Integrando LLMs Locais no Visual Studio Code para Test Driven Development

Ana Clara Lannes, David Tadokoro, Gustavo Rodrigues, Gustavo Serra e Lucas Rosa

Instituto de Matemática e Estatística da Universidade de São Paulo

Introdução

MOTIVAÇÃO

A **Inteligência Artificial (IA)** está transformando a engenharia de software, especialmente no desenvolvimento e teste de sistemas, ao automatizar tarefas repetitivas e acelerar processos. Modelos generativos, como ChatGPT e Gemini, auxiliam na escrita de testes e refatoração, mas enfrentam desafios como dependência de servidores remotos e restrições de segurança e privacidade.

Já **soluções open-source**, como LLaMA (Meta) e DeepSeek, permitem a execução local de IA, garantindo maior autonomia e segurança, essenciais para setores regulamentados e projetos com dados sensíveis.

O **Test Driven Development (TDD)** segue como uma prática fundamental para a qualidade do software, promovendo um design modular e reduzindo bugs. Sua adoção, porém, pode ser desafiadora, especialmente sob prazos apertados. A IA generativa pode facilitar esse processo ao automatizar partes do TDD, permitindo que os desenvolvedores foquem em tarefas mais estratégicas.

OBJETIVO

Este projeto tem como objetivo adaptar uma ferramenta de IA generativa para suporte ao TDD, originalmente apresentada pelo professor Jorge Melegati na disciplina MAC5913, para funcionar localmente. A proposta é integrá-la ao **Visual Studio Code (VS Code)**, um dos editores de código mais populares, permitindo que desenvolvedores interajam diretamente com a IA dentro do próprio ambiente de trabalho, sem depender de servidores externos.

Além de auxiliar na escrita de testes e na geração de código, a ferramenta aproveita técnicas apresentadas no curso como engenharia de prompts para aumentar a eficiência da IA na criação incremental de código e na refatoração de soluções existentes. Isso torna o processo de desenvolvimento mais ágil e iterativo, reduzindo erros e incentivando boas práticas de programação. A solução também permite que os usuários escolham entre diferentes modelos de IA, como LLaMA e DeepSeek, para melhor atender às suas necessidades e requisitos de hardware.

Ao trazer a IA para o ambiente local, o projeto busca oferecer mais controle, segurança e privacidade, eliminando a dependência de APIs externas e minimizando riscos de vazamento de dados. Além disso, a integração direta com o VS Code garante uma experiência fluida, sem interrupções no fluxo de trabalho ou a necessidade de alternar entre diferentes ferramentas e plataformas.

Metodologia

IMPLEMENTAÇÃO

Um protótipo foi desenvolvido em **TypeScript**, utilizando a API do VS Code para criar botões e interfaces interativas. Além disso, o código original, escrito em Python, foi adaptado para o TypeScript. Ela permite que os desenvolvedores usem uma IA local para auxiliar no desenvolvimento orientado a testes, sem depender de conexões com servidores externos.

FUNCIONALIDADES IDEALIZADAS

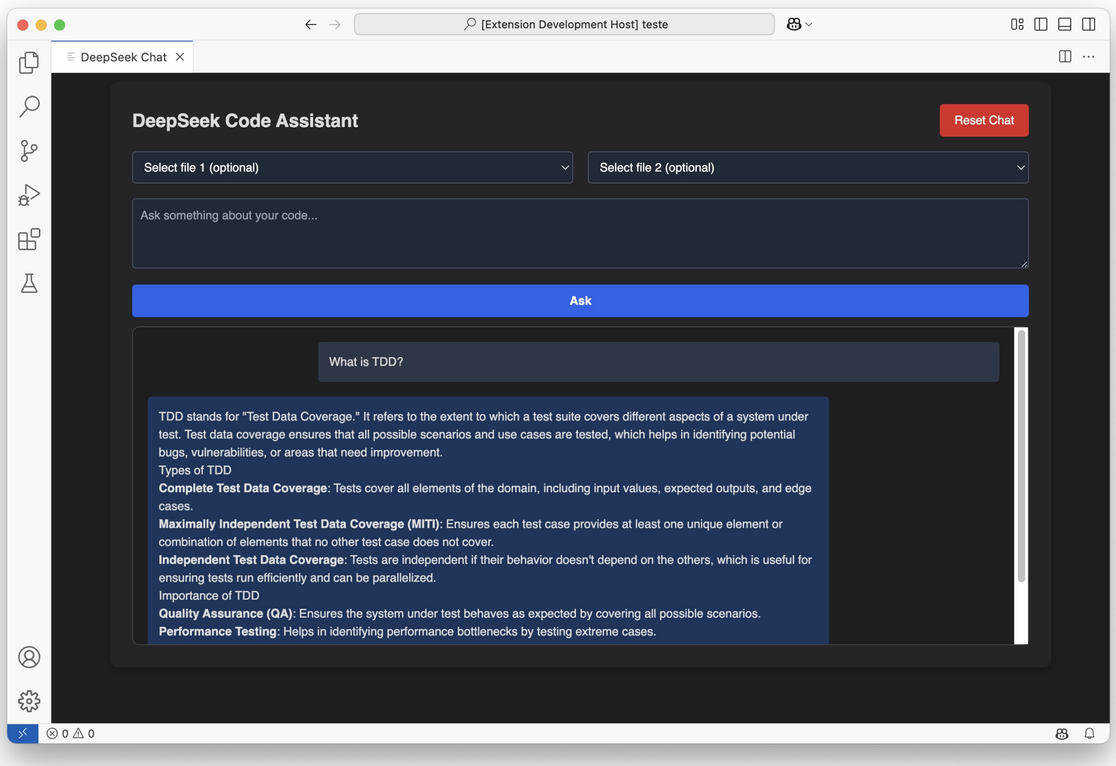
- **Chat integrado**: Interface gráfica com um chat embutido diretamente no VS Code;
- **LLMs locais**: LLMs são executadas na máquina do desenvolvedor;
- **Seleção de arquivos**: Identificação e seleção de arquivos relevantes dentro do ambiente de desenvolvimento;
- **TDD automatizado**: Implementação e refatoração de código com base no conceito de TDD;
- **Suporte a Refactor**: Ajuda nas três etapas principais do TDD.

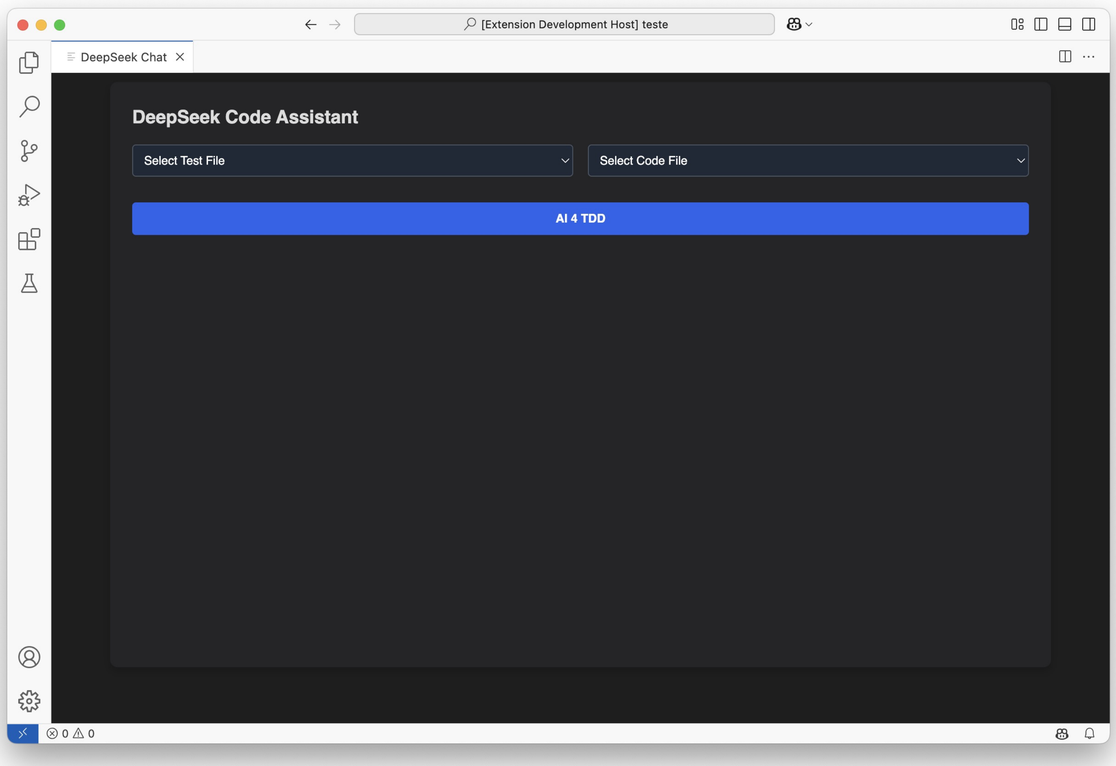
Protótipo Desenvolvido

INTERFACE DA FERRAMENTA

Dado o escopo e limitações de tempo do projeto, desenvolvemos um protótipo incial, mas funcional. A ferramenta apresenta um chat integrado ao VS Code, permitindo a seleção de arquivos e a interação dinâmica com a IA. Além disso, uma segunda janela permite configurar os arquivos usados nas etapas de TDD (ver figuras abaixo). O sistema é uma evolução de alguns scripts apresentados na disciplina MAC5913, com melhorias que tornam a experiência mais integrada e produtiva. A implementação está disponível no repositório GitHub através do link `https://github.com/gustavohenriquesr/deepseek-ollama-extension`.

O protótipo atual ainda apresenta algumas limitações. Entre elas: o mecanismo de refatoração presente no código original não foi implementado; não há notificações ou saídas explícitas sobre a execução do código, sendo necessário verificar a janela de depuração; além disso, a janela não permite personalização do tema.





Conclusões

A solução proposta aprimora o desenvolvimento baseado em TDD ao integrar uma ferramenta de linha de comando diretamente ao editor de código mais popular da atualidade. Além disso, estende a implementação original ao permitir o uso de modelos de IA locais, proporcionando mais autonomia e controle no processo. Isso elimina a dependência de APIs externas e reduz preocupações com privacidade e segurança.

Os próximos passos envolvem superar as limitações identificadas, aprimorando a interação entre o chat e a ferramenta de TDD. O objetivo seria oferecer sugestões contextuais durante a conversa com a IA, tornando o processo mais dinâmico. Com isso, a IA poderá recomendar modificações no código alinhadas à metodologia TDD, além de sugerir padrões de refatoração e otimizações de desempenho.

Outra área de melhoria é a implementação de técnicas avançadas de engenharia de prompts, elevando a qualidade das respostas geradas pela IA. Isso garantirá maior coerência e precisão na automação do desenvolvimento.

Referências

[1] Beck, K. (2022). Test Driven Development: By Example. Addison-Wesley.

[2] Melegati, J. (2023). Software Maintenance and Evolution 2022-23: AI for Software Engineering. Material de aula.

[3] Mock, M., Melegati, J., & Russo, B. (2025). Generative AI for Test Driven Development: Preliminary Results. AI for Agile Software Engineering Workshop @XP2024. Disponível em: Springer.

[4] Riemenschneider, C.K., Hardgrave, B.C., & Davis, F.D. (2002). Explaining software developer acceptance of methodologies: a comparison of five theoretical models. IEEE Transactions on Software Engineering.