AuditAI: Automatizando e Facilitando a Auditoria de Contratos Inteligentes com Relatórios Contextuais Gerados por IA

Lívia Carrera[1], **Ramón Cordeiro**[2], **Antônio Abelém**[1]

[1] Universidade Federal do Pará (UFPA)
Belém – PA – Brasil

[2]Delft University of Technology (TU Delft)
Delft, Netherlands

lixpluv@proton.me, r.ramoncordeiro@tudelft.nl, abelem@ufpa.br

Abstract. *Smart contract security remains a major challenge in blockchain development, often limited by manual audits and static analysis tools.This paper introduces AuditAI, a platform that combines the Slither static analyzer with Large Language Models (LLMs) served via Groq Artificial Intelligence (AI) to detect vulnerabilities and generate contextualized, customizable audit reports. By merging formal analysis with natural language interpretation, AuditAI enhances the clarity and accessibility of security assessments. Preliminary results indicate improvements, in both the efficiency and interpretability of smart contract audits, while acknowledging the persistent challenges in automated vulnerability detection.*

Resumo. *A segurança de contratos inteligentes continua sendo um desafio significativo no desenvolvimento blockchain, frequentemente limitada por auditorias manuais e ferramentas de análise estática. Este artigo apresenta o AuditAI, uma plataforma que combina o analisador estático Slither com modelos de linguagem de grande escala (Large Language Models - LLMs) servidos por meio da Inteligência Artificial (IA) Groq para detectar vulnerabilidades e gerar relatórios de auditoria contextualizados e personalizáveis. Ao unir análise formal com interpretação em linguagem natural, o AuditAI aprimora a clareza e acessibilidade das avaliações de segurança. Resultados preliminares indicam melhorias, tanto na eficiência quanto na interpretabilidade das auditorias de contratos inteligentes, reconhecendo os desafios persistentes na detecção automatizada de vulnerabilidades.*

1. Introdução

Oferecendo uma plataforma descentralizada, transparente e imutável para aplicações que vão muito além das transações com criptomoedas, tecnologia *blockchain* transformou fundamentalmente a infraestrutura digital. No centro desse novo paradigma tecnológico estão os contratos inteligentes — programas autoexecutáveis que automatizam acordos e transações sem a necessidade de intermediários. A *blockchain* Ethereum, pioneira nesse

campo, introduziu a *Ethereum Virtual Machine (EVM)* como ambiente de execução para esses contratos, sendo o Solidity a linguagem de programação primária e mais amplamente adotada para o desenvolvimento de contratos inteligentes.

O destaque do Solidity deve-se às suas características estratégicas de design, incluindo uma sintaxe semelhante à do JavaScript, o que facilita sua adoção por desenvolvedores *web*, além de uma arquitetura especializada para a EVM. Diferentemente de outras linguagens de programação utilizadas em *blockchains*, como Rust (empregada na Solana) ou Go (utilizada na Hyperledger Fabric), o Solidity foi projetado meticulosamente para expressar lógicas complexas de contratos inteligentes, incorporando recursos distintos como suporte a herança, tipos de dados específicos para *blockchain* e mecanismos de segurança integrados.

De forma consistente, desafios significativos estão presentes no desenvolvimento de contratos inteligentes, portanto, têm-se uma proporção substancial das vulnerabilidades de segurança em *blockchains* originárias de falhas lógicas intrínsecas na implementação dos contratos ou da utilização subótima dos paradigmas do Solidity [Jiang et al. 2022, Palladino 2023]. Essas vulnerabilidades podem acarretar perdas financeiras consideráveis, permitindo que agentes maliciosos explorem brechas e comprometam a integridade de protocolos descentralizados.

Apesar da disponibilidade de ferramentas como o *Slither*, as auditorias ainda enfrentam desafios consideráveis. Relatórios produzidos por ferramentas tradicionais costumam ser excessivamente técnicos e difíceis de interpretar, resultando em uma demanda significativa de tempo e esforço por parte dos auditores para compreender e priorizar vulnerabilidades. Em auditorias tradicionais, o processo manual de interpretação desses relatórios técnicos pode levar horas ou dias, dependendo da complexidade do contrato analisado, aumentando o custo e atrasando o ciclo de desenvolvimento ágil, especialmente em equipes com poucos especialistas em segurança *blockchain*.

Neste contexto, este trabalho apresenta o *AuditAI*, uma plataforma híbrida que busca otimizar o processo de auditoria de contratos inteligentes escritos em Solidity por meio da integração entre análise estática tradicional e inteligência artificial generativa. A solução proposta combina a robustez do analisador *Slither* [Feist et al. 2019], amplamente reconhecido na detecção de vulnerabilidades em contratos Solidity, com a capacidade interpretativa de *modelos de linguagem de grande escala (LLMs)* acessados via *Groq AI* [Groq Inc. 2024], permitindo a geração automatizada de relatórios auditáveis, personalizáveis e semanticamente contextualizados. A proposta central é manter a precisão técnica das análises realizadas pelo *Slither*, enquanto melhora substancialmente a interpretabilidade, acessibilidade e praticidade dos resultados. Dessa forma, o tempo gasto por desenvolvedores e auditores na compreensão dos riscos identificados é significativamente reduzido, permitindo uma ação mais rápida e eficaz na correção de vulnerabilidades.

As principais contribuições deste artigo são:

- Desenvolvimento de uma plataforma híbrida que combina análise estática (*Slither*) com inteligência artificial generativa (*Groq AI*) para aprimorar relatórios de auditoria de contratos inteligentes.
- Avaliação qualitativa inicial que comprova a viabilidade técnica da abordagem proposta e evidencia ganhos substanciais em interpretabilidade e usabilidade em

comparação às ferramentas tradicionais.
- Disponibilização da ferramenta *AuditAI* como *software* livre e de código aberto, permitindo contribuição e uso pela comunidade acadêmica e profissional.

A estrutura deste artigo está organizada da seguinte forma: na Seção 2, são discutidos os trabalhos relacionados, delineando o estado da arte em auditoria automatizada de contratos inteligentes. A Seção 3 apresenta o detalhamento da arquitetura do *AuditAI*, seus componentes técnicos e a integração entre os módulos de análise e geração textual. A Seção 4 descreve a implementação prática do sistema, seguida da Seção 5, dedicada à avaliação experimental e à análise crítica dos resultados. Por fim, a Seção 6 sintetiza as conclusões obtidas e propõe direções futuras para o aprimoramento e extensão da ferramenta proposta.

2. Trabalhos Relacionados

O avanço da tecnologia *blockchain* tem impulsionado o desenvolvimento de ferramentas especializadas para auditoria de contratos inteligentes, com ênfase na detecção e mitigação de vulnerabilidades de segurança. Nesse cenário, o *SmartBugs* destacou-se como uma das soluções inovadoras mais adotadas na academia e na indústria [Ferreira et al. 2020], graças à sua arquitetura modular e à integração com diversas ferramentas de análise. A versão 2.0 da ferramenta [di Angelo et al. 2023] introduziu suporte à análise de código-fonte e *bytecode* da *Ethereum Virtual Machine (EVM)*, padronização dos relatórios com base no *SWC Registry* e integração com 19 ferramentas distintas, representando um marco importante em auditoria automatizada. No entanto, trabalhos como o de [Dinis 2022], apontam que cerca de 93% dos contratos analisados foram classificados como vulneráveis, em grande parte devido à detecção de falhas de baixa criticidade, evidenciando um excesso de falsos positivos e a necessidade de aprimoramentos na interpretação e priorização dos achados.

Com o objetivo de mitigar esses problemas, o *AuditAI* propõe uma abordagem híbrida que alia a análise estática realizada pelo *Slither* ao uso de modelos de linguagem de grande escala (*LLMs*) fornecidos via API da *Groq* [Groq Inc. 2024], permitindo a geração de relatórios técnicos contextualizados, personalizáveis e organizados por severidade. Essa integração visa não apenas automatizar o processo de auditoria, mas também torná-lo mais compreensível e acessível, reduzindo a dependência de interpretações manuais extensas. Estudos comparativos como o de Nguyen et al. [Revol et al. 2021] reforçam a relevância dessa proposta ao acentuar que a dificuldade de interpretar resultados, muitas vezes em formatos como *JSON*, compromete a eficiência e a precisão da auditoria. O *AuditAI* busca superar essa limitação ao entregar relatórios semanticamente enriquecidos, prontos para análise e exportação.

Trabalhos recentes como de [David et al. 2023] exploram o uso de *LLMs* em auditorias automáticas, principalmente em contratos *DeFi*. Apesar dos avanços, esses modelos ainda carecem de precisão, reforçando a importância da validação humana. O *AuditAI* responde a esse desafio ao estruturar os achados com suporte de IA, mantendo, entretanto, o especialista como elemento decisivo na validação final. Além disso, a proposta se alinha e expande iniciativas como a de [Dinis 2022], que aplicaram aprendizado de máquina para priorização automática de vulnerabilidades, reduzindo significativamente o tempo de análise.

Embora existam ideias semelhantes na literatura e na *web*, não foram encontrados registros de implementações práticas com a mesma arquitetura e escopo funcional do *AuditAI*. Sua inovação técnica reside na integração de análise formal, geração de linguagem natural e interface interativa em um único sistema voltado especificamente à auditoria de contratos inteligentes.

Adicionalmente, é importante destacar que automação de auditorias e geração automática de relatórios já são práticas consolidadas em áreas tradicionais como segurança da informação, auditorias financeiras e certificações de conformidade. Ferramentas nessas áreas frequentemente geram relatórios automáticos para auditorias baseadas em padrões conhecidos, tais como *ISO/IEC 27001*, *PCI DSS* ou *SOX*. No entanto, essas abordagens tradicionais não lidam diretamente com os desafios únicos associados à auditoria em *blockchains*, como a complexidade da execução determinística de contratos inteligentes, a irreversibilidade das transações e a necessidade crítica de precisão absoluta devido aos potenciais impactos financeiros. O *AuditAI* adapta funcionalidades tradicionais como geração automática de relatórios e avaliação contextualizada para resolver esses desafios específicos do ambiente *blockchain*, destacando-se pela integração única de ferramentas de análise estática com inteligência artificial generativa para fornecer relatórios interpretáveis. Ademais, a Tabela 1 apresenta uma breve comparação dos trabalhos e suas características em relação ao AuditAI.

Tabela 1. Comparação entre o *AuditAI* e trabalhos relacionados

Trabalho	Análise Estática	Geração de Relatórios	Contextualização via IA	Interface Interativa	Exportação (PDF/JSON)	Personalização Textual
SmartBugs	✓	Parcial	✗	✗	✓	✗
SmartBugs 2.0	✓	✓	✗	✗	✓	✗
Nguyen et al.	✓	Parcial	✗	✗	✗	✗
Manual Audit [David et al. 2023]	✗	Parcial	✓	✗	✗	✓
AuditAI	✓	✓	✓	✓ (Chatbot)	✓	✓

3. Arquitetura e Funcionamento do AuditAI

Este trabalho propõe o desenvolvimento do *AuditAI*, uma plataforma experimental e de código aberto voltada à melhoria do processo de auditoria de contratos inteligentes escritos em Solidity. A proposta busca unir análise estática automatizada com recursos de inteligência artificial generativa, de modo a tornar os relatórios de vulnerabilidade mais compreensíveis, organizados e prontos para uso por desenvolvedores e auditores. A metodologia adotada envolveu a construção de um protótipo funcional dividido em módulos distintos e a definição de um fluxo de execução claro, desde a submissão do contrato até a geração do relatório final. Informações adicionais sobre o projeto, incluindo o link para o repositório *GitHub*, estão disponíveis no Apêndice A.

3.1. Arquitetura Geral

O *AuditAI* é estruturado em três módulos principais: **interface *web***, **servidor *backend*** e **motor de análise em ambiente isolado**. A interface *web* foi desenvolvida com o *framework Svelte*, utilizando o *Vite* para otimização de performance e experiência do usuário. Essa interface permite o envio de arquivos `.sol`, a interação com o *chatbot* para ajustes nos textos gerados e a visualização dos relatórios. O *backend*, implementado com *FastAPI*, é responsável por orquestrar o fluxo de dados entre os módulos, realizar o pré e pós-processamento dos resultados e gerenciar a comunicação com a API externa da

Groq. O motor de análise está encapsulado em um contêiner *Docker*, que executa a ferramenta *Slither* para detectar vulnerabilidades estáticas diretamente no código Solidity. A utilização do *Docker* garante isolamento completo, reprodutibilidade e segurança nas análises. A Figura 1 ilustra a visão geral da ferramenta *AuditAI*.

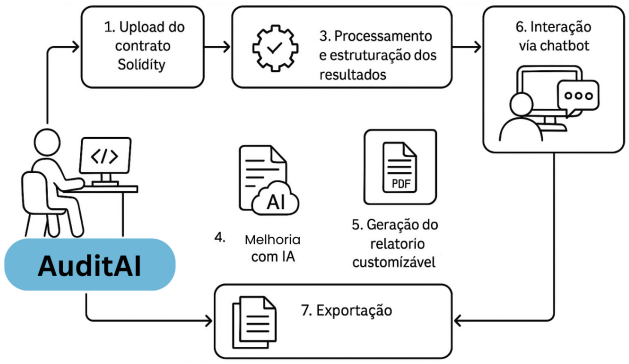

Figura 1. Visão Geral do uso da ferramenta *AuditAI*, desde o *upload* do contrato até a exportação e personalização do relatório via IA.

3.2. Fluxo de Análise

O processo inicia-se com o *upload* de um contrato inteligente via interface *web*. O arquivo é então enviado ao *backend*, que o repassa para o contêiner onde o *Slither* é executado. A ferramenta retorna uma análise técnica contendo as vulnerabilidades encontradas, incluindo informações como categoria, severidade, linha de código afetada e descrição bruta do problema. Esses dados são organizados em um arquivo intermediário no formato *JSON*, que serve como base para a geração dos relatórios.

Em seguida, esse *JSON* é enviado para a API da *Groq*, que utiliza modelos de linguagem de grande escala (*LLMs*) para reescrever as descrições técnicas de forma mais acessível. O modelo também pode adicionar explicações, contextualizações e sugestões, sempre com o objetivo de tornar o relatório mais útil e informativo para desenvolvedores com diferentes níveis de familiaridade com segurança em *blockchain*. O sistema ainda permite que o usuário interaja com a IA por meio de um *chatbot* integrado, fornecendo comandos para personalizar o conteúdo textual do relatório, como modificar o título, solicitar um resumo ou explicações adicionais.

A Figura 2 ilustra o fluxo completo da ferramenta *AuditAI*, desde o envio do contrato inteligente até a exportação e personalização do relatório gerado com suporte de inteligência artificial.

O processo inicia-se da seguinte forma:

- **(1)** O usuário realiza o *upload* do arquivo *Upload.sol* por meio da interface *web*, juntamente com a seleção das ferramentas de análise desejadas.
- **(2)** O *backend* recebe o arquivo e as opções selecionadas, encaminhando-os ao contêiner *Docker* responsável pela execução da análise.

- **(3)** Dentro do contêiner, o *Slither* é executado para realizar uma análise estática no contrato inteligente.
- **(4)** O *Slither* retorna um arquivo no formato *JSON* contendo os resultados da análise, incluindo vulnerabilidades encontradas, nível de severidade, categorias associadas e trechos de código afetados.
- **(5)** O *backend* envia o *JSON* com os dados da análise para a API da *Groq*, juntamente com o contexto necessário para a geração textual.
- **(6)** A *Groq* retorna um relatório redigido em linguagem acessível, incorporando explicações, contextualizações e sugestões de melhorias para facilitar a compreensão, inclusive para desenvolvedores menos familiarizados com segurança em *blockchain*.
- **(7)** O sistema salva o arquivo *latest_formatted.json* com o conteúdo formatado e gera um relatório em formato *PDF* preliminar.
- **(8)** O usuário pode realizar o download do *JSON* intermediário caso deseje examinar os dados diretamente.
- **(9)** Na interface de *chatbot*, o usuário interage com a IA, fornecendo comandos personalizados, como ajustes de título, adição de resumos ou reformulações de seções do relatório.
- **(10)** Após a interação, o usuário solicita a atualização do relatório clicando em "Atualizar Preview".
- **(11)** As instruções fornecidas são aplicadas ao *JSON* formatado para atualizar o conteúdo textual do relatório.
- **(12)** Um novo *PDF* é gerado com base nas modificações aplicadas.
- **(13)** O usuário pode, então, solicitar o *preview* final e realizar o download do relatório completo em *PDF*.

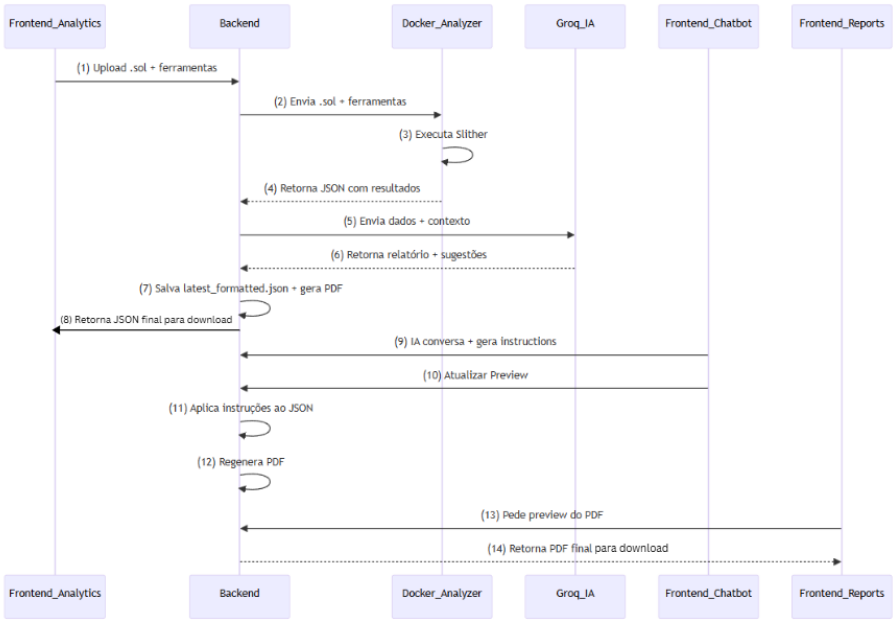

Figura 2. Fluxo de análise da ferramenta *AuditAI*, desde o *upload* do contrato até a exportação e personalização do relatório via IA.

3.3. Geração e Visualização de Relatórios

Após o processamento pela IA, o conteúdo textual é inserido em uma estrutura de relatório que pode ser exportada em diferentes formatos, como *PDF*. O sistema utiliza um modelo fixo de *layout*, garantindo consistência visual e organizacional entre diferentes análises. Os relatórios apresentam as vulnerabilidades agrupadas por severidade e organizadas em seções claras, com objetivo de facilitar tanto a revisão técnica quanto a comunicação com equipes de desenvolvimento.

A aba *Reports* da interface permite visualizar os relatórios gerados, enquanto a aba *Chatbot* oferece uma interface para personalização textual e esclarecimentos pontuais. A aba *Analytics* centraliza o envio e execução de novas análises. Já a aba *Dashboard*, ainda em construção, será destinada a visualizações estatísticas e acompanhamento histórico das auditorias realizadas.

A Figura 3 mostra a interface do AuditAI, exemplificando as abas principais: *Analytics*, *Chatbot*, *Dashboard* e a tela inicial.

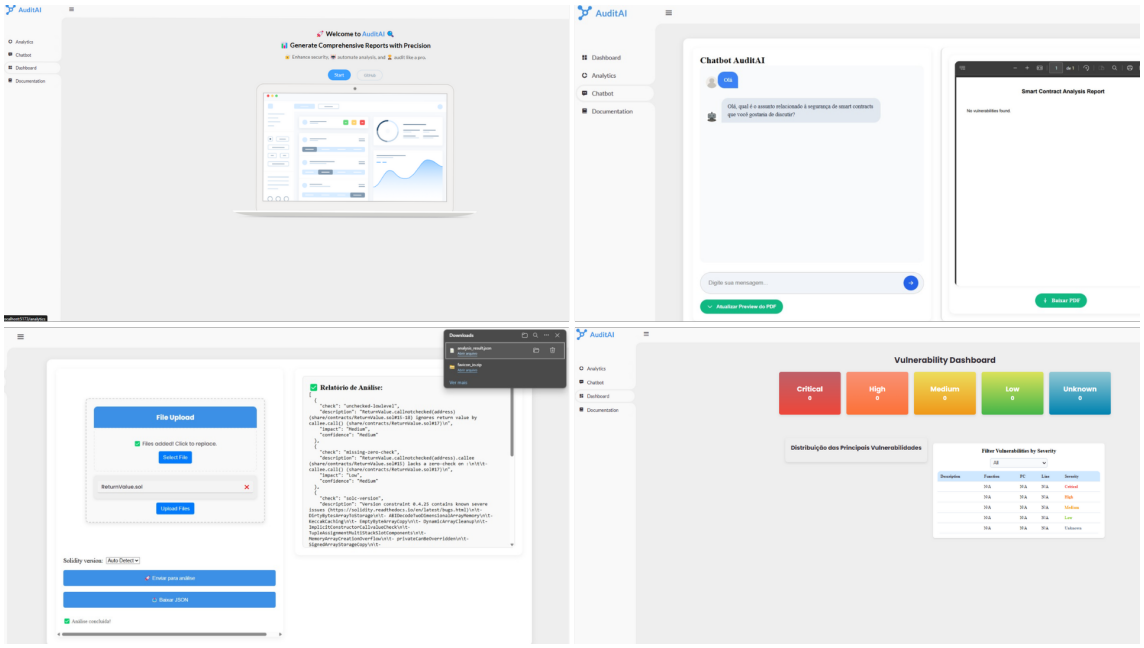

Figura 3. Interface do *AuditAI*, com exemplos das abas *Analytics*, *Chatbot*, *Dashboard* e tela inicial.

4. Implementação

O desenvolvimento do projeto foi conduzido com o objetivo de construir uma plataforma leve, modular e de fácil manutenção, que pudesse ser executada localmente e adaptada conforme as necessidades dos usuários. Esta seção descreve as principais tecnologias adotadas, a estrutura do projeto e os desafios enfrentados durante a implementação do protótipo funcional.

4.1. Tecnologias Utilizadas

O *frontend* da plataforma foi desenvolvido utilizando o *framework Svelte* em conjunto com o *Vite*, proporcionando uma aplicação reativa, com baixo tempo de carregamento

e boa experiência de usuário. A escolha do *Svelte* justifica-se por sua simplicidade de sintaxe, ausência de dependências pesadas e excelente performance na manipulação de estados, aspectos relevantes para o fluxo dinâmico de uma ferramenta de auditoria interativa. Enquanto isso, o *backend* foi construído com *FastAPI*, um *framework Python* moderno, leve e performático. O uso do *FastAPI* permitiu a definição clara de rotas assíncronas, integração com validação de tipos e documentação automática da API. Toda a comunicação com a ferramenta de análise estática e com a API da *Groq* é intermediada por esse *backend*, garantindo isolamento entre a lógica de apresentação e a lógica de negócio.

A análise de vulnerabilidades é realizada com o *Slither*, ferramenta de análise estática amplamente utilizada para contratos inteligentes em *Solidity*. Sua adoção se deve à maturidade da ferramenta, à documentação consistente e à ampla aceitação tanto na comunidade acadêmica quanto no setor de segurança *blockchain*. O *Slither* é executado dentro de um contêiner *Docker*, configurado com a versão adequada do compilador *Solidity* e suas dependências.

4.2. Estrutura do Projeto

O projeto foi organizado em três núcleos principais, refletindo a arquitetura modular da plataforma:

- `frontend/` – contém a aplicação *Svelte*, incluindo os componentes visuais e páginas principais: *Analytics*, *Chatbot*, *Reports* e *Dashboard*.
- `backend/` – aplicação *FastAPI* que orquestra as requisições, gerencia o envio ao analisador e à *Groq AI*, e realiza a renderização dos relatórios.
- `analyzer/` – diretório contendo a configuração *Docker* do ambiente que executa o *Slither* com isolamento completo.

Além disso, há pastas auxiliares para *scripts* de transformação e arquivos estáticos usados na geração e exportação dos documentos.

4.3. Desafios e Soluções Adotadas

Durante a construção do protótipo, diversas dificuldades técnicas foram identificadas. Um dos primeiros obstáculos foi configurar corretamente o ambiente de execução do *Slither*, que exige compatibilidade precisa com versões do compilador *Solidity* e bibliotecas específicas. A adoção do *Docker* como solução de empacotamento permitiu isolar esse ambiente, garantindo reprodutibilidade e estabilidade. Além disso, o encapsulamento em contêiner agrega um fator de segurança e confiabilidade, uma vez que o processo de análise é executado de forma determinística e não pode ser manipulado por processos externos. Essa abordagem reforça a veracidade dos resultados — aspecto crítico em auditorias de segurança.

Outro desafio significativo foi o *parsing* da saída *JSON* gerada pelo *Slither*, que possui estrutura extensa e sem formatação *human-readable* (facilmente legível por humanos). Para contornar isso, foi desenvolvida uma camada intermediária de pré-processamento que organiza os dados por severidade, linha de código, categoria *SWC* e descrição técnica simplificada. Essa camada é essencial para alimentar o modelo de linguagem com informações organizadas e compreensíveis.

A comunicação com a API da *Groq* também exigiu cuidados específicos, especialmente no controle de formato e conteúdo das requisições enviadas. Foi necessário realizar testes iterativos para obter respostas consistentes, claras e adaptáveis à estrutura dos relatórios, respeitando os limites de *tokens* e formatação do modelo. Na geração dos relatórios em *PDF* e *JSON*, exigiu-se a criação de *templates* flexíveis, capazes de suportar personalizações e diferentes estilos de apresentação.

Ademais, embora o *AuditAI* utilize ferramentas prontas em seu núcleo — como o *Slither* para análise estática e *LLMs* acessadas via *Groq* — a integração entre essas tecnologias revelou-se um desafio não trivial. Trabalhar com componentes já existentes exige compreender seus parâmetros internos, formatos de entrada e saída, limitações e fluxos de execução. Fazer com que esses módulos distintos se comuniquem de forma fluida, mantendo coerência entre seus resultados e garantindo uma experiência equilibrada ao usuário final, exigiu ajustes finos, reestruturações e testes contínuos. Essa etapa foi crucial para que a plataforma funcionasse como um sistema coeso, e não apenas como a soma de ferramentas independentes.

5. Resultados

Os resultados obtidos com a implementação do *AuditAI* indicam que a proposta é tecnicamente viável e funcional no contexto de auditoria estática de contratos inteligentes. Embora ainda em estágio inicial, a ferramenta apresenta um fluxo operacional completo, com análise automatizada, geração de relatórios e personalização textual assistida por inteligência artificial. As funcionalidades já implementadas demonstram a efetividade da arquitetura proposta e o potencial da plataforma como recurso complementar para o desenvolvimento seguro de contratos *Solidity*.

5.1. Validação Funcional

A validação funcional foi conduzida utilizando um *dataset* curado, denominado *smartbugs-curated*[1], um grande suporte adotado na literatura de segurança em *blockchain*. Um dos contratos destacados nessa validação foi o *parity_wallet_bug_1.sol*, que possui sua vulnerabilidade historicamente conhecida e documentada como podemos ver de acordo com [Zeppelin 2017], sendo utilizado frequentemente para testes de ferramentas de segurança. O contrato foi submetido à plataforma *AuditAI*, seguindo o fluxo completo: *upload* via interface *web*, análise estática com *Slither*, formatação automática dos achados e geração de relatório técnico em *PDF*.

O código utilizado nos testes pode ser visto no Anexo B.

Os resultados obtidos foram consistentes com o esperado. Vulnerabilidades conhecidas, como problemas de reentrância e chamadas inseguras, foram detectadas corretamente pelo *Slither*. Em seguida, o conteúdo técnico foi interpretado pelo modelo *LLM* da *Groq*, resultando em descrições contextualizadas e compreensíveis para usuários sem formação avançada em segurança. A Figura 4 e a Figura 5 ilustram exemplos práticos da plataforma em uso.

[1] `https://github.com/smartbugs/smartbugs-curated`

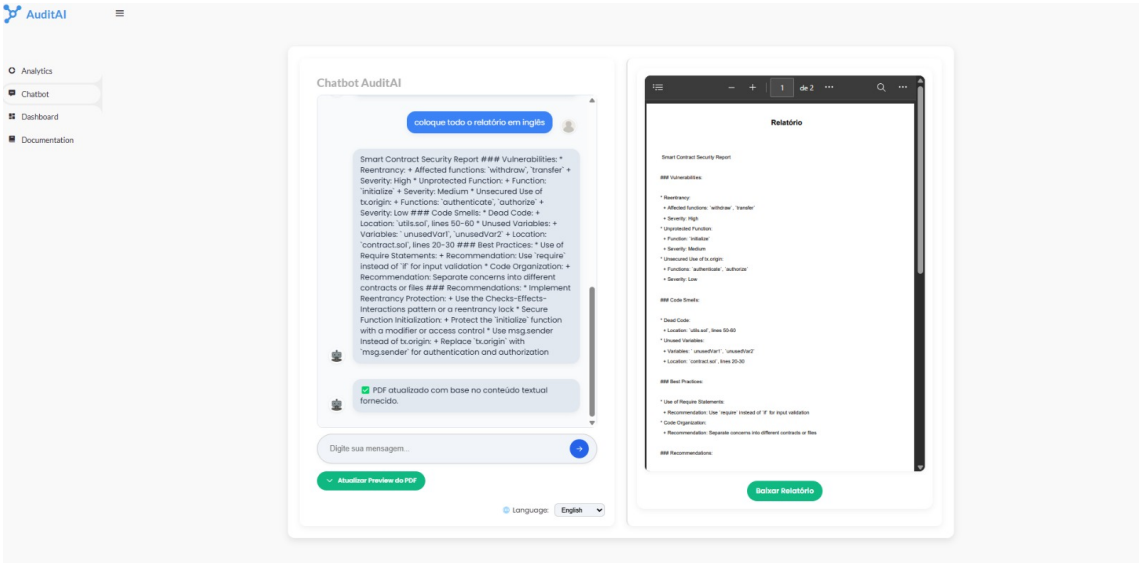

Figura 4. Interface *Analytics* exibindo o relatório de análise do contrato *parity_wallet_bug_1.sol*.

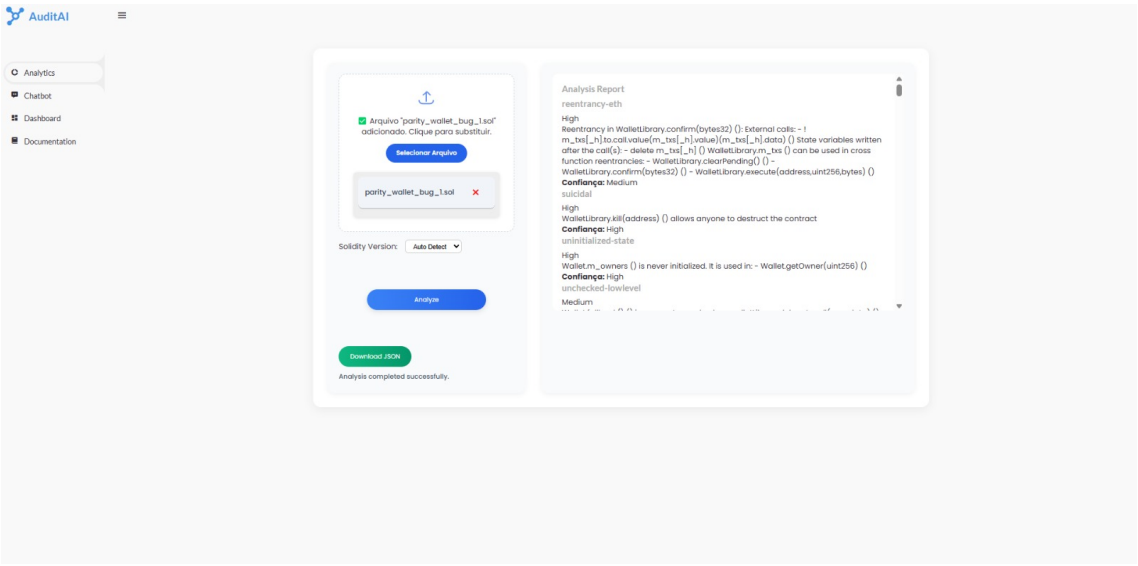

Figura 5. Interface *Chatbot* mostrando o relatório traduzido e personalizado via IA em tempo real.

5.2. Interação com IA e Personalização

Um destaque significativo do sistema é a possibilidade de personalização textual via *chatbot*. Os usuários podem solicitar reformulações, mudanças de idioma, ajustes nos níveis de severidade ou explicações adicionais utilizando linguagem natural. Essas instruções são processadas pela IA e refletidas imediatamente nas versões atualizadas dos relatórios. Durante os testes realizados, o *chatbot* demonstrou alta capacidade de adaptação textual, mantendo clareza e coerência sem comprometer a precisão técnica.

5.3. Ganhos Percebidos

Para ilustrar quantitativamente os ganhos proporcionados pelo *AuditAI*, foram realizadas medições específicas durante os testes experimentais:

- Tempo médio para análise e geração inicial do relatório: 45 segundos;
- Tempo médio para reformulação textual do relatório via IA: 20 segundos (dependendo do grau de complexidade da requisição);
- Redução estimada de tempo em comparação à geração manual de relatórios semelhantes por auditores humanos: cerca de 80%.

Além disso, a Tabela 2 apresenta uma comparação qualitativa detalhada entre o uso direto do *Slither* e a utilização da plataforma desenvolvida.

Tabela 2. Comparação qualitativa entre *Slither* e *AuditAI*

Critério	***Slither***	***AuditAI***
Requer conhecimento técnico	Alto	Baixo
human-readable	✗	✓
Personalização textual	✗	✓ (via IA)
Exportação automática (PDF)	✗	✓
Interface gráfica	✗	✓
Explicações contextualizadas	✗	✓

5.4. Limitações Atuais

Apesar dos resultados positivos, a versão atual do sistema possui limitações como: suporte exclusivo a contratos escritos em *Solidity*; restrição à análise individual de contratos; dependência atual de conexão com a API externa da *Groq* para funcionamento da IA; e ausência de armazenamento persistente dos relatórios gerados. Adicionalmente, ainda não há suporte para execução em lote, histórico detalhado de versões, nem integração robusta com banco de dados. Essas limitações serão priorizadas nas próximas etapas do desenvolvimento do projeto, conforme discutido na Seção 6.

6. Conclusões

A motivação central do desenvolvimento do *AuditAI* foi explorar a possibilidade de integrar ferramentas consolidadas de análise estática, como o *Slither*, com tecnologias emergentes de inteligência artificial generativa, representadas pelos modelos de linguagem da *Groq*. A partir dessa integração, buscou-se oferecer uma abordagem prática, acessível e eficaz para a geração de relatórios de auditoria mais claros, interpretáveis e personalizáveis. Os resultados obtidos confirmam essa viabilidade técnica: a plataforma demonstrou ser funcional em todas as etapas-chave do processo de auditoria, desde o envio do contrato até a exportação do relatório final, com alto grau de automação e clareza comunicativa.

O *AuditAI* mostrou-se particularmente útil em contextos com limitação de tempo ou de especialização técnica, atuando como uma ferramenta complementar que traduz

achados técnicos em conteúdos mais compreensíveis. Sua arquitetura modular,composta por frontend leve em *Svelte*, backend em *FastAPI* e um contêiner isolado com *Slither*, proporcionou robustez, reprodutibilidade e escalabilidade. Embora o sistema ainda esteja em estágio inicial, sua implementação valida a proposta de combinar análise formal com linguagem natural para aprimorar a comunicação entre segurança e desenvolvimento. Acredita-se que, com ajustes futuros, o *AuditAI* poderá evoluir para se tornar uma ferramenta estratégica no ciclo de desenvolvimento seguro de contratos inteligentes.

6.1. Trabalhos Futuros

Como próximos passos, pretende-se ampliar o escopo da plataforma por meio das seguintes ações:

- Realização de testes adicionais com outros contratos, visando validar a robustez da ferramenta em ambientes reais e em diferentes perfis de vulnerabilidades.
- Integração com outras ferramentas de análise estática e dinâmica, como fuzzers, simuladores de execução e mecanismos de detecção em bytecode, para enriquecer os diagnósticos com múltiplas perspectivas.
- Expansão do suporte para contratos desenvolvidos em outras linguagens além do Solidity, como Vyper, Move e Rust (utilizado em blockchains como Solana), ampliando o alcance da ferramenta.
- Inclusão de aprendizado de máquina dedicado à priorização de vulnerabilidades, complementando a severidade técnica com contexto histórico e estatístico.
- Adição de suporte offline, salvamento de versões e visualização interativa dos dados por meio de dashboards integrados à interface gráfica.

Esses aprimoramentos visam consolidar o *AuditAI* como uma solução aberta, extensível e alinhada às demandas práticas do ecossistema blockchain, permitindo sua adoção tanto por desenvolvedores quanto por equipes de segurança e auditores independentes.

Agradecimentos

Este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), e pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) projeto 2023/00811-0, projeto 2023/00673-7, projeto 2021/00199-8 (CPE SMARTNESS), projeto 2020/04031-1, e projeto 2018/23097-3.

Referências

David, I., Zhou, L., Qin, K., Song, D., Cavallaro, L., and Gervais, A. (2023). Do you still need a manual smart contract audit? *arXiv preprint arXiv:2306.12338*.

di Angelo, M., Durieux, T., Ferreira, J. F., and Salzer, G. (2023). SmartBugs 2.0: An execution framework for weakness detection in ethereum smart contracts. In *38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 2102–2105. IEEE Computer Society.

Dinis, J. T. S. (2022). *Automatic Bug Prioritization of SmartBugs Reports using Machine Learning*. PhD thesis, Instituto Superior Técnico, Universidade de Lisboa.

Feist, J., Grieco, G., and Groce, A. (2019). Slither: A static analysis framework for smart contracts. In *Proceedings of the 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pages 8–15. IEEE.

Ferreira, J. F., Cruz, P., Durieux, T., and Abreu, R. (2020). SmartBugs: A framework to analyze Solidity smart contracts. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 1349–1352.

Groq Inc. (2024). About us - groq. Accessed: 2024-03-01.

Jiang, B. et al. (2022). Systematic vulnerability assessment in smart contract ecosystems. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 187–201.

Palladino, E. (2023). A statistical analysis of blockchain vulnerability patterns. *International Journal of Blockchain Studies*, 12(3):45–62.

Revol, S., van der Auwera, J., and Domaschka, J. (2021). Comparison of ethereum smart contract vulnerability detection tools. *arXiv preprint arXiv:2105.06940*.

Zeppelin (2017). On the parity wallet multisig hack. Accessed: 2024-04-01.

A. Repositório do Projeto

O código-fonte completo da plataforma *AuditAI* encontra-se disponível como software livre no seguinte endereço:

- `https://github.com/Lixipluv/AuditAI`

O repositório contém instruções para instalação local, execução dos contêineres Docker, estrutura de pastas e orientações para contribuir com o desenvolvimento futuro da ferramenta.

B. Contrato Inteligente Testado

Para maior clareza, recomenda-se a importação direta de seu repositório. Segue parte do código fonte, utilizado nos testes do contrato *parity_wallet_bug_1.sol*:

```
pragma solidity ^0.4.11;

contract WalletLibrary {
    address constant lib = 0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4;

    function () payable {
        lib.delegatecall(msg.data);
    }
}

contract Wallet {
    address public owner;

    function Wallet(address _owner) payable {
        owner = _owner;
    }

    function () payable {
```

```
19            if(msg.sender != owner) revert();
20        }
21    }
```