vão desde trechos até funções completas. Essas sugestões são geradas com base nos seus comentários e no código já existente.

O CodeWhisperer não só oferece suporte a conclusões de linha de comando (CLI) e tradução de linguagem natural para Bash durante a execução de comandos, mas também disponibiliza seu uso de forma gratuita para desenvolvedores individuais. Para organizações, o acesso à ferramenta está sujeito a uma taxa de assinatura fixa, cobrada "por usuário/por mês", sem a imposição de taxas iniciais ou a necessidade de compromissos de longo prazo.

A fim de avaliar a eficácia da ferramenta, elaborou-se um código dedicado à validação de números de telefone, sendo conduzidos testes detalhados, como evidenciado nas Figuras 1 a 3.

Figura 1. A figura apresenta uma sugestão de código inicial para a resolução do problema descrito nas linhas da IDE.

```
//make a function in js that receives the phone number and can validate it and return a message if it is invalid.  
function validatePhoneNumber(phoneNumber) {  
    if (phoneNumber.length !== 10) {  
        return "Invalid phone number";  
    }  
    if (!/^\\d+$/ .test(phoneNumber)) {  
        return "Invalid phone number";  
    }  
    return "Valid phone number";  
}
```

Figura 2. A figura enfatiza o uso da ferramenta por meio da Interface de Linha de Comando (CLI), destacando que a eficiência da ferramenta é otimizada ao inserir comandos em inglês. Além disso, a figura também oferece possíveis referências para a solução do problema.

improve the function regex for BR phone numbers

Here is one way to improve the validation function to handle Brazilian phone numbers: [1]

```
function validatePhone(phone) {  
    const regex = /^\\([1-9]{2}\\)\\s?[9]{1}[0-9]{4}-[0-9]{4}$/;  
  
    if(regex.test(phone)) {  
        return "Valid Brazilian phone number";  
    } else {  
        return "Invalid phone number";  
    }  
}
```

js Insert at cursor Copy

The changes made to the regular expression:

- Added parentheses to match the format of (xx)
- Added \\s? to optionally match a space after the closing parenthesis
- Changed \\d to [0-9] for clarity
- Added {1} to specify exactly 1 9 digit
- Added hyphen between blocks of numbers

This regex will match numbers in the format:

(xx) 9xxxx-xxxx

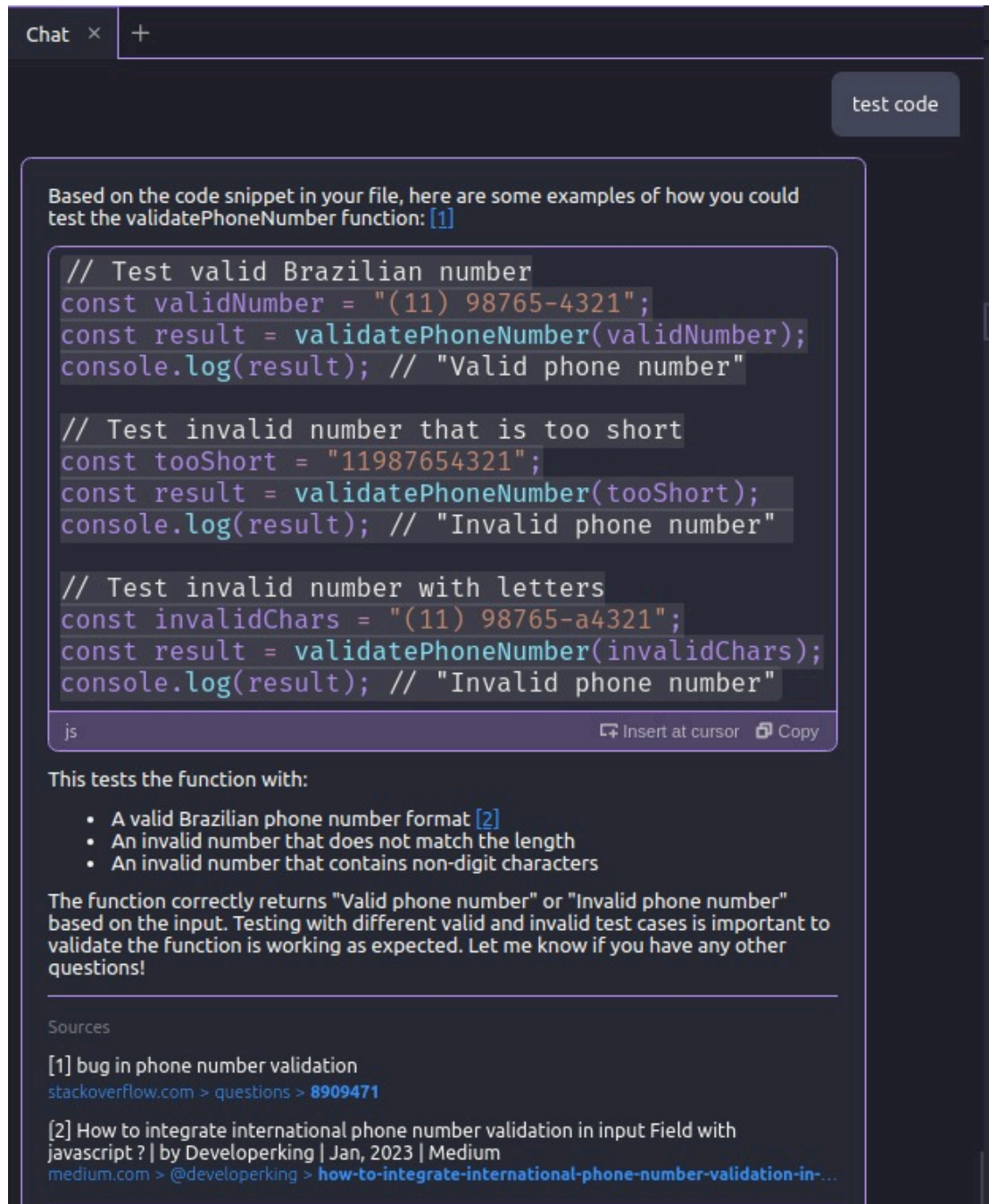
Where xx are digits 2-9, xxxx are digits 0-9.

Let me know if you need any part of the expression explained further or have additional requirements for validating Brazilian phone numbers. Regular expressions can certainly be improved to handle different number formats.

Sources

[1] Regular Expressions in JavaScript
mayukh551.hashnode.dev > [regular-expressions-for-beginners](#)

Figura 3. A figura destaca a capacidade de solicitar testes à ferramenta para o código fornecido, apresentando as potenciais soluções de teste disponíveis e fornecendo uma explicação detalhada para cada teste realizado.



3.4. Sourcery

Inserida no cenário em 2019, concentra-se na refatoração automática e geração de sugestões de código. Oferece uma versão gratuita com funcionalidades limitadas e uma versão paga, Sourcery Pro, que expande suas capacidades. Destaca-se por sua capacidade de aprimorar a estrutura do código, proporcionando sugestões significativas para otimização (Sourcery, 2019).

Para os propósitos de teste da ferramenta, conforme evidenciado nas figuras de 1 a 4, empregou-se um código destinado à avaliação das notas dos alunos. Destaca-se que o fornecimento dos códigos é realizado por meio do chat da ferramenta.

Figura 1. A figura exhibe algumas das funcionalidades existentes na IDE.

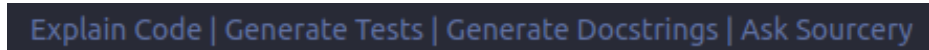


Figura 2. A figura exhibe a interface de chat da ferramenta, onde é possível visualizar o diálogo entre a IA, representado pela pergunta, a resposta fornecida e possíveis exemplos de uso.

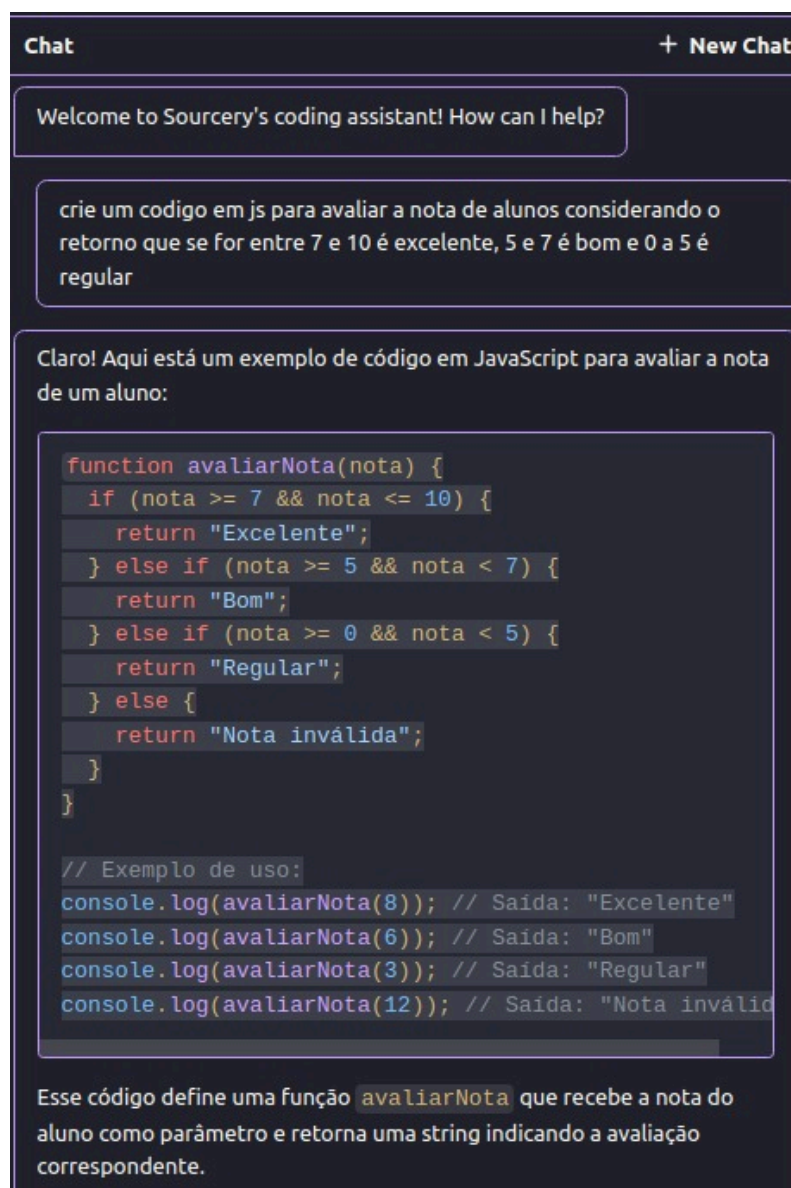


Figura 3. A figura destaca duas funcionalidades na interface da ferramenta. A primeira permite copiar o código fornecido, enquanto a segunda possibilita transferir o código para a página da IDE.

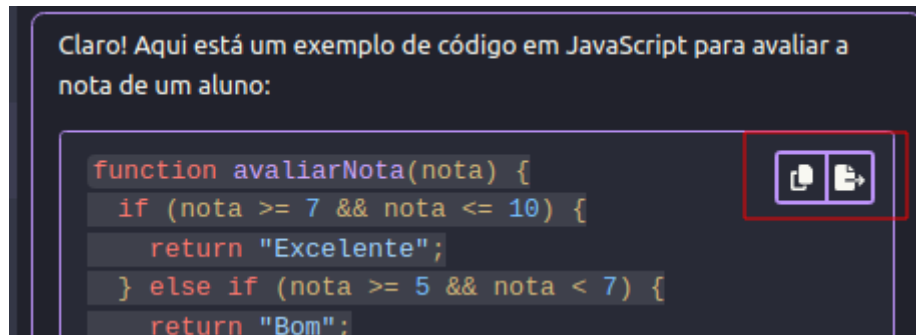


Figura 3. A figura mostra a solicitação de teste para o código em questão, bem como o resultado correspondente.

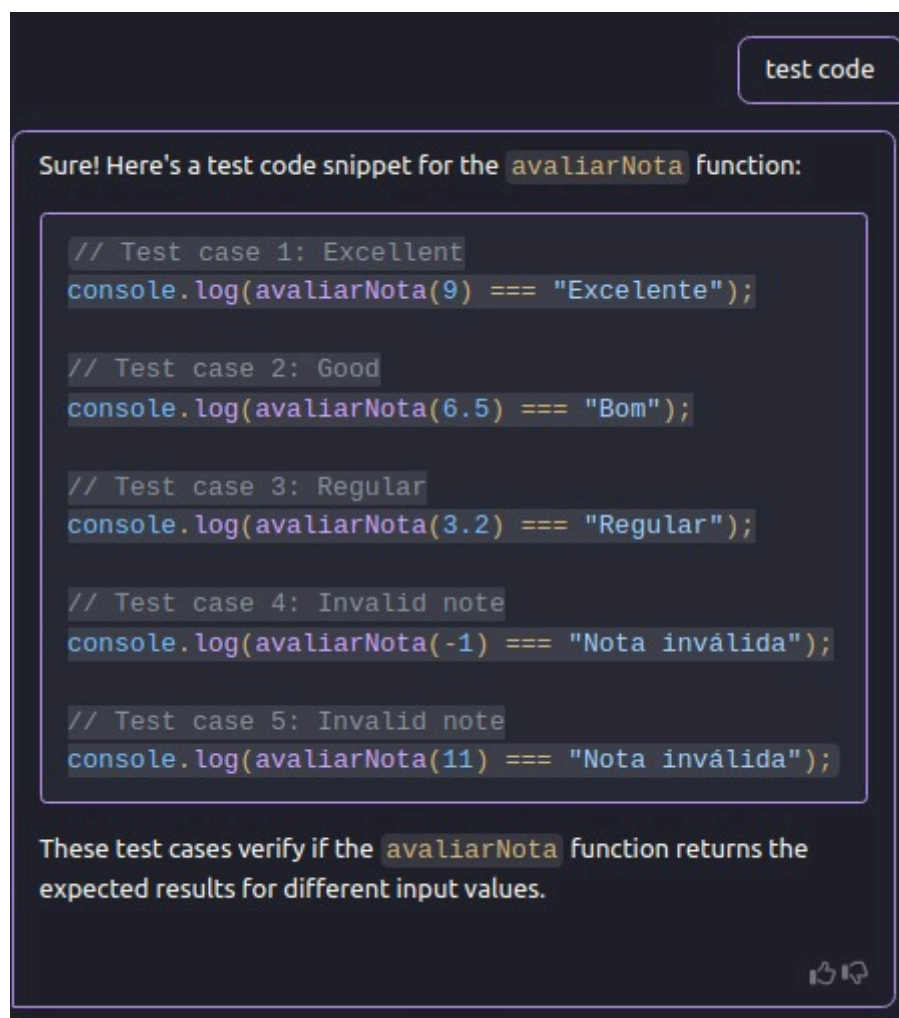


Figura 4. A figura mostra o teste da funcionalidade "Generate Docstrings", permitindo a visualização completa da documentação associada à função.

```

> ...
/**
 * Evaluates the grade of a student based on the given score.
 *
 * @param {number} nota - The score of the student.
 * @returns {string} The evaluation of the grade.
 *
 * @description
 * This function takes a numeric score as input and returns a string indicating the evaluation of the grade.
 * If the score is between 7 and 10 (inclusive), it returns "Excelente".
 * If the score is between 5 and 7 (exclusive), it returns "Bom".
 * If the score is between 0 and 5 (exclusive), it returns "Regular".
 * If the score is outside the valid range, it returns "Nota inválida".
 *
 * @example
 * // Returns "Excelente"
 * avaliarNota(9);
 *
 * // Returns "Bom"
 * avaliarNota(6.5);
 *
 * // Returns "Regular"
 * avaliarNota(3.2);
 *
 * // Returns "Nota inválida"
 * avaliarNota(-1);
 */
function avaliarNota(nota) {
  if (nota ≥ 7 && nota ≤ 10) {

```

3.5. Visual Studio IntelliCode

Introduzida em 2018, visa oferecer sugestões personalizadas com base em padrões de código previamente identificados. Gratuito como parte integrante do ambiente de desenvolvimento Visual Studio, utiliza o aprendizado de padrões para sugerir alternativas de código que se alinham ao estilo do desenvolvedor (IntelliCode, 2018).

Ao utilizar a ferramenta, notou-se que a geração de sugestões de código ocorre por meio de palavras, em vez de sugerir um código completo, como exemplificado na Figura 1.

Figura 1: A imagem mostra o início da escrita de um código em Python, utilizado para testar a ferramenta IntelliCode. Ela destaca as possíveis palavras e fornece informações sobre elas.

```
import re

def validar_email(email):
    #expressao regular para validar o formato do email

    padrao = r'^\S+@\S+\.\S+$'

    if re.m
```

M

★ match

Match

MULTILINE

```
def match(
    pattern: str | Pattern[str],
    string: str,
    flags: _FlagsType = 0
) → (Match[str] | None): ...

def match(
    pattern: bytes | Pattern[b],
    string: ReadableBuffer,
    flags: _FlagsType = 0
) → (Match[bytes] | None): ...
```

Try to apply the pattern at the start of the string, returning a Match object, or None if no match was found.

4. CONSIDERAÇÕES FINAIS

Este estudo abordou a evolução histórica e as aplicações contemporâneas da Inteligência Artificial (IA), com ênfase na sua convergência com o desenvolvimento de software. A complexidade inerente à engenharia de software foi destacada como um desafio enfrentado pelos desenvolvedores, sendo a IA uma ferramenta crucial para mitigar essas dificuldades.

Ao explorar a história da IA, desde suas origens até as realizações contemporâneas, observou-se um progresso notável em direção à criação de máquinas inteligentes. A diversidade de aplicações das IA's na vida cotidiana, incluindo áreas como robótica, sistemas especialistas e processamento de linguagem natural, destaca a relevância crescente dessa tecnologia.

O foco específico na interseção entre IA e desenvolvimento de software permitiu a análise de ferramentas inovadoras, como TabNine, GitHub Copilot, Amazon CodeWhisperer, Sourcery e Visual Studio IntelliCode. Essas ferramentas demonstram contribuições significativas para a eficiência e produtividade no campo da programação.

Em conclusão, este estudo proporcionou uma compreensão aprofundada das implicações históricas e contemporâneas da IA, destacando sua importância vital no

desenvolvimento de software. O avanço contínuo nessa convergência promete moldar significativamente o cenário da programação, influenciando positivamente a eficácia e qualidade do desenvolvimento de software.

5. REFERÊNCIAS UTILIZADAS

Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville

Coders at Work: Reflections on the Craft of Programming. Peter Seibel

Turing, A. (1950). Computing Machinery and Intelligence. Mind, 59(236), 433-460.

The Mythical Man-Month Essays on Software Engineering. Frederick P. Brooks, Jr.

ALVES, P. M. Inteligência artificial e redes neurais. IPEA: Centro de Pesquisa em Ciência, Tecnologia e Sociedade: [Brasil], 2020. Disponível em: https://www.ipea.gov.br/cts/pt/centralde-conteudo/artigos/artigos/106-inteligencia-artificial-e-redes-neurais#_ftn1 Acesso em: 24 jul. 2021

TabNine. (2019). TabNine: Code Faster with the All-Language AI Assistant. Disponível em: <https://docs.tabnine.com/enterprise/private-installation/readme>.

GitHub Copilot. (2021). GitHub Copilot: Your AI Pair Programmer. Disponível em: <https://github.com/features/copilot>.

Amazon CodeWhisperer. (2022). Amazon CodeWhisperer - Sua ferramenta de produtividade baseada em IA para o IDE e a linha de comandos. Disponível em: <https://aws.amazon.com/pt/codewhisperer/>.

Sourcery. (2019). Sourcery - Refactor Python Code. Disponível em: <https://sourcery.ai/>.

IntelliCode. (2018). Visual Studio IntelliCode. Disponível em: <https://visualstudio.microsoft.com/pt-br/services/intellicode/>.

