Universidade Federal de Pernambuco

Centro de Informática

Kevin Beltrão de Melo

GERAÇÃO DE REGISTROS DE DECISÃO DE ARQUITETURA UTILIZANDO GPT 3.5

Orientador: Vinícius Cardoso Garcia

Recife, 2024

Universidade Federal de Pernambuco

Centro de Informática

Sistemas de Informação

Kevin Beltrão de Melo

GERAÇÃO DE REGISTROS DE DECISÃO DE ARQUITETURA UTILIZANDO GPT 3.5

Projeto apresentado no curso de Sistemas de Informação, como um requisito parcial para obter o Título de bacharel em Sistemas de Informação, no Centro de Informática da Universidade Federal de Pernambuco.

Orientador: Vinícius Cardoso Garcia

Recife, 2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Melo, Kevin Beltrão de.
Geração de registros de decisão de arquitetura utilizando GPT 3.5 / Kevin Beltrão de Melo. - Recife, 2024.
40

Orientador(a): Vinícius Cardoso Garcia
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Sistemas de Informação - Bacharelado, 2024.

1. Registros de decisão de arquitetura. 2. Arquitetura de software. 3. Engenharia de software. 4. Inteligência artificial. 5. Processamento de linguagem natural. I. Garcia, Vinícius Cardoso. (Orientação). II. Título.

000 CDD (22.ed.)

The mind cannot foresee its own advance .

Friedrich Hayek

Agradecimentos

Primeiramente, expresso minha sincera gratidão ao meu orientador, Professor Vinicius Garcia, cuja expertise, paciência e insights foram fundamentais para a concretização deste trabalho.

Um agradecimento especial é direcionado à minha esposa, Deborah, cujo amor, compreensão e apoio incondicional foram o meu porto seguro durante os desafios e conquistas que acompanharam o desenvolvimento deste trabalho. Sua presença constante e motivadora foi um pilar de força e inspiração.

Aos meus pais, cujo amor e sacrifícios abriram o caminho para as oportunidades que me foram dadas. Este trabalho é também o fruto do seu incansável apoio e da educação que me proporcionaram.

Estendo também meus agradecimentos aos meus irmãos, cujo apoio e trocas de conhecimentos foram fundamentais para o sucesso deste projeto. Em especial Leo, por compartilhar a sua experiência escrevendo e avaliando artigos científicos na sua carreira acadêmica.

A todos vocês, meu sincero obrigado. Cada um de vocês desempenhou um papel crucial nesta jornada e nas conquistas que ela representa.

RESUMO

O crescente complexo cenário da Tecnologia da Informação exige tomadas de decisão mais ágeis e documentadas, principalmente em relação às escolhas arquiteturais de sistemas. Neste contexto, os Architecture Decision Records (ADRs) têm se mostrado instrumentos valiosos. Entretanto, a criação manual desses registros pode ser demorada e susceptível a omissões. Este trabalho propõe o desenvolvimento e avaliação de um protótipo que utiliza inteligência artificial, especificamente o modelo GPT-3.5 da OpenAI, para gerar automaticamente ADRs. Através de um frontend intuitivo, o usuário insere uma descrição do projeto e recebe, em retorno, um ADR gerado que pode ser editado conforme necessário. A metodologia empregada abrangeu o desenvolvimento do protótipo e uma série de experimentos com profissionais de TI para avaliar a eficácia e relevância da solução. Os resultados demonstram que, embora a ferramenta possa não substituir completamente a criação manual, ela pode acelerar significativamente o processo e servir como um ponto de partida robusto para documentação arquitetural.

Palavras-chave: **ADR, Registro de Decisão de Arquitetura, Processamento de Linguagem Natural, GPT**

ABSTRACT

The increasingly complex Information Technology landscape demands swift and documented decision-making, especially concerning system architectural choices. In this context, Architecture Decision Records (ADRs) have proven to be invaluable tools. However, their manual creation can be time-consuming and prone to oversights. This study proposes the development and evaluation of a prototype leveraging artificial intelligence, specifically OpenAI's GPT-3.5 model, to automatically generate ADRs. Through an intuitive frontend interface, users input a project description and receive a generated ADR, which can be further edited as needed. The employed methodology encompassed prototype development and a series of experiments with IT professionals to assess the tool's effectiveness and relevance. Findings indicate that while the tool might not entirely replace manual creation, it can significantly expedite the process, serving as a robust starting point for architectural documentation.

Keywords: **ADR, Architecture Decision Record, Natural Language Processing, GPT.**

SUMÁRIO

1 Introdução

A crescente complexidade dos sistemas de software, juntamente com as demandas variadas do mundo tecnológico atual, torna as decisões de arquitetura cruciais para o sucesso de qualquer projeto de TI. Como destacado por (CLEMENTS et al., 2011), documentar a arquitetura é de extrema relevância para comunicar pelo arquiteto e garantir o entendimeneto dos stakeholders sem ambiguidade e permiti-los que executem seus trabalhos. Isso também permite que os próprios arquitetos possam focar no seu próprio trabalho ao invés de dedicar grande parte do seu tempo a responder dúvidas. Neste cenário, os Architecture Decision Records (ADRs) são ferramentas indispensáveis, atuando como registros detalhados das escolhas arquiteturais tomadas ao longo de um projeto, como também pode ser evidenciado por (AHMETI; LINDER; WOHLRAB, 2023). Contudo, elaborar ADRs, embora essencial, pode ser uma tarefa meticulosa e desafiadora, muitas vezes levando a potenciais omissões ou imprecisões.

A inteligência artificial (IA) tem demonstrado um potencial revolucionário em diversas áreas. Um exemplo é (VOBUGARI et al., 2022), que traz o uso de AI na área da oncologia, como detecção de câncer, classificação e descobrimento de novas drogas. Esse potencial impulsiona a pergunta: poderia a IA ser usada para facilitar a criação de ADRs? O valor dessa proposta não reside apenas na automação, mas também na capacidade de gerar um esboço ou estrutura inicial que, mesmo que não seja perfeito, pode ser ajustado e refinado posteriormente pelos arquitetos de software. Esta abordagem híbrida, combinando a eficiência da IA com a expertise humana, tem o potencial de otimizar significativamente o processo de documentação arquitetural, como (REITER; LEVINE, 1995) destacou no estudo de geração de documentação técnica com inteligência artificial.

1.1 Objetivo Geral

Desenvolver e avaliar um protótipo que, com o auxílio da API do OpenAI, gere Architecture Decision Records editáveis a partir de descrições fornecidas pelo usuário, servindo como um ponto de partida para a documentação final.

1.2 Objetivos Específicos

1. Investigar os fundamentos e formatos padrão dos Architecture Decision Records.
2. Integrar a API do OpenAI ao protótipo para aproveitar suas capacidades de processamento de linguagem natural.
3. Definir critérios de avaliação para os ADRs gerados, considerando tanto a qualidade inicial quanto a facilidade de edição.
4. Coletar feedback de profissionais da área para avaliar a eficácia, precisão e utilidade do protótipo no cenário real de desenvolvimento.

Concluindo a introdução, este trabalho estabelece objetivos claros e bem definidos, tanto geral quanto específicos, que guiam a investigação sobre o desenvolvimento e a avaliação de um protótipo para a geração de Architecture Decision Records (ADRs) utilizando o GPT-3.5. A realização desses objetivos envolve não apenas a exploração técnica e a integração de tecnologias avançadas de processamento de linguagem natural, mas também uma análise aprofundada dos padrões e critérios de qualidade dos ADRs, assim como a avaliação prática do protótipo em um cenário de desenvolvimento real.

Para embasar esses objetivos e fornecer um contexto teórico sólido para o estudo, a próxima seção deste trabalho se dedica à revisão de literatura. Nela, abordaremos os conceitos fundamentais relacionados à arquitetura de software, aos registros de decisão de arquitetura e ao processamento de linguagem natural, especificamente focando em como essas áreas se interconectam e se aplicam ao uso do GPT-3.5 na geração de ADRs. Esta revisão é essencial para compreender o terreno em que o nosso estudo está situado e para justificar a relevância e a necessidade da pesquisa proposta.

2 Revisão de literatura

No âmbito da Engenharia de Software, a arquitetura de software desempenha um papel crucial na definição de estruturas e soluções para sistemas complexos. Este documento visa aprofundar a compreensão sobre como as decisões arquitetônicas são fundamentais para o sucesso de projetos de software, enfatizando a importância de registros adequados dessas decisões. A capacidade de documentar e comunicar efetivamente as decisões arquitetônicas é essencial para a manutenção e evolução de sistemas de software.

Neste contexto, a emergência de tecnologias avançadas de Processamento de Linguagem Natural (PLN), particularmente o modelo GPT-3.5, oferece novas perspectivas para a automação e aprimoramento da geração de registros de decisão de arquitetura. A utilização de ferramentas baseadas em PLN para gerar documentação técnica não só promete eficiência, mas também uma possível melhoria na qualidade dos registros, facilitando a compreensão e a colaboração entre os envolvidos no desenvolvimento de software.

A revisão de literatura a seguir aborda três áreas principais:

- **Arquitetura de Software na Engenharia de Software:** Explorando os fundamentos da arquitetura de software e sua importância estratégica no desenvolvimento de sistemas.
- **Registro de Decisão de Arquitetura:** Discutindo a relevância e as metodologias para documentar as decisões arquitetônicas, um aspecto frequentemente subestimado na prática da engenharia de software.
- **Processamento de Linguagem Natural Para Gerar Textos:** Analisando como as tecnologias de PLN, especialmente modelos avançados como o GPT-3.5, podem ser aplicadas para gerar automaticamente documentação técnica e registros de decisão, revolucionando a maneira como interagimos e documentamos processos de software.

Este segmento busca, portanto, estabelecer um entendimento abrangente desses elementos e sua interconexão, estabelecendo uma base sólida para a investigação subsequente sobre a aplicação do GPT-3.5 na geração de registros de decisão de arquitetura em projetos de software.

2.1 Arquitetura de Software na Engenharia de Software

(BASS; CLEMENTS; KAZMAN, 2012) define que arquitetura é um conjunto de estruturas de software, que por sua vez são um conjunto de elementos que estão juntos por uma relação. Também destaca algumas características sobre arquitetura, como que arquitetura é uma abstração que busca domar a complexidade de um sistema, todo sistema de software tem uma arquitetura, e nem toda arquitetura tem que ser boa. Em alguns casos a arquitetura é feita de tal forma que resolva um problema específico, isso é chamado de padrão arquitetural. Alguns exemplos de padrões arquiteturais são o padrão de camadas e o padrão client-server.

Segundo (KOUROSHFAR et al., 2015), arquitetura de software tem um papel relevante na manutenção de software. Particularmente importante pela facilidade que traz de entender e trabalhar em cima de diferentes camadas de código sem precisar do contexto de outras partes do projeto ou de se aprofundar no código de mais baixo nível. É concluído que arquitetura de software é um dos fatores chave que afetam a qualidade de um sistema de software em mudança. Também é destacado como a arquitetura deve quebrar uma aplicação a ponto de auxiliar o a localização de grandes defeitos, assim como as suas causas.

Como apontado por (AKMEL et al., 2017), diferentes arquiteturas trazem diferentes vantagens e desvantagens. O entendimento dessas vantagens e desvantagens é essencial para que a arquitetura sirva bem o padrão de projeto. O design pattern teve uma definição trazida por (ALEXANDER CHRISTOPHER, 1977), considerada canônica, como uma solução para um problema recorrente em nosso ambiente. Em outras palavras, uma forma de reutilização de uma solução múltiplas vezes para um mesmo problema.

2.2 Registro de Decisão de Arquitetura

Registro de decisão de arquitetura, ou Architecture Decision Record (ADR) é um tipo de documento que, como aponta (CHAPPEN et al., 2021), serve como um registro de decisões de um software. A ideia é documentar as mudanças para ter mapeado os fatores como o contexto, consequências e alternativas. Esse documento evita a concentração de conhecimento, auxilia na disseminação da informação através de diferentes times e força quem está escrevendo a considerar diferentes alternativas e justificar a escolha.

Em (AWS,), o processo de adoção de um ADR é explicado. Primeiramente é necessário selecionar quem vai ser o responsável por escrever e comunicar o conteúdo. Ao escrever, o documento estará pronto para revisão e o time deverá definir se a mudança de arquitetura será aceita, rejeitada ou se o ADR necessita mais trabalho antes da decisão ser tomada.

Mais um ponto de importância é citado por (KEELING, 2022), quando aponta que, antes, os arquitetos eram guardiões do design, mas a adoção de ADRs empodera outros desenvolvedores a serem responsáveis por ele. Isso acontece pois ADR é um documento de texto que pode ser escrito por qualquer um que possua um editor de texto, é armazenado no mesmo software de controle de versão que o projeto, e não há a necessidade de conhecimentos ou notações especiais para contribuir com o documento.

Como exemplificado em (AHMETI; LINDER; WOHLRAB, 2023), as seções típicas de um ADR incluem: Contexto, que descreve o problema que a decisão visa resolver; Decisão, onde a solução escolhida é explicitada; Alternativas, que detalha as opções consideradas, incluindo prós e contras de cada uma; Consequências, que discute os impactos da decisão, tanto positivos quanto negativos; e Status, indicando se a decisão está proposta, aceita, rejeitada ou deprecada. Este formato ajuda a garantir que as decisões arquiteturais sejam bem documentadas, compreendidas e facilmente acessíveis para toda a equipe de desenvolvimento. Um exemplo de ADR seria:

```
# Título
## Contexto
Contextualizando a situação atual do projeto
## Decisão
Explicando a mudança que ocorrerá
## Alternativas
Foi considerado A e B mas foi preferido C por este motivo
## Status
Aprovada
## Consequências
Agora o serviço X precisará se comunicar com o serviço Y através de Z
```

2.3 Processamento de Linguagem Natural Para Gerar Textos

Como explicado por (BIRD; KLEIN; LOPER, 2009), processamento de linguagem natural, ou natural language processing (NLP), é a área de inteligência artificial que se dedica a interpretar linguagem humana, como inglês, espanhol ou português. Como é explicitado por (LANE; HAPKE; HOWARD, 2019), NLP se distingue de linguagem de programação pois não se propõe a ser transformado em um conjunto de operações matemáticas finito. Trata-se de comandos mais complexos que não vão depender de escrever um código para explicitar exatamente o que o computador vai fazer.

Também é possível fazer ferramentas de diálogo que geram textos para responder um input. Isso funciona tanto para respostas curtas, como em chatbots e assistentes virtuais, quanto para gerar textos mais longos, como é feito com o ChatGPT do OpenAI. O ChatGPT pode ser usado como um auxílio para desenvolver conteúdos mais extensos e foi utilizado como ferramenta para interpretar a descrição do usuário e devolver um documento de ADR neste trabalho.

(LI et al., 2016) traz um bom exemplo do uso de NLP dentro do desenvolvimento de software. Foi usado o abordagem chamada de UnitTestScribe para gerar linguagem natural para documentar testes unitários. Como resultado, acontecia menos de informações importantes serem deixadas de fora, o texto teve menos informação redundante e o output manteve uma boa legibilidade e inteligibilidade.

A revisão de literatura realizada revelou aspectos fundamentais e interconectados no campo da Engenharia de Software, com foco na arquitetura de software, no registro de decisões arquitetônicas e na aplicação do PLN para a geração de textos. Através desta análise, é possível reconhecer a importância crítica da arquitetura de software como a espinha dorsal de projetos de desenvolvimento, onde cada decisão arquitetônica influencia diretamente a qualidade, a manutenibilidade e a escalabilidade dos sistemas.

A documentação dessas decisões, frequentemente subestimada na prática de engenharia, emerge como um componente vital para a comunicação eficaz, a gestão do conhecimento e a continuidade dos projetos de software. No entanto, observa-se que a criação de registros detalhados e acessíveis permanece um desafio, dada a complexidade inerente e as demandas de tempo.

Nesse contexto, o advento de modelos avançados de PLN, especificamente o GPT-3.5, apresenta uma oportunidade transformadora. Os insights obtidos destacam o po-

tencial do GPT-3.5 para automatizar e enriquecer o processo de geração de registros de decisão de arquitetura. A capacidade deste modelo de gerar textos coerentes, contextuais e tecnicamente precisos sugere uma revolução não apenas na eficiência, mas também na eficácia da documentação em projetos de software.

A interseção dessas três áreas - arquitetura de software, registro de decisões e PLN - abre caminhos inovadores para a pesquisa e a prática na engenharia de software. Esta revisão de literatura fornece uma base teórica robusta para a investigação proposta neste trabalho, que explora a aplicabilidade do GPT-3.5 na otimização dos registros de decisão de arquitetura, potencialmente remodelando as práticas de documentação na engenharia de software.

Com base nos insights e fundamentações teóricas apresentadas nesta revisão, a próxima seção deste trabalho se dedica a descrever a metodologia empregada. Esta seção detalhará os procedimentos, técnicas e ferramentas utilizadas para explorar a aplicação prática do GPT-3.5 na geração de registros de decisão de arquitetura, estabelecendo o caminho para uma análise profunda e uma avaliação criteriosa dos resultados obtidos.

3 Metodologia

A escolha da metodologia é uma etapa crucial no desenvolvimento de qualquer pesquisa, pois fornece o arcabouço sobre o qual se baseiam as etapas subsequentes de investigação. Essa importância é apresentada por (COSTA; JüNI, 2014), mostrando que os dados devem ser extraídos de forma padronizada. No contexto deste TCC, cujo tema centra-se na geração automática de Architecture Decision Records (ADR) com auxílio da inteligência artificial, a metodologia adotada buscou integrar práticas de desenvolvimento de software com a avaliação da usabilidade e eficácia da solução proposta.

3.1 Abordagem metodológica

A abordagem metodológica selecionada foi orientada principalmente pelo desenvolvimento iterativo e integrado de um protótipo. Esta escolha permitiu que o sistema fosse refinado progressivamente com base nos feedbacks e resultados de testes iniciais. A integração entre frontend e backend, assim como a interação com a API do OpenAI, foram componentes fundamentais nesta fase. Adicionalmente, a abordagem também considerou aspectos qualitativos, como a experiência do usuário e a relevância do conteúdo gerado.

Em essência, a metodologia de pesquisa adotada foi caracterizada por uma combinação de desenvolvimento prático, avaliação técnica e feedback do usuário, garantindo que a solução desenvolvida fosse não apenas tecnicamente robusta, mas também de valor prático para os profissionais da área.

As entrevistas foram conduzidas com um grupo selecionado de profissionais de diferentes áreas de tecnologia da informação. Totalizando 12 participantes, dentre as diferentes áreas de especialização: Software Reliability Engineering, Data Science, Frontend Engineering, Backend Engineering, Mobile Engineering e Platform Engineering; profissionais de 2 a 14 anos de experiência; de bacharelandos a doutores; de diferentes nacionalidades; e que atuam em mercados de diferentes nacionalidades. Eles trabalhando em diferentes setores, oferece uma perspectiva diversificada e rica sobre a utilidade e eficácia da ferramenta ADR.

As entrevistas foram realizadas seguindo um protocolo consistente, onde os participantes primeiro utilizaram a ferramenta ADR e, subsequente a essa experiência, responderam a um conjunto de perguntas padronizadas. O protocolo incluiu solicitações para que os participantes editassem textos com a assistência da ferramenta, seguido de uma avaliação qualitativa da experiência.

3.2 Protótipo e desenvolvimento

3.2.1 Estruturação do Repositório

O desenvolvimento do protótipo foi realizado em um monorepo hospedado no GitHub ((MELO,)). Este repositório é dividido em dois pacotes: frontend e backend, ambos escritos em TypeScript, garantindo tipagem estática e prevenção de erros em tempo de compilação, como é mostrado por (BOGNER; MERKEL, 2022).

3.2.2 Desenvolvimento Frontend

A implementação do frontend foi fundamentada no uso do ReactJS, empregando o Vite como acelerador de compilação. Esta combinação de ferramentas proporciona uma experiência de desenvolvimento mais ágil e responsiva devido ao SWC. SWC é uma plataforma baseada em Rust e em (SWC. . . ,) promete que é 20x mais rápido do que Babbel, que é utilizado pelo create-react-app, ferramenta desenvolvida pela mesma empresa que desenvolveu o ReactJS, o Meta. A principal interação do usuário na plataforma ocorre através de um campo de entrada, designado para receber a descrição do projeto. Ao ser submetida, essa descrição é enviada ao backend via uma requisição HTTP. Na resposta desta requisição, o frontend recebe o conteúdo em *Markdown*, que é, então, apresentado ao usuário. Para este propósito, foi adotado o módulo *React-markdown*, permitindo uma visualização interativa e simultânea do conteúdo em um textarea. Esta abordagem visa não apenas entregar ao usuário um ADR gerado, mas também proporcionar um ambiente onde este ADR possa ser editado e refinado conforme as necessidades específicas do usuário. Além disso, foi adicionada uma funcionalidade que permite ao usuário baixar o conteúdo final em formato .md.

3.2.3 Desenvolvimento Backend

A construção do backend foi realizada sobre o framework Express.js, uma opção robusta e amplamente reconhecida na comunidade de desenvolvimento, para criar servidores web em Node.js. O Stack Overflow Survey 2023 (STACK...,) teve como resultado Node.js como a ferramenta mais popular para web. A principal responsabilidade deste backend é a interação com a API do OpenAI para geração dos ADRs. A integração foi alcançada por meio do módulo NPM openAI. A escolha do modelo de inteligência artificial foi fundamental para a qualidade dos resultados. Inicialmente, foi testado o modelo text-davinci-003 e um prompt pedindo explicitamente para o ADR ser gerado. Os resultados durante os testes foram outputs muito parecidos com os inputs de descrição porém com bastante repetição. Após essa análise inicial, foi testado modelo gpt-3.5-turbo e foi definido um novo prompt, que buscava não simplesmente passar uma instrução, mas fazer uma encenação pedindo para o ChatGPT simular ser um arquiteto. Essas atualização fizeram com que a ferramenta passase a gerar resultados mais complexos e menos prolíxos. O prompt final utilizado foi:

```
‘You are a staff engineer responsible for creating Architecture Decision
Records.

You will be provided a description of an architecture decision and you will
need to create an ADR.

As a staff engineer, it’s also expected you will provide new insights that are
not mentioned on the input, such as consequences, alternatives, and rationale.

Description:\n\n‘
```

Como mencionado, o prompt definido inicialmente não estava desempenhando bem. O texto de output estava sendo uma versão prolixa do input e não estava trazendo todas as seções relevantes para um ADR. Este prompt era *"Just respond using markdown syntax. Create an architecture decision record with the following description:"*. Em um experimento com um usuário, o ADR gerado só continha seções de contexto, decisão e consequências, sem levantar alternativas e novos insights. Para demonstrar a superficialidade, a seguir foi o prompt usado pelo usuário:

```
There is a need to scale the provisioning capabilities of account manager.
Currently, we are using a terraform code to provision AWS sub-accounts
dynamically, this approach has many pitfalls mainly when we start to push
changes and upgrades to existing infrastructure. To make this process easier to
manage we will switch terraform to crossplane and use a pub sub-architecture to
trigger the creation of crossplane composition claims by templating and pushing
yaml manifests to a git repository that will later be applied in the cluster by
argocd.

To make the implementation easier we will use Argo Workflows and Argo Events to
subscribe to a rabbitmq queue in which the account manager application will
send messages to trigger the provisioning process.
```

E a seguir o conteúdo da seção de consequências:

```
The main benefit of this approach is that it will make the provisioning process
easier to manage and more fault tolerant. The main drawback is that it will
require more complex code and more resources to run the Argo Workflows and Argo
Events.
```

3.3 Coleta de dados

Os experimentos foram conduzidos por meio de chamadas virtuais com profissionais de diversas áreas da TI. Estes participantes incluíam: cientistas de dados, engenheiros de backend, engenheiros de frontend, engenheiros mobile e SREs (Site Reliability Engineers). O objetivo destas chamadas era convidar cada profissional a testar a aplicação em um cenário que simulasse uma situação real de seu trabalho cotidiano. Ou seja, cada entrevistado escolheu um exemplo dentro do seu próprio escopo. Estas interações permitiram uma avaliação prática da ferramenta em contextos diversificados, proporcionando uma visão mais abrangente de sua utilidade e aplicabilidade.

3.3.1 Fase de Teste de Usuários

Nesta etapa do experimento, o foco estava em coletar dados objetivos sobre a interação do usuário com a ferramenta. Dois principais indicadores foram monitorados:

- Quantidade editado: Foi avalido o quanto cada profissional editou o ADR gerado automaticamente pela ferramenta. Isso forneceu insights sobre o quão próximo o resultado gerado estava do esperado e quanto esforço adicional era necessário para adequá-lo.
- Motivo da Edição: Foi avaliado o motivo que o usuário editou as partes específicas no ADR gerado. Isso proporcionou uma perspectiva sobre a precisão e relevância do conteúdo gerado.

3.3.2 Fase de Questionamento

Após a interação prática com a ferramenta, os participantes responderam a um conjunto de perguntas qualitativas e quantitativas, desenhadas para avaliar tanto aspectos técnicos quanto de usabilidade da solução:

1. Nota de 0 a 10 do quão relevante foi o ADR gerado
2. Nota de 0 a 10 sobre o quão intuitiva é a ferramenta
3. Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?
4. Você usaria essa ferramenta em seus projetos futuros? Por quê?
5. Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?
6. Gostaria que alguma funcionalidade fosse adicionada à ferramenta?
7. Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?:

A metodologia adotada neste estudo reflete um equilíbrio entre teoria e prática, visando uma compreensão profunda da aplicabilidade do GPT-3.5 na geração de registros de decisão de arquitetura. A construção e utilização de uma aplicação prática, testada por

profissionais de TI, forneceram um campo de estudo realista para avaliar a eficácia desta tecnologia inovadora. A abordagem metodológica escolhida, combinando o desenvolvimento de um protótipo funcional com a coleta de dados através de entrevistas, permitiu uma análise detalhada das interações dos usuários com a ferramenta e suas percepções.

O desenvolvimento do protótipo, com sua arquitetura dividida entre frontend e backend e utilizando tecnologias atuais como React, TypeScript e Express.js, não só demonstra a viabilidade técnica do projeto, mas também serve como um caso de estudo relevante para a comunidade de desenvolvimento de software.

A coleta de dados, meticulosamente planejada e executada, fornece um conjunto robusto de informações, provenientes das experiências e opiniões dos profissionais de TI. As entrevistas realizadas oferecem insights valiosos sobre a usabilidade e a utilidade da aplicação, além de destacar áreas para futuras melhorias.

Com base nesta abordagem metodológica e nos dados coletados, a próxima seção deste trabalho se concentra na apresentação e análise dos resultados obtidos. Esta seção irá explorar as implicações práticas e teóricas dos dados coletados, fornecendo uma avaliação crítica do impacto do GPT-3.5 na geração de registros de decisão de arquitetura em ambientes de desenvolvimento de software.

4 Resultados

Os resultados foram analisados através da utilização da aplicação por usuários seguida de uma entrevista estruturada, destinada a avaliar a percepção dos usuários sobre a relevância e intuição da ferramenta de ADR após a realização de edições em textos.

Dentro deste contexto, os resultados das entrevistas serão apresentados como um resumo agregado, que enfatizará os temas e padrões emergentes, assim como os insights individuais quando notáveis. As respostas dos participantes foram avaliadas em termos de satisfação com a relevância das informações fornecidas pela ADR e pela facilidade de uso percebida. Além disso, as notas atribuídas por eles em relação à relevância (com uma média de 9.1 em uma escala de 0 a 10 pontos) e intuição (com uma média de 9.4 em uma escala de 0 a 10 pontos) da ferramenta também serão discutidas para oferecer uma avaliação qualitativa do seu desempenho.

Este capítulo segue com uma apresentação dos resultados coletados, o que fornecerá uma base para a discussão posterior sobre as implicações destes achados para a prática regulatória e para o aprimoramento contínuo da ferramenta ADR.

4.1 Ameaças à Validade

É importante reconhecer as limitações e possíveis fontes de viés que podem afetar a validade dos resultados obtidos. Uma ameaça significativa à validade deste estudo reside no conhecimento prévio do pesquisador sobre os entrevistados. Para mitigar esse viés, adotaram-se estratégias metodológicas, como:

- **Triangulação de Dados:** Utilizou-se a triangulação de dados, comparando as respostas dos entrevistados com dados quantitativos obtidos através da ferramenta, para assegurar que as interpretações não dependessem unicamente da perspectiva do pesquisador.

- **Protocolo de Não Intervenção:** Evitou-se tirar dúvidas do entrevistado uma vez que o experimento era iniciado para resultados mais autênticos das ações e interpretações do usuário.

4.2 Dados coletados

Participante 1

Software Reliability Engineer, bacharel, 6 anos de experiência, brasileiro, mercado americano

- **Edição:**
  - Não editou o ADR gerado.
- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 9
- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 8
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - Sim, a ferramenta organizou bem o pensamento e ajudou a trazer insights valiosos principalmente na parte de vantagens e desvantagens.
- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** Adição de tradeoffs nas tecnologias usadas, adição de outras tecnologias possíveis para atingir o mesmo objetivo a serem consideradas, organização do fluxo da arquitetura num passo a passo.
  - **Pontos Fracos:** Algumas definições sobre as tecnologias usadas estavam um pouco imprecisas (na parte do argo workflows ser o ponto que garante a idempotência) ou vagas (com relação aos benefícios do crossplane). Mas no final das contas isso me direcionou a um ponto de informação que eu poderia trabalhar melhor na ADR.
- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Continuo achando que um "wizard"que guie a escrita do prompt ajudaria bastante principalmente pra quem não tem prática fazendo ADR

- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Ajudou a trazer clareza sobre pontos ou tecnologias que eu não mencionei no meu prompt. Visto que as ferramentas que eu utilizei eram simplemente templates no estilo "fill the blanks"ter um insight de tecnologias que eu deixei de fora da avaliação e alguns pontos negativos mais opinativos ajuda a pensa melhor.

Participante 2

Mobile Engineer, bacharelando, 3 anos de experiência, brasileiro, mercado americano.

- **Edição:**
  - Removeu um parágrafo que achou pouco relevante, reescreveu uma frase para aumentar clareza, adicionou outras 2 frases e achou um parágrafo redundante.
- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 9
- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 10
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - im, tanto pela geração de conteúdo automática quanto pela formatação em Markdown.
- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** Simples, fácil de usar, resultado que não é muito longo e bem dividido
  - **Pontos Fracos:** Talvez alguns tópicos fossem prolíxos ou pouco relevantes
- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**

  - Ter opção de dar um regenerate em um tópico específico e botão de copiar markdown

- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**

  - Mais fácil do que usar o ChatGPT diretamente

Participante 3

Frontend Engineer, bacharelando, 2 anos de experiência, brasileiro, mercado brasileiro.

- **Edição:**

  - Não editou o ADR gerado.

- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 10

- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 10

- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim

- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**

  - Sim. Vai ter a necessidade de criar novos ADRs e a ferramenta ajudou.

- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**

  - **Pontos Fortes:** Gerar novos insights
  - **Pontos Fracos:** Nenhum

- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**

  - Seria bom poder enviar o esboço da sua organização de pasta e ele retornar um exemplo com a nova arquitetura.

- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**

  - Nunca usou outras ferramentas para criar ADRs.

Participante 4

Backend Engineer, bacharelando, 2 anos de experiência, brasileiro, mercado brasileiro.

- **Edição:**
  - Não editou o ADR gerado.
- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 10
- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 10
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - Sim, demoraria 4 ou 5 vezes mais para fazer um ADR manualmente.
- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** Rápido.
  - **Pontos Fracos:** Sentiu falta de consequência negativas.
- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Seria bom poder armazenar e usar outros ADRs que ela usou como contexto.
- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Não conhece outros métodos.

Participante 5

Frontend Engineer, bacharel, 8 anos de experiência, canadense, mercado americano.

- **Edição:**

  - Não editou nada mas disse que mudaria um dos tópicos para ser mais claro. Seria bom ser mais aprofundado com alguns pitfalls mais específicos. Adicionou mais detalhes e gerou um novo output.

- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 7.5
- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 9.5
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - Sim, isso pula uma etapa para formatar e dá um "skeleton"para o ADR final. É uma etapa acima do GPT.
- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** É ótimo para skeleton, levanta novos insights.
  - **Pontos Fracos:** O output é um pouco superficial, não tem muita profundidade (talvez o input não tenha sido tão específico)
- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Seria interessante ter mais label explicando como usar a ferramenta e explicações, por exemplo, explicando o quão profunda deve ser a descrição da aplicação.
- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Normalmente usaria puro ChatGPT para os skeletons. É melhor que o GPT porque é já é mais específico e só pede a descrição do projeto.

Participante 6

Frontend Engineer, bacharelando, 2 anos de experiência, brasileiro, mercado brasileiro.

- **Edição:**

  - Removeu uma frase por achar repetitivo

- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 10
- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 9
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - Sim. Pois facilita, economiza tempo, foi assertivo nos tópicos, gerou insights que ele não teria pensado, boa padronização.
- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** Adição de tradeoffs nas tecnologias usadas, adição de outras tecnologias possíveis para atingir o mesmo objetivo a serem consideradas, organização do fluxo da arquitetura num passo a passo.
  - **Pontos Fracos:** Algumas definições sobre as tecnologias usadas estavam um pouco imprecisas (na parte do argo workflows ser o ponto que garante a idempotência) ou vagas (com relação aos benefícios do crossplane). Mas no final das contas isso me direcionou a um ponto de informação que eu poderia trabalhar melhor na ADR.
- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Continuo achando que um "wizard"que guie a escrita do prompt ajudaria bastante principalmente pra quem não tem prática fazendo ADR
- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Ajudou a trazer clareza sobre pontos ou tecnologias que eu não mencionei no meu prompt. Visto que as ferramentas que eu utilizei eram simplemente templates no estilo "fill the blanks"ter um insight de tecnologias que eu deixei de fora da avaliação e alguns pontos negativos mais opinativos ajuda a pensa melhor.

Participante 7

Software Reliability Engineer, bacharel, 3 anos de experiência, brasileiro, mercado americano.

- **Edição:**
  - Não editou o ADR gerado mas adicionaria imagens.
- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 10
- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 8
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - Usaria com certeza. Somente o git não é o suficiente para você ver a mudança da arquitetura, apenas de código.
- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** Foi muito preciso, adicionou mais coisa do que ele tinha comentado e que eram relevantes, bem organizado e estruturado.
  - **Pontos Fracos:** A seção de contexto foi praticamente ctrl c ctrl v do que escrevi e não tem imagem.
- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Gostaria de manter uma conversa para ir editando o texto gerado.
- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Foi a primeira ferramenta que usou.

Participante 8

Backend Engineer, bacharelando, 2 anos de experiência, brasileiro, mercado brasileiro.

- **Edição:**
  - Não editou o ADR gerado por estar satisfeito.
- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 10
- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 10
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - Sim, porque economiza tempo para documentar.
- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** Conseguir compreender o contexto e tentar simular casos para gerar insights.
  - **Pontos Fracos:** Poderia ter levantado mais pontos negativos.
- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Seria interessante incluir o prompt usado na ferramenta ao final do ADR gerado.
- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Não usou outras ferramentas.

Participante 9

Platform Engineer, bacharel, 14 anos de experiência, brasileiro, mercado americano.

- **Edição:**
  - Não editou o ADR gerado.
- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 8

- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 9
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - Sim.
- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** Bem formatado, bem estruturado, proposta boa.
  - **Pontos Fracos:** Senti falta de referências, achei genérico.
- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Seria interessante passar o link do repositório para o app ler
- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Nunca usou outra ferramenta para criar ADRs

Participante 10

Frontend Engineer, bacharel, 9 anos de experiência, argentino, mercado americano.

- **Edição:**
  - Não editou o ADR gerado mas diminuiria repetição, removeria algumas partes ou juntaria.
- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 8
- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 10
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - Sim.

- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** Ajuda no inglês, economizou tempo, editor markdown, gera markdown estruturado.
  - **Pontos Fracos:** Muita repetição, criou uma referência que não existe
- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Seria bom escolher um template de ADR, escolher as sections.
- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Normalmente escreve do 0 e já usou template da empresa, mas preferiu a ferramenta.

Participante 11

Data scientist, doutor, 4 anos de experiência, brasileiro, mercado brasileiro.

- **Edição:**
  - Ajustou uma frase por clareza, ajustou um erro que apareceu por erro no input, tirou uma das alternativas porque achou pouco relevante.
- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 8
- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 10
- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim
- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - - Já tem o hábito de criar ADRs e com poucos ajustes já seria suficiente para apresentar aos superiores.
- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**

  - **Pontos Fortes:** Conseguiu pegar meu texto e botar todas as condições no lugar certo, reordenou bem, estruturou bem.
  - **Pontos Fracos:** Uma das alternativas ser pouco relevante.

- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Labels pros inputs. Ter um prompt de palavras-chave para especializar mais o output.

- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Não conhece nenhuma ferramenta que faça isso, mas já usou o ChatGPT para isso e achou que a ferramenta dá uma segurança no resultado. Sente que o ChatGPT é menos controlado.

Participante 12

Data scientist, mestre, 7 anos de experiência, brasileiro, mercado brasileiro.

- **Edição:**
  - Editou uma frase para adicionar relevância. Removeu frase por ser pouco relevante. Adicionou mais uma alternativa.

- **Nota de 0 a 10 do quão relevante foi o ADR gerado:** 9.5

- **Nota de 0 a 10 sobre o quão intuitiva é a ferramenta:** 9

- **Você acredita que a ferramenta economizou seu tempo em relação à criação manual de um ADR?** Sim

- **Você usaria essa ferramenta em seus projetos futuros? Por quê?**
  - Sim, pois economizou tempo.

- **Quais foram os principais pontos fortes e fracos que você identificou no ADR gerado?**
  - **Pontos Fortes:** Estrutura correta tanto do bom uso do markdown quanto trazendo os pontos relevantes de um ADR. Trouxe insights importantes, destacou os pontos de mudança bem.

  - **Pontos Fracos:** Trouxe pontos irrelevantes e esqueceu de trazer ponto importante sobre custo.

- **Gostaria que alguma funcionalidade fosse adicionada à ferramenta?**
  - Uma forma de ajudar a estruturar conhecimento do domínio ou de extrair mais informações sobre o domínio antes de dar a resposta. Por exemplo: através de um formulário escolher uma área de especialidade, e um prompt especificando o escopo ao GPT para evitar ambiguidade.

- **Como você compara essa ferramenta com outros métodos que você já usou para criar ADRs?**
  - Muito mais rápido, ele falou o que eu queria falar e precisava ser documentado, o que os demais stakeholders gostariam de ler. Bem mais rápido do que fazer na mão. Normalmente faria manualmente.

4.3 Análise dos dados agregados

4.3.1 Avaliação Geral da Relevância dos ADRs gerados

A média geral de relevância das informações apresentadas pela ADR foi avaliada em 9.1 de um máximo de 10 pontos. Este resultado expressa uma validação positiva dos usuários sobre a aplicabilidade e pertinência dos dados fornecidos pela ferramenta no contexto regulatório. Indica que, na maioria dos casos, a ADR conseguiu atender às expectativas dos usuários em prover um ADR completo o suficiente para ser adotado ou apenas adaptado antes de ser utilizado como documentação no trabalho.

Foi notado, no entanto, uma diferença de expectativa dos usuários nas notas mais baixas da pesquisa. A nota mais baixa para esse critério foi 7.5 e o usuário sentiu falta de uma análise mais profunda e insights mais completos do impacto da mudança de arquitetura na empresa. Vale destacar que a ferramenta tem como objetivo auxiliar para gerar um ADR e também pode auxiliar com novos insights, mas não tem como objetivo primário fazer uma análise aprofundada sobre a arquitetura.

4.3.2 Avaliação Geral da Intuição da Interface de Uso

A média geral para a intuição de uso da interface foi muito próxima da anterior, situando-se em 9.4 de 10. Esta pontuação reflete uma aceitação geral da usabilidade da ferramenta, mas com evidência de que melhorias são necessárias para otimizar a experiência do usuário. Comentários qualitativos apontam para desafios no compreendimento do que é esperado no input fornecido pelo usuário para ter os melhores outputs. Além disso, foi levantado por múltiplos usuários que a falta de labels nos inputs deixou a aplicação menos clara. Esses insights sugerem que, embora a ferramenta seja funcional, aprimoramentos na interface podem potencializar a eficiência e a satisfação do usuário.

4.3.3 Economia de tempo em relação a produção manual de um ADR

100% dos entrevistados apontaram que economizaram tempo em relação à criação manual de um ADR. Mesmo os que sentiram a necessidade de editar mais o texto gerado sentiram que tiveram menos trabalho do que se tivessem parado para escrever o conteúdo do zero, se preocupando também com formatação do texto.

4.3.4 Uso da ferramenta em seus projetos futuros

Nesta questão, mais uma vez, os entrevistados foram unânimes em suas respostas. Todos afirmaram que usariam esta ferramenta em projetos futuros. Alguns dos motivos que mais apareceram foram a formatação automática em Markdown, a economia de tempo e os insights gerados. Esse talvez seja o feedback mais importante da entrevista, já que demonstra que os usuários da aplicação preferem usá-la do que alternativas.

4.3.5 Principais pontos fortes e fracos

Os principais pontos fortes utilizados foram relacionados aos insights gerados, economia de tempo e formatação do texto. Já os pontos fracos, sentiram falta de profundidade em alguns tópicos, acharam algumas partes do texto gerados redundantes ou prolixas e acharam parte do texto muito similar ao input dado pelo usuário.

4.3.6 Sugestão de funcionalidade

Todos os entrevistados trouxeram sugestões de novas funcionalidades à aplicação. Uma das que foi mais citada foi a capacidade de trocar múltiplas mensagens para ir aperfeiçoando o texto baseado em outputs anteriores. Outras sugestões interessantes foram a geração de imagens para ilustrar o ADR e integração com o GitHub para abrir um pull request com o ADR gerado.

4.3.7 Comparação com outras ferramentas

‘A outra forma que os usuários já utilizaram para gerar ADR foi ou de forma manual ou com auxílio do ChatGPT. Todos demonstraram preferência pela utilização da ferramenta em relação à alternativa devido ao tempo economizado, prompt definido e formatação Markdown automática.

Os resultados apresentados e analisados nesta seção revelam insights valiosos sobre a eficácia do GPT-3.5 na criação de registros de decisão de arquitetura e seu impacto na prática da engenharia de software. Através da análise dos dados coletados, pudemos explorar tanto os benefícios quanto os desafios enfrentados pelos profissionais de TI ao interagir com a aplicação desenvolvida. Estas descobertas não apenas confirmam algumas das hipóteses iniciais, mas também abrem novos caminhos para pesquisas futuras. Ao avançarmos para a seção de conclusão, refletiremos sobre as implicações gerais desses resultados, sintetizando as principais contribuições do estudo para o campo de engenharia de software e delineando recomendações para práticas futuras e investigações acadêmicas.

5 Conclusão

A incorporação da ferramenta automatizada para a geração de Architectural Decision Records (ADRs) nos fluxos de trabalho de engenharia de software, conforme explorado por meio das entrevistas com profissionais do campo, revela um horizonte promissor onde a eficiência e a qualidade da documentação podem ser substancialmente aprimoradas.

A unanimidade observada na economia de tempo oferecida pela ferramenta não apenas reforça a sua eficácia, mas também ressalta a importância de otimizar práticas de documentação em um setor onde o tempo é um recurso precioso. A capacidade de acelerar processos sem comprometer a integridade e a profundidade dos ADRs tem o potencial de liberar recursos humanos valiosos para o engajamento em tarefas de maior valor agregado e inovação.

A intuitividade e usabilidade da ferramenta, aliadas à qualidade do conteúdo gerado, refletem um desenvolvimento tecnológico bem-sucedido que encontra ressonância na comunidade técnica. No entanto, a busca contínua por melhorias e ajustes é uma clara indicação de que a evolução tecnológica é um processo iterativo e colaborativo.

As perspectivas futuras para a ferramenta são otimistas, com a potencialidade de integrações mais profundas e personalizadas sugeridas pelos entrevistados, ampliando seu alcance e eficácia. O entusiasmo pela continuidade do uso da ferramenta nos projetos futuros é um indicativo de sua relevância e alinhamento com as necessidades dos profissionais da área.

Conclui-se, portanto, que a adoção desta ferramenta de IA representa um avanço significativo na maneira como os ADRs são concebidos e produzidos, marcando um passo rumo a uma prática mais ágil e inteligente de documentação de decisões arquiteturais. Como consequência, instiga-se uma reflexão sobre o papel crescente da automação e da inteligência artificial em todas as facetas da engenharia de software, uma tendência que está claramente se mostrando não apenas inevitável, mas altamente benéfica.

É imperativo que continuemos a monitorar, avaliar e aperfeiçoar essas ferramentas, pois sua maturação e refinamento serão cruciais para a maximização de seu potencial.

Afinal, a tecnologia, no seu melhor, deve servir para ampliar e enriquecer a capacidade humana, um objetivo que esta ferramenta demonstra estar cumprindo com mérito.

Referências

AHMETI, B.; LINDER, M.; WOHLRAB, R. Exploring the adoption and effectiveness of architecture decision records in agile software development: An action research study. In: ACM. *Proceedings of the 11th ACM Celebration of Women in Computing: womENcourage™ 2023*. Trondheim, Norway, 2023.

AKMEL, F. et al. A comparative analysis on software architecture styles. *International Journal in Foundations of Computer Science & Technology*, v. 7, p. 11–22, 11 2017.

ALEXANDER CHRISTOPHER, e. a. *A Pattern Language*. [S.l.]: Oxford University Press, 1977.

AWS, A. <https://docs.aws.amazon.com/prescriptive-guidance/latest/architectural-decision-records/adr-process.html>. Accessed: 2023-11-24.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice: Software Architect Practice_ c3*. Pearson Education, 2012. 1-35 p. (SEI Series in Software Engineering). ISBN 9780132942782. Disponível em: <https://books.google.com.br/books?id=-II73rBDXCYC>.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. Beijing: O'Reilly, 2009. 1-35 p. ISBN 978-0-596-51649-9. Disponível em: <http://www.nltk.org/book>.

BOGNER, J.; MERKEL, M. To type or not to type? a systematic comparison of the software quality of javascript and typescript applications on github. In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. [S.l.: s.n.], 2022. p. 658–669.

CHAPPEN, E. et al. *18F: Digital service delivery*. 2021. <https://18f.gsa.gov/2021/07/06/architecture_decision_records_helpful_now_invaluable_later/>. Accessed: 2023-11-24.

CLEMENTS, P. et al. [S.l.]: Software Engineering Institute | Carnegie Mellon, 2011. 9-21 p.

COSTA, B. R. da; JüNI, P. Systematic reviews and meta-analyses of randomized trials: principles and pitfalls. *European Heart Journal*, Oxford University Press (OUP), v. 35, n. 47, p. 3336–3345, nov. 2014. ISSN 0195-668X. Disponível em: <http://dx.doi.org/10.1093/eurheartj/ehu424>.

KEELING, M. Love unrequited: The story of architecture, agile, and how architecture decision records brought them together. *IEEE Software*, v. 39, n. 4, p. 90–93, 2022.

KOUROSHFAR, E. et al. A study on the role of software architecture in the evolution and quality of software. In: *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. [S.l.: s.n.], 2015. p. 246–257.

LANE, H.; HAPKE, H.; HOWARD, C. *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python*. Manning Publications, 2019. 1-40 p. ISBN 9781617294631. Disponível em: <https://books.google.com.br/books?id=UyHgswEACAAJ>.

LI, B. et al. Automatically documenting unit test cases. In: *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. [S.l.: s.n.], 2016. p. 341–352.

MELO, K. B. de. <https://github.com/KevBeltrao/adr-generator>. Accessed: 2023-12-14.

REITER, C. M. E.; LEVINE, J. Automatic generation of technical documentation. *Applied Artificial Intelligence*, Taylor and Francis, v. 9, n. 3, p. 259–287, 1995. Disponível em: <https://doi.org/10.1080/08839519508945476>.

STACK Overflow Developer Survey 2023. Disponível em: <https://survey.stackoverflow.co/2023/>.

SWC Docs. Disponível em: <https://swc.rs/>.

VOBUGARI, N. et al. Advancements in oncology with artificial intelligence: A review article. *Cancers*, v. 14, n. 5, 2022. ISSN 2072-6694. Disponível em: <https://www.mdpi.com/2072-6694/14/5/1349>.