AICodeReview: Uma Ferramenta para Revisão de Código por Meio de Inteligência Artificial

Yonatha Almeida[1], **Danyllo W. Albuquerque**[1,2], **Mirko Perkusich**[2], **Katyusco de Farias Santos**[1]

[1]Instituto Federal da Paraíba (IFPB)

[2]Universidade Federal de Campina Grande (UFCG)

{yonatha.almeida, danyllo.albuquerque, katyusco.santos}@ifpb.edu.br

Abstract. *This article presents a study on the use of Artificial Intelligence (AI) in code review with the aim of improving the quality and efficiency of this activity. For this, a plugin was developed for IntelliJ IDEA using ChatGPT as a basis for automated code review. The tool performs analysis of code snippets to identify syntax and semantic problems, suggesting possible solutions. As a result, we demonstrated the architectural details, configuration mode, and usage scenarios of the tool in detecting logic errors and syntax problems. It is concluded that using AI technologies can be an efficient way to reduce the time and effort required to perform code reviews, contributing to software quality improvement.*

Resumo. *Este artigo apresenta um estudo sobre o uso de inteligência artificial (IA) na revisão de código com o objetivo de melhorar a qualidade e a eficiência dessa atividade. Para isso, desenvolveu-se um plugin para o IntelliJ IDEA utilizando o ChatGPT como base para a revisão automatizada de código. A ferramenta realiza análise de trechos de código para identificar problemas de sintaxe e semântica, sugerindo possíveis soluções. Como resultado, demonstra-se os detalhes arquiteturais, o modo de configuração e cenários de uso da ferramenta na detecção de erros de lógica e problemas de sintaxe. Conclui-se que a utilização de tecnologias de IA pode ser uma maneira eficiente de reduzir o tempo e o esforço necessários para realizar a revisão de código, contribuindo para melhoria da qualidade do software.*

1. Introdução

Revisões de código são uma prática comum na engenharia de software que envolve a análise sistemática do código-fonte por desenvolvedores, com o objetivo de identificar e corrigir possíveis problemas sintáticos e semânticos[Bacchelli and Bird 2013]. As revisões de código podem ser feitas de várias maneiras, incluindo revisão manual do código-fonte, revisões automatizadas com ferramentas de análise de código, revisões por pares e revisões em grupo [Tizard and Liu 2018]. No entanto, revisar manualmente grandes volumes de código pode ser um processo demorado e tedioso para os desenvolvedores, surgindo a necessidade de simplificar e aprimorar esse processo através de ferramentas [Rigby 2014].

A utilização de ferramentas de Inteligência Artificial (IA) para revisão de código pode ser uma maneira eficiente de melhorar a qualidade do software [Wang et al. 2018]. O ChatGPT, o *Google Bard* e outras tecnologias emergentes de IA no mercado podem ser úteis no contexto da revisão de código, pois possuem recursos avançados de análise de linguagem natural e aprendizado de máquina. A partir da utilização destas tecnologias, ferramentas podem estar aptas a (i) analisar e entender a estrutura sintática e semântica do código; (ii) identificar e analisar possíveis problemas bem como catalogar; e (iii) priorizar sugestões para resolução destes problemas [Gupta et al. 2020].

Neste contexto, o presente trabalho apresenta o desenvolvimento de um *plugin* para o IntelliJ IDEA[1] que utiliza o ChatGPT[2], uma tecnologia de inteligência artificial baseada em aprendizado de máquina, para realizar a revisão de código de forma automatizada. Ao utilizar o ChatGPT como base para a revisão de código, é possível explorar recursos avançados de análise de linguagem natural para identificar problemas de sintaxe, erros de lógica e vulnerabilidades de segurança. O *plugin* também fornece sugestões para soluções e melhorias no código, simplificando o processo de revisão e garantindo um resultado mais eficiente.

Mediante uso do *plugin*, os desenvolvedores podem contar com uma ferramenta poderosa para identificar possíveis problemas no código, melhorando a qualidade e a segurança do software em desenvolvimento. Assim, a tarefa de revisão de código pode ser realizada de forma automatizada, poupando tempo e esforço dos desenvolvedores e aumentando a produtividade da equipe. Este artigo apresenta os passos necessários para a construção da ferramenta *AICodeReview*, detalhes a respeito de sua configuração e utilização, bem como implicações práticas do uso desse tipo de ferramenta no contexto da revisão de código.

2. Fundamentação Teórica

Nesta seção são apresentados os conceitos essenciais para compreensão deste trabalho. Inicialmente, apresenta-se os conceitos associados à revisão de código (Seção 2.1). Em seguida, descreve-se aspectos associados à inteligência artificial e aplicações (Seção 2.2). Por fim, detalha-se o ambiente de desenvolvimento integrado utilizado no estudo (Seção 2.3).

2.1. Revisões de Código

As revisões de código são uma prática comum na engenharia de software que envolve a análise sistemática do código-fonte por desenvolvedores, com o objetivo de identificar e corrigir possíveis problemas [Bacchelli and Bird 2013]. Essa técnica é utilizada para melhorar a qualidade do código, reduzir erros e garantir que o software atenda aos requisitos do usuário [Tizard and Liu 2018]. A revisão de código pode ser feita de várias maneiras, incluindo revisão manual do código-fonte, revisões automatizadas com ferramentas de análise de código, revisões por pares e revisões em grupo.

A revisão manual é uma abordagem em que um ou mais desenvolvedores examinam o código-fonte linha por linha, procurando por possíveis problemas [Albuquerque et al. 2022a]. Essa abordagem pode ser demorada e tediosa, além de exigir que os desenvolvedores tenham um conhecimento aprofundado da linguagem de programação em questão [Rigby 2014]. Por outro lado, as revisões automatizadas podem ser mais eficientes em termos de tempo e recursos. Essas revisões podem ser realizadas por ferramentas de análise de código que verificam automaticamente o código-fonte em busca de possíveis erros, como problemas de sintaxe, vazamentos de memória, vulnerabilidades de segurança e outras questões de qualidade de código [Tizard and Liu 2018].

2.2. Inteligência Artificial e ChatGPT

A inteligência artificial (IA) é um campo da ciência da computação que envolve o desenvolvimento de sistemas capazes de realizar tarefas que normalmente requerem inteligência humana, como aprendizado, raciocínio, percepção e tomada de decisão [Chen et al. 2018]. A utilização de IA para revisão de código é uma área em crescimento na engenharia de software [Albuquerque et al. 2022c], uma vez que estas tecnologias podem oferecer uma análise mais

[1]https://www.jetbrains.com/idea/

[2]https://openai.com/blog/chatgpt

avançada do código do que as revisões manuais tradicionais. Alguns exemplos de técnicas de IA comumente utilizadas na revisão de código incluem processamento de linguagem natural, aprendizado de máquina, mineração de dados e análise estática de código. A combinação dessas técnicas pode ajudar a identificar problemas e sugerir soluções para melhorar a qualidade do código e, por sua vez, do software em geral [Tian et al. 2020].

Nesse contexto surge a utilização da tecnologia emergente ChatGPT (*Generative Pretrained Transformer*), um modelo de linguagem de IA desenvolvido pela OpenAI, que tem a capacidade de gerar texto em linguagem natural de alta qualidade. Na revisão de código, tecnologias como o ChatGPT podem ser empregadas para identificar e classificar possíveis problemas de sintaxe, tais como erros de digitação, declarações incorretas de variáveis e outros erros gramaticais que podem afetar o desempenho do software [Arora et al. 2021]. Adicionalmente, podem ser usadas para analisar a lógica do código, identificando possíveis problemas de fluxo de controle, laços infinitos e outros problemas que possam afetar a qualidade do software [Arora et al. 2021]. Por fim, a tecnologia do ChatGPT pode ser utilizada para detectar vulnerabilidades de segurança, incluindo possíveis ataques de injeção de SQL e outros tipos de vulnerabilidades de segurança que podem ser exploradas por *hackers*.

2.3. IntelliJ e construção de *plugins*

O IntelliJ IDEA é um dos ambientes de desenvolvimento integrados (do inglês, *Integrated Development Envinronment* [IDE]) mais populares para desenvolvimento de software [Hartmann 2020]. Este ambiente fornece suporte para uma ampla variedade de linguagens de programação incluindo Java, Kotlin, Groovy e Scala [Breault 2017]. Ele é reconhecido pela comunidade de desenvolvimento por sua capacidade de expansão através de *plugins*, que são pequenos módulos de software que adicionam novos recursos e funcionalidades a IDE. Os *plugins* podem ser desenvolvidos por terceiros ou pela própria equipe do IntelliJ, sendo facilmente instalados e gerenciados dentro do IDE [Hartmann 2020].

A criação de um *plugin* para o IntelliJ envolve o uso da linguagem de programação Java e a utilização da API do próprio IntelliJ para criar novas funcionalidades. O processo de desenvolvimento do *plugin* é facilitado pelo conjunto de ferramentas disponibilizado pelo IntelliJ, como a documentação da API, o assistente de criação de projetos e a ferramenta de depuração [Hartmann 2020]. Para construir um *plugin* para o IntelliJ IDEA, é necessário seguir algumas etapas fundamentais, como a definição do escopo, a criação do projeto, a implementação das funcionalidades e a integração com o IntelliJ IDEA [Breault 2017]. É importante destacar que o IntelliJ IDEA possui uma grande comunidade de desenvolvedores de *plugins*, o que torna a tarefa de construção de novos *plugins* acessível a suporte especializado em tempo hábil.

3. Exemplo Motivador

Para motivar a utilização da ferramenta *AICodeReview*, descreve-se um exemplo motivador. Consideremos dois desenvolvedores trabalhando em um projeto de software: Alice e Bob. Ambos estão desenvolvendo uma nova funcionalidade e precisam revisar o código para garantir que ele está livre de erros e vulnerabilidades. Alice decide utilizar o *AICodeReview*, enquanto Bob opta por fazer a revisão manualmente, sem o auxílio desta ferramenta.

Mediante suporte da ferramenta, Alice consegue identificar rapidamente vários problemas no código que Bob não percebeu durante sua revisão manual. Por exemplo, o mecanismo de IA provido pelo *AICodeReview* identifica diversos erros como (i) uma variável que não está sendo inicializada corretamente, (ii) um potencial vazamento de memória em um trecho

de código e (iii) um erro de lógica em uma estrutura condicional. Além disso, o *AICodeReview* forneceu oportunamente sugestões para solucionar tais problemas, permitindo que Alice compreendesse e corrigisse os erros mais rapidamente.

Por outro lado, Bob teve dificuldades para identificar alguns desses problemas durante sua revisão manual e, consequentemente, demandou mais tempo e esforço para corrigi-los. Como resultado, o trabalho de Bob além de mais demorado, apresentou mais erros e vulnerabilidades se comparado ao trabalho realizado por Alice mediante suporte do *AICodeReview*. Isso conduziu a erros que tinham possibilidade de impactar negativamente a qualidade e a segurança do software, colocando em risco os usuários e as informações do sistema.

O uso do *AICodeReview* mostrou-se uma solução eficiente que pode trazer inúmeros benefícios para os desenvolvedores. A ferramenta contribui para melhoria das habilidades e conhecimentos sobre programação, fornecendo *insights* valiosos sobre a arquitetura do software, a lógica de programação e as boas práticas de codificação. Outra vantagem do uso é a possibilidade de encontrar erros que poderiam passar despercebidos durante a revisão manual. Por fim, o *AICodeReview* também pode fornecer sugestões para soluções de problemas, o que pode ser especialmente útil em projetos de software que envolvem prazos apertados ou onde a qualidade do código é crítica para o sucesso do projeto

4. Trabalhos Relacionados

Esta seção descreve alguns trabalhos relacionados à utilização de técnicas de IA no contexto da revisão de código. Entre estes trabalhos, destaca-se o estudo de Singh e Kumar [Singh and Kumar 2019] que realizaram uma revisão sistemática para avaliar o uso de técnicas de aprendizado de máquina em revisões de código. Os autores identificaram 33 estudos, indicando que o uso de IA e aprendizado de máquina podem melhorar a eficácia e eficiência das revisões de código automatizadas. No que segue, descreve-se algumas ferramentas específicas que aplicam alguma técnica de IA em revisões de código.

Li et al. [Li and Li 2018] desenvolveram um sistema de revisão de código chamado CoderAI, que usa técnicas de aprendizado de máquina para identificar possíveis problemas de qualidade de código. Similarmente, Zhang et al. [Zhang et al. 2018] desenvolveram um sistema chamado DeepCom, que usa redes neurais para gerar comentários de código automaticamente. Ainda, Sadowski e Grechanik [Sadowski and Grechanik 2016] desenvolveram um sistema chamado CLAMI (Classificador de Problemas de Manutenção de Código) que usa técnicas de aprendizado de máquina para classificar problemas de código em categorias. Os resultados destes trabalhos mostraram que, de modo geral, as ferramentas de revisão de código propostas obtiveram desempenho superior ao de revisores humanos em determinados cenários.

Como outros trabalhos, Nagappan et al.[Nagappan et al. 2014] desenvolveram um sistema de revisão de código chamado CodeRush, que usa análise estática e técnicas de aprendizado de máquina para identificar possíveis problemas de código. Analogamente, Jiang et al. [Jiang et al. 2011] desenvolveram um sistema chamado Jupiter, que usa regras para identificar possíveis problemas de código. Ainda, Albuquerque et al. [Albuquerque et al. 2022b] desenvolveram uma ferramenta que mescla analise estática e decisão baseada em critérios para detecção contínua de anomalias de código. Ambos os trabalhos avaliaram suas ferramentas do ponto de vista de eficácia, demonstrando resultados promissores do uso das respectivas ferramentas em detrimento a revisões manuais ou semi-automatizadas.

Cada um desses trabalhos apresenta uma abordagem diferente para a utilização de técnicas de IA na revisão de código. Algumas dessas abordagens utilizam aprendizado de

máquina, enquanto outras focam em gerar comentários descritivos sobre o código. Há também trabalhos que utilizam abordagens mistas, combinando IA e análise estática. A diferença deste trabalho em relação aos demais é a utilização específica do ChatGPT, uma tecnologia emergente baseada em aprendizado de máquina e análise de linguagem natural. Além disso, este trabalho propõe o desenvolvimento de um *plugin* para o IntelliJ IDEA, uma ferramenta popular entre os desenvolvedores. O *plugin* utiliza recursos avançados de análise de linguagem natural para identificar problemas de sintaxe, erros de lógica e vulnerabilidades de segurança, fornecendo sugestões para soluções e melhorias no código. Portanto, este trabalho representa um avanço no estado da arte ao utilizar uma tecnologia emergente e apresentar uma abordagem inovadora para a revisão de código, com um *plugin* específico para uma das IDEs mais utilizadas pelos desenvolvedores atualmente.

5. A Ferramenta *AiCodeReview*

Esta seção apresenta a ferramenta *AICodeReview* em detalhes. Inicialmente, apresenta-se uma visão geral da ferramenta (Seção 5.1). Em seguida, descreve-se detalhes da arquitetura e implementação (Seção 5.2). Finalmente, exibe-se detalhes a respeito do funcionamento desta ferramenta (Seção 5.3).

5.1. Características e Funcionalidades

A ferramenta *AICodeReview*[3] encontra-se disponível para uso de forma gratuita junto ao *Marketplace* do IntelliJ IDE. Sua primeira versão operacional foi publicada em meados de março de 2023 e já foi utilizada por mais de 2.300 usuários distintos. Uma vez que a ferramenta seja configurada no ambiente de desenvolvimento integrado IntelliJ IDE, o usuário pode interagir com esta ferramenta para realizar uma ou mais das seguintes atividades abaixo descritas:

- **Suporte para revisão de código em diversas linguagens de programação**: O *AICodeReview* oferece suporte para revisão de código em diversas linguagens de programação relevantes no mercado, como Java, Kotlin, C++, Python, entre outras. Isso significa que a ferramenta pode ser utilizada em projetos desenvolvidos mediante uso de diferentes linguagens, sem quaisquer restrições.
- **Parametrização nas sugestões de soluções**: Com o *AICodeReview*, é possível ajustar a precisão das sugestões para resolução de problemas na revisão de código visando atender às necessidades específicas de cada projeto. Essa configuração permite que o usuário ajuste o nível de detalhamento das sugestões, de forma que ele possa receber desde sugestões gerais até sugestões mais específicas e detalhadas.
- **Explanação e demonstração do raciocínio nas sugestões**: Uma das principais características do *AICodeReview* é a capacidade de sugerir revisões de código, com explicações detalhadas sobre o raciocínio utilizado para a sugestão. Dessa forma, o usuário pode entender o motivo da sugestão e aprender com as recomendações feitas pela ferramenta.
- **Preferências de idioma**: O *AICodeReview* oferece suporte para mensagens em diferentes idiomas, como inglês, espanhol e português. Essa característica torna o uso da ferramenta mais acessível a usuários que não falam inglês, por exemplo, permitindo que eles recebam mensagens em seu idioma de preferência.
- **Compatibilidade com produtos da família Jetbrains**: O *AICodeReview* é compatível com todos os produtos da família Jetbrains, o que significa que ele pode ser utilizado em conjunto com outras ferramentas desenvolvidas pela mesma empresa (e.g.,

[3]https://plugins.jetbrains.com/plugin/21106-ai-code-review/

PyCharm, WebStorm e o Android Studio). Isso permite que o usuário tenha uma experiência de desenvolvimento mais integrada e eficiente em diversas ferramentas.

Em suma, o *AICodeReview* é uma ferramenta altamente versátil e eficaz que pode ser usada para revisar códigos em várias linguagens de programação. Possui recursos de personalização de precisão e suporte para sugestões gerais ou específicas. Além disso, é uma ferramenta de aprendizado, pois oferece explicações detalhadas do raciocínio usado para a sugestão de revisão de código. Isso pode ser particularmente útil no contexto dos erros semânticos que podem ser mais difíceis de identificar e requerer um processo mais elaborado de depuração e correção. Ademais, o *AICodeReview* é uma ferramenta altamente recomendada para qualquer desenvolvedor que deseja revisar seu código de forma simples, precisa e eficaz.

5.2. Detalhes Arquiteturais

O *AICodeReview* foi desenvolvido utilizando a linguagem de programação Java, sendo composto por oito componentes, descritos na Figura 1, que se relacionam entre si, permitindo a integração da IntelliJ Platform com a API do ChatGPT.

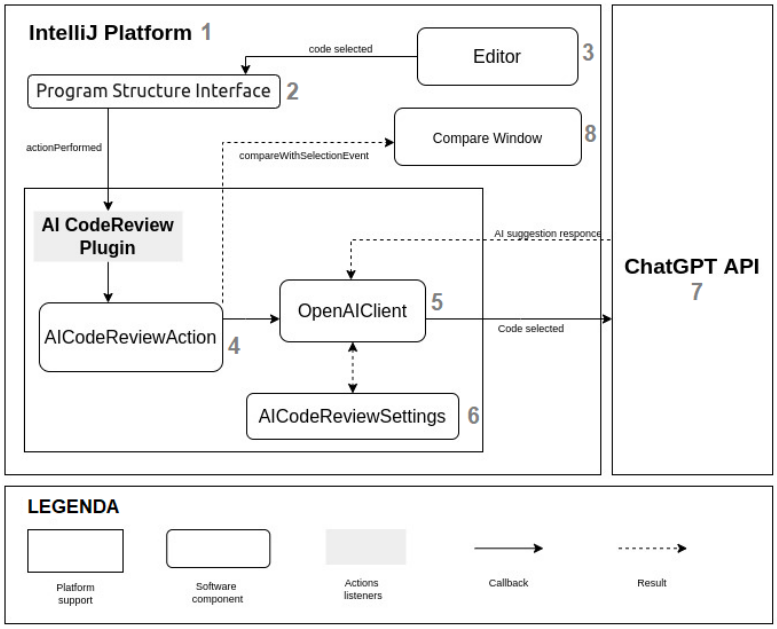

Figura 1. Arquitetura da Ferramenta.

1. **IntelliJ Platform**. É um conjunto de ferramentas de desenvolvimento de software criado pela JetBrains, composto pelo IntelliJ IDEA, fornecendo infraestrutura para criar aplicativos de desktop, web e mobile para diversos domínios como finanças, saúde, governo e tecnologia. A plataforma possibilita que *plugins* desenvolvidos tanto em linguagem Java ou Kotlin, tenham acesso a recursos da IDE (e.g., editor de código, depurador, menus e janelas) através de sua SDK.
2. **Program Structure Interface (PSI)**. Esse componente possui o conjunto de funcionalidades que possibilita navegação rápida para arquivos, tipos e símbolos, até inspeções e refatoração do código. Embora o *AICodeReview* tenha sido criado em Java, seu uso não se restringe a revisão de códigos nesta linguagem, devido ao poder do PSI e seus modelos sintáticos e semânticos de código compatíveis com múltiplas linguagens.
3. **Editor**. É o componente onde os desenvolvedores escrevem o código utilizando recursos da PSI que interpreta a linguagem de programação utilizada. É através deste

componente que o usuário aciona *AICodeReview*, através da *actionPerformed*, o componente *AICodeReviewAction* através do *actionListener CodeReviewAction*, registrado no *groupid CodeMenu* do componente IntelliJ Platform, permitindo que a revisão de código seja acessada no menu principal Code da IDE, ou através do *actionListener* registrado com *grupoid EditorPopupMenu3*, permitindo o mesmo acesso conveniente função de revisão do código do *plugin*, no menu de contexto do componente *Editor*.

4. **AICodeReviewAction**. Esse componente representa a ação executada pelo usuário após selecionar o código que deseja obter sugestões da Inteligência Artificial.
5. **OpenAIClient**. Esse componente é acionado através do componente *AICodeReviewAction* e utiliza os parâmetros de configuração contidos no componente AICodeReviewSettings, realizando uma requisição para camada externa da IntelliJ Platform, descrita no componente ChatGPT API.
6. **AICodeReviewSettings**. É neste componente, que são armazenadas as informações apresentadas na Figura 1. Para que o componente *OpenAIClient* possa se comunicar com a ChatGPT API, este componente dispõe de configurações predefinidas como modelo (e.g., text-davinci-003, code-davinci-002, code-cushman-001, text-babbage-001, text-curie-001) e idioma (e.g., inglês, espanhol e português) das sugestões de revisão de código da ChatGPT API.
7. **ChatGPT API**. Esse componente representa a camada externa do *plugin*. Trata-se do serviço fornecido pela empresa desenvolvedora do ChatGPT (i.e., OpenAI). O respectivo componente utiliza um *token bearer*, que é configurado no componente *AICodeReviewSettings* no campo *Secret key*. Este componente é responsável por alimentar as sugestões de melhoria do código selecionado pelo usuário e exibi-las no componente *Compare Window*, através do *AICodeReviewAction*.
8. **Compare Window**. Este componente tem a função de comparar o código selecionado pelo usuário com o código sugerido do ChatGPT, para tal finalidade, são usados os recursos do PSI, para realizar a análise sintática e comparação em uma única janela como é apresentado na Etapa 4 da Figura 2.

A arquitetura do AICodeReview apresenta-se de forma modular, permitindo uma fácil integração da ferramenta com a IntelliJ Platform e a ChatGPT API (Figura 1). Seus módulos encontram-se bem definidos, facilitando a manutenção e a evolução da ferramenta. Além disso, a ferramenta possui um grande potencial de expansão, uma vez que ela pode ser facilmente adaptada para suportar outras linguagens de programação e ser integrada a outros serviços de inteligência artificial ou aprendizado de máquina. Dessa forma, a evolução da ferramenta pode proporcionar uma melhoria significativa no processo de revisão de código, trazendo ganhos em qualidade, produtividade e eficiência para os desenvolvedores e equipes de desenvolvimento.

5.3. Funcionamento

Essa seção demonstra em linhas gerais o funcionamento da ferramenta *AICodeReview*. No que segue iremos descrever os detalhes de sua configuração e utilização no contexto do ambiente de desenvolvimento integrado IntelliJ IDE.

5.3.1. Configuração

Antes do uso da ferramenta, é necessário realizar adequadamente sua configuração. A escolha da *engine* de IA é um passo importante na configuração da ferramenta. No momento, o

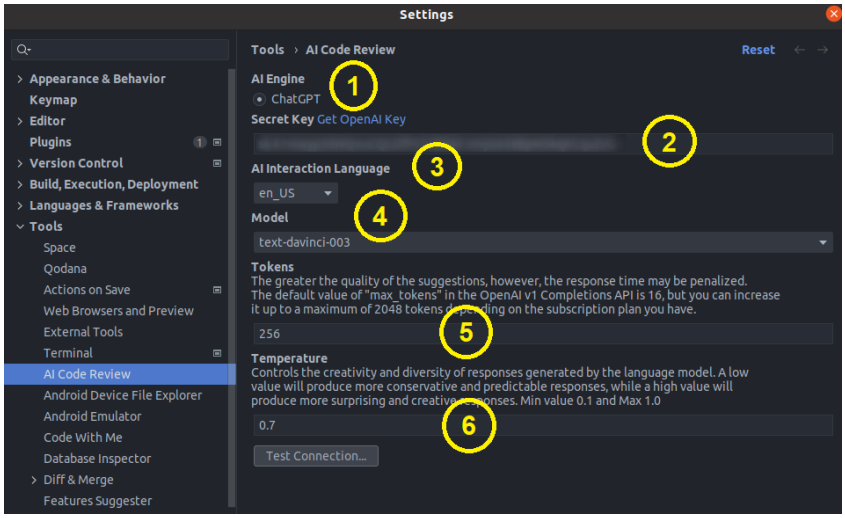

Figura 2. Configuração da Ferramenta.

ChatGPT é a *engine* padrão (Figura 2. - rótulo 1), o que significa que é necessário ter uma OpenAIkey[4] para utilizá-la (Figura 2. - rótulo 2). É possível, no entanto, que outras *engines* sejam adicionadas no futuro devido a detalhes da arquitetura da ferramenta. Após a escolha da *engine*, é necessário definir o idioma de interação (Figura 2. - rótulo 3). A ferramenta suporta três idiomas: português, inglês e espanhol. Isso é útil para garantir que as sugestões de código sejam apresentadas na linguagem mais apropriada para o desenvolvedor.

Em seguida, é necessário escolher um modelo de geração de texto (Figura 2. - rótulo 4). Existem alguns modelos disponíveis, cada um com suas próprias vantagens e desvantagens. Por exemplo, o modelo *text-davinci-003* é considerado um dos mais promissores para a geração de texto de recomendação de alta qualidade. A configuração dos *tokens* também é importante (Figura 2. - rótulo 5). O valor padrão é 256, mas é possível aumentá-lo até o limite de 2048. Quanto maior o valor, maior será o tempo de resposta da *engine*. No entanto, isso pode resultar em sugestões de código mais precisas e detalhadas. Por fim, a temperatura é um parâmetro que necessita ser configurado com valores entre 0.1 e 1.0 (Figura 2. - rótulo 6). Quanto menor o valor, mais conservadoras e previsíveis serão as sugestões de código geradas. Quanto maior o valor, mais criativas e surpreendentes podem ser as sugestões. É importante encontrar um equilíbrio entre a previsibilidade e a criatividade, para que as sugestões de código sejam úteis para o desenvolvedor.

5.3.2. Utilização

Após a configuração da ferramenta, sua utilização se dá através de um processo simples composto basicamente por quatro etapas, conforme descrito na Figura 3. Cada uma das etapas serão descritas em detalhes a seguir:

1. *Seleção do trecho de código*: O primeiro passo da utilização da ferramenta *AICodeReview* é selecionar o trecho de código que o desenvolvedor deseja revisar. Isso pode ser feito através do cursor do mouse ou mesmo por meio de atalhos de teclado;
2. *Solicitação da revisão de código*: Após selecionar o trecho de código, o desenvolvedor deve solicitar explicitamente a revisão de código. Isso pode ser feito através de um atalho de teclado ou de um botão específico na interface da ferramenta;

[4]https://platform.openai.com/account/api-keys

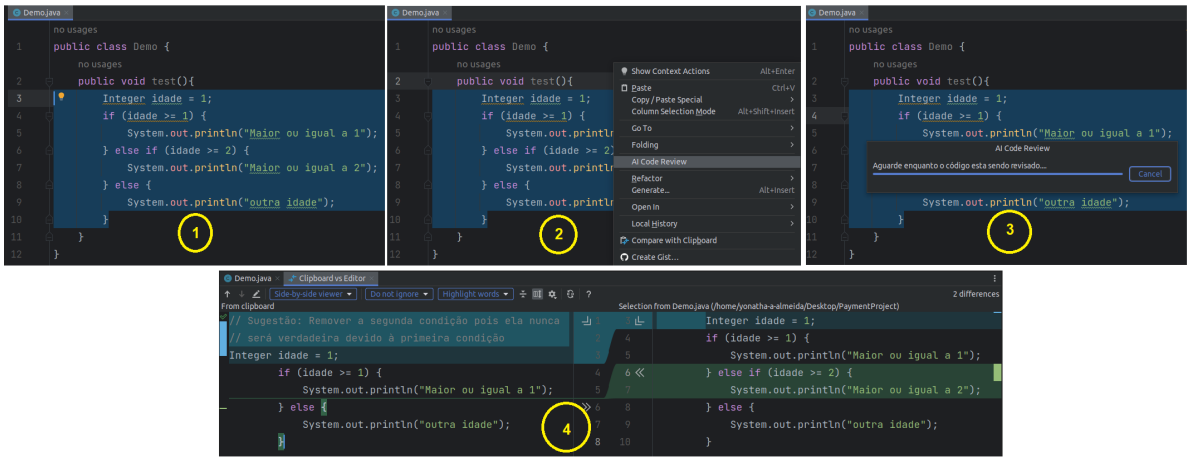

Figura 3. Etapas de Funcionamento da Ferramenta.

3. *Revisão de código*: Uma vez que a revisão de código é solicitada, o *AICodeReview* inicia o processo de análise do código selecionado de acordo com os parâmetros definidos durante a configuração. Essa análise normalmente demanda poucos segundos, dependendo da complexidade do código e das configurações previamente definidas;
4. *Apresentação dos resultados da revisão*: Após a conclusão da análise, a ferramenta *AICodeReview* apresenta o resultado da revisão de código na interface do usuário. Esse resultado inclui uma sugestão específica para melhorar o trecho de código, bem como uma explicação detalhada do raciocínio envolvido para fornecimento da sugestão. Isso ajuda o desenvolvedor a entender não apenas o que precisa ser melhorado no código-fonte, mas também as razões e motivações envolvidas diretamente com a sugestão.

Em resumo, o *AICodeReview* apresenta suas etapas de configuração e utilização de modo simplificado e facilitado. Cabe comentário de que a ferramenta só realiza revisão de código mediante solicitação explícita do desenvolvedor, o que minimiza a possibilidade de intrusão na atividade de programação. O uso da ferramenta simplifica o processo de revisão de código, oferecendo sugestões personalizadas e explicando o raciocínio por trás dessas sugestões. Finalmente, mediante o uso da ferramenta, os desenvolvedores podem economizar tempo e esforço em revisões de código, contribuindo para melhoria da qualidade do seu código.

6. Considerações Finais

Neste artigo, apresentou-se o desenvolvimento de um *plugin* para o IntelliJ IDEA que utiliza a tecnologia emergente do ChatGPT para realizar a revisão de código de forma automatizada. O objetivo deste trabalho foi a propositura de uma ferramenta para simplificar e aprimorar o processo de revisão de código. Como resultado, demonstrou-se que a ferramenta *AICodeReview* é capaz de analisar e entender a estrutura e a semântica do código, identificando problemas de sintaxe, erros de lógica e vulnerabilidades de segurança. Além disso, a ferramenta está apta a fornecer sugestões para soluções e melhorias no código, simplificando o processo de revisão e garantindo um resultado mais eficiente para esta atividade.

Conclui-se que a utilização de ferramentas de IA para revisão de código pode ser uma maneira de melhorar a qualidade e a segurança do software, além de aumentar a produtividade da equipe de desenvolvimento. O ChatGPT e outras tecnologias de IA no mercado têm o potencial de tornar o processo de revisão de código mais preciso, rápido e eficiente. Como possibilidades de trabalhos futuros, destacamos a incorporação de outras tecnologias de IA para aprimorar a análise do código, como redes neurais e algoritmos genéticos. Além disso,

pode-se explorar a integração do *plugin* com outras ferramentas de análise de código, como SonarQube e PMD, para ampliar a cobertura de detecção de problemas. Outra possibilidade é a análise de dados gerados pelos usuários para o aprimoramento contínuo do modelo de IA, tornando-o cada vez mais preciso e eficiente. Finalmente, pretende-se realizar avaliações empíricas com desenvolvedores a fim de validar algumas hipóteses a respeito da utilidade, confiabilidade e eficiência da ferramenta no contexto da revisão de código.

Referências

Albuquerque, D., Guimarães, E., Braga, A., Perkusich, M., Almeida, H., and Perkusich, A. (2022a). Empirical assessment on interactive detection of code smells. In *2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6. IEEE.

Albuquerque, D., Guimaraes, E., Perkusich, M., Almeida, H., and Perkusich, A. (2022b). Concad: A tool for interactive detection of code anomalies. In *Anais do X Workshop de Visualização, Evolução e Manutenção de Software*, pages 31–35. SBC.

Albuquerque, D., Guimarães, E., Tonin, G., Rodriguez, P., Perkusich, M., Almeida, H., Perkusich, A., and Chagas, F. (2022c). Managing technical debt using intelligent techniques-a systematic mapping study. *IEEE Transactions on Software Engineering*.

Arora, S., Ge, R., and Ma, T. (2021). A survey of ai for code review. *Communications of the ACM*, 64(3):64–71.

Bacchelli, A. and Bird, C. (2013). Review practices in software development. *IEEE Software*, 30(5):96–96.

Breault, J. (2017). *Mastering IntelliJ IDEA*. Packt Publishing.

Chen, T., Huang, L., Wang, X., Peng, X., and Zhou, P. (2018). A review of machine learning applications in software development. *Journal of Computer Science and Technology*, 33(4):545–563.

Gupta, A., Jain, H., Agrawal, D., and Vyas, O. (2020). Evaluating chatbot performance in software development peer review. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

Hartmann, M. (2020). Intellij idea: A powerful and extensible ide for jvm-based development. In *Proceedings of the 14th European Conference on Software Architecture: Companion Proceedings*.

Jiang, L., Misherghi, G., Su, Z., and Glondu, S. (2011). Automatic code review using a rule-based approach. In *2011 IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 449–452. IEEE.

Li, C. and Li, X. (2018). A machine learning approach to software code review. *Information and Software Technology*, 99:24–34.

Nagappan, N., Vouk, M., and Sitaraman, R. (2014). Code review guided by static analysis and machine learning. *IEEE Software*, 31(2):64–71.

Rigby, P. C. (2014). Peer reviews in software: A practical guide. *IEEE Software*, 31(3):74–81.

Sadowski, C. and Grechanik, M. (2016). Improving code review quality using machine learning. *IEEE Software*, 33(3):56–63.

Singh, N. K. and Kumar, R. (2019). Automated code review using machine learning techniques: A systematic review. *Journal of Systems and Software*, 157:110396.

Tian, Y., Zhang, W., and Chen, T. (2020). A survey on artificial intelligence techniques for software development. *Journal of Systems and Software*, 161:110464.

Tizard, J. and Liu, K. (2018). A review of automated code review approaches. *Journal of Systems and Software*, 144:306–325.

Wang, P., Liu, X., and Zhang, H. (2018). Deep code review. *IEEE Software*, 35(2):34–40.

Zhang, H., Liao, S., and Wang, T. (2018). Deep code comment generation. In *Proceedings of the 2018 International Conference on Software Engineering: Software Engineering in Practice*, pages 56–63. ACM.