

Os Riscos da Dependência Cega de IA para o Desenvolvimento de Software

Aline Yuka Noguti¹, Eduardo Albuquerque¹, Vitor Tavares¹

¹Instituto Federal do Paraná - Campus Paranavaí - IFPR

1. Introdução

A Inteligência Artificial tem revolucionado diversas áreas da tecnologia, e o desenvolvimento de software não é exceção, com a popularização de assistentes de programação, como ChatGPT e Copilot, desenvolvedores podem gerar código rapidamente, otimizar tarefas repetitivas e solucionar problemas com mais eficiência. No entanto, essa facilidade levanta um debate importante: até que ponto a IA no desenvolvimento de software pode ser benéfica, e quando ela se torna um problema?

Este trabalho explora os riscos da utilização indiscriminada da IA como ferramenta principal no desenvolvimento de software, especialmente quando usada por indivíduos sem conhecimento técnico suficiente para validar as soluções propostas, por meio de um estudo de caso prático, analisaremos o desenvolvimento de um scanner de IP para identificar portas abertas e protocolos de rede, feito por um usuário sem experiência prévia na área de redes.

A criação de um sistema sem um planejamento adequado, validação técnica e compreensão sólida de seus fundamentos pode resultar em código ineficiente, inseguro e difícil de manter e à medida que o projeto se torna mais complexo, esses problemas tendem a se agravar, comprometendo não apenas seu desempenho, mas também sua confiabilidade e escalabilidade.

Com isso pretende-se demonstrar como a aceitação cega de códigos gerados por IA pode resultar em problemas que o próprio programador não consegue solucionar, criando um ciclo vicioso de dependência, a pesquisa destaca tanto os benefícios da IA no desenvolvimento de software quanto os riscos associados ao seu uso inconsequente, evidenciando a importância do conhecimento técnico para interpretar, validar e corrigir o código gerado.

2. Metodologia da Análise

A análise foi conduzida com o objetivo de identificar os impactos da dependência da Inteligência Artificial (IA) no desenvolvimento de software, avaliando tanto os benefícios quanto os desafios enfrentados por um usuário sem conhecimento prévio da área de redes ao construir um scanner de IP com auxílio da IA.

2.1. Monitoramento das Interações

Um dos membros da equipe foi responsável por acompanhar o participante que utilizou a IA para desenvolver o sistema. Durante o desenvolvimento, foram registradas todas as interações realizadas com a IA, incluindo:

- As perguntas feitas pelo participante.
- Os códigos gerados pela IA.
- As edições aplicadas pelo participante nos códigos gerados.
- As dificuldades enfrentadas durante o processo.

O processo de tomada de decisão do participante foi analisado, observando quando e como ele aceitava, modificava ou rejeitava as respostas fornecidas pela IA. Interações problemáticas foram documentadas, destacando casos onde o código gerado pela IA apresentou:

- Erros de funcionamento.
- Falhas de segurança.
- Inconsistências no código.
- Dificuldades de implementação.

2.2. Registro de Resultados e Problemas Identificados

As principais dificuldades encontradas pelo participante foram registradas, incluindo:

- Erros de funcionamento do código gerado.
- Falhas na interpretação do código fornecido pela IA.
- Necessidade de múltiplas interações para resolver um único problema.

Foi avaliada a capacidade do participante de compreender e corrigir os erros por conta própria ou se ele precisou recorrer novamente à IA, criando um ciclo de dependência.

2.3. Análise Externa e Conclusões

Após o período de desenvolvimento e monitoramento, um terceiro membro da equipe, que não participou diretamente da criação do código nem do acompanhamento, realizou uma análise crítica aprofundada. Esse membro avaliou os dados coletados, buscando:

- Padrões de dependência da IA.
- Impactos na qualidade do código gerado.
- Desafios enfrentados pelo participante durante o desenvolvimento.

A análise buscou extrair insights sobre os benefícios proporcionados pela IA, mirando em aspectos específicos como por exemplo:

- Agilidade no desenvolvimento.
- Auxílio na solução de problemas.

E a análise buscou identificar possíveis problemas da utilização da ferramenta de forma indiscriminada pelo desenvolvedor, focando principalmente nos seguintes aspectos:

- Falta de compreensão do código gerado.
- Riscos de segurança e manutenção associados ao uso da IA.

A análise final relacionou esses aspectos com conceitos da disciplina de Projeto Avançado de Software, destacando a importância de um desenvolvimento estruturado e da validação técnica no uso de IA. Essa abordagem permitiu uma avaliação detalhada dos efeitos da dependência da IA no desenvolvimento de software, oferecendo uma visão equilibrada entre seus benefícios e os riscos associados ao seu uso sem conhecimento técnico adequado.

3. Prompts e Respostas

3.1. Prompt: "Criar um script básico em Python para aceitar um endereço IP ou domínio como entrada".

```
1 def main():
2     target = input("Digite o endereço IP ou domínio para
3         escanear: ")
4     if not target.strip():
5         print("Erro: Nenhum endereço IP ou domínio foi
6             fornecido.")
7         return
8     print(f"Alvo recebido: {target}")
9
10 if __name__ == "__main__":
11     main()
```

3.2. Prompt: "O que significa e para que serve o .strip()"

Resposta: O método .strip() em Python é usado para remover espaços em branco (ou outros caracteres especificados) do início e do fim de uma string.

3.3. Prompt: "Trabalhar apenas com IP"

```
1 import ipaddress
2
3 def main():
4     target = input("Digite o endereço IP para escanear:
5         ")
6     if not target.strip():
7         print("Erro: Nenhum endereço IP foi fornecido.")
8         return
9     try:
10         ipaddress.ip_address(target)
11         print(f"Alvo recebido: {target}")
12     except ValueError:
13         print("Erro: O endereço fornecido não é um IP
14             válido.")
15
16 if __name__ == "__main__":
17     main()
```

3.4. Prompt: "Como integrar isso: 'Usar socket para verificar quais portas estão abertas. Integrar a biblioteca nmap para identificar os serviços ativos.'"

```
1 import ipaddress
2 import socket
3 import nmap
4
```

```

5 def scan_ports_with_socket(target):
6     print("\n[+] Escaneando portas abertas com socket...")
7     open_ports = []
8     for port in range(1, 1025): # Escaneia as portas de
9         try:
10             with socket.socket(socket.AF_INET, socket.
11                 SOCK_STREAM) as s:
12                 s.settimeout(0.5) # Define um timeout
13                     para evitar travamentos
14                 if s.connect_ex((target, port)) == 0:
15                     open_ports.append(port)
16                     print(f"[+] Porta {port} aberta")
17             except Exception as e:
18                 print(f"Erro ao escanear a porta {port}: {e}")
19
20     return open_ports
21
22 def scan_services_with_nmap(target):
23     print("\n[+] Identificando servicos ativos com nmap
24         ...")
25     nm = nmap.PortScanner()
26     try:
27         nm.scan(hosts=target, arguments='-sV') #
28             Escaneia servicos com deteccao de versao
29         for host in nm.all_hosts():
30             print(f"\nHost: {host} ({nm[host].hostname()
31                 })")
32             print(f"Estado: {nm[host].state()}")
33             for proto in nm[host].all_protocols():
34                 print(f"\nProtocolo: {proto}")
35                 ports = nm[host][proto].keys()
36                 for port in ports:
37                     service = nm[host][proto][port]
38                     print(f"Porta: {port}\tEstado: {
39                         service['state']}\tServico: {
40                             service['name']}")
41         except Exception as e:
42             print(f"Erro ao usar nmap: {e}")
43
44 def main():
45     # Solicita ao usuario um endereco IP
46     target = input("Digite o endereco IP para escanear:
47         ")
48
49

```

```

40     # Valida se o usuario forneceu uma entrada
41     if not target.strip():
42         print("Erro: Nenhum endereco IP foi fornecido.")
43         return
44
45     # Valida se o endereco fornecido e um IP valido
46     try:
47         ipaddress.ip_address(target)
48         print(f"Alvo recebido: {target}")
49
50         # Escaneia portas abertas com socket
51         open_ports = scan_ports_with_socket(target)
52
53         # Identifica servicos ativos com nmap
54         if open_ports:
55             scan_services_with_nmap(target)
56         else:
57             print("Nenhuma porta aberta encontrada para
58                 escanear servicos.")
59     except ValueError:
60         print("Erro: O endereco fornecido nao e um IP
61             valido.")
62
63 if __name__ == "__main__":
64     main()

```

3.5. Prompt: "O que seria a biblioteca Socket?"

Resposta: Boa pergunta! A biblioteca socket em Python é um módulo que permite a comunicação entre computadores em uma rede — seja local (LAN) ou pela internet — usando os protocolos TCP/IP ou UDP.

3.6. Prompt: "Por que o processo de leitura das portas está demorando tanto?"

Resposta: Devido ao tempo de leitura em cada porta, `settimeout(0.5)` por porta causa lentidão. Sugestão de usar 0.1, com risco de falsos negativos.

3.7. Prompt: *Erro sobre o nmap não encontrado* "Por que esse erro? Como resolver?"

Resposta: necessário instalar o Nmap no Windows e garantir que esteja no PATH.

3.8. Prompt: "Como fazer para aceitar IPV6?"

```

1     import ipaddress
2     import socket
3     import nmap
4
5     def scan_ports_with_socket(target):

```

```

6 print("\n[+] Escaneando portas abertas com socket
   ...")
7 open_ports = []
8 try:
9     # Detecta automaticamente se o alvo e IPv4 ou
      IPv6
10    family = socket.AF_INET6 if ':' in target else
      socket.AF_INET
11    for port in range(22, 91): # Escaneia as
      portas de 22 a 89
12        try:
13            with socket.socket(family, socket.
      SOCK_STREAM) as s:
14                s.settimeout(0.5) # Define um
      timeout para evitar travamentos
15                if s.connect_ex((target, port)) ==
      0:
16                    open_ports.append(port)
17                    print(f"[+] Porta {port}
      aberta")
18            except Exception as e:
19                print(f"Erro ao escanear a porta {port
      }: {e}")
20    except Exception as e:
21        print(f"Erro ao configurar o socket: {e}")
22    return open_ports
23
24 def scan_services_with_nmap(target):
25     print("\n[+] Identificando servicos ativos com
      nmap...")
26     nm = nmap.PortScanner()
27     try:
28         # Adiciona o suporte a IPv6 com o argumento -6
29         nm.scan(hosts=target, arguments='-sV --version
      -intensity 5 -p 22 -6')
30         for host in nm.all_hosts():
31             print(f"\nHost: {host} ({nm[host].hostname
      ()})")
32             print(f"Estado: {nm[host].state()}")
33             for proto in nm[host].all_protocols():
34                 print(f"\nProtocolo: {proto}")
35                 ports = nm[host][proto].keys()
36                 for port in ports:
37                     service = nm[host][proto][port]
38                     print(f"Porta: {port}\tEstado: {
      service['state']}\tServico: {

```

```

39         service['name']})
40     except Exception as e:
41         print(f"Erro ao usar nmap: {e}")
42
43 def main():
44     # Solicita ao usuario um endereco IP
45     target = input("Digite o endereco IP para escanear
46                   : ")
47
48     # Valida se o usuario forneceu uma entrada
49     if not target.strip():
50         print("Erro: Nenhum endereco IP foi fornecido.
51               ")
52         return
53
54     # Valida se o endereco fornecido e um IP valido
55     try:
56         ipaddress.ip_address(target)
57         print(f"Alvo recebido: {target}")
58
59         # Escaneia portas abertas com socket
60         open_ports = scan_ports_with_socket(target)
61
62         # Identifica servicos ativos com nmap
63         if open_ports:
64             scan_services_with_nmap(target)
65         else:
66             print("Nenhuma porta aberta encontrada
67                   para escanear servicos.")
68     except ValueError:
69         print("Erro: O endereco fornecido nao e um IP
70               valido.")
71
72 if __name__ == "__main__":
73     main()

```

3.9. Prompt: "Por que o nmap não está identificando o serviço correto na porta?"

```

1
2 import ipaddress
3 import socket
4 import nmap
5
6 def scan_ports_with_socket(target):
7     print("\n[+] Escaneando portas abertas com socket
8           ...")
9     open_ports = []

```

```

9      try:
10          # Detecta automaticamente se o alvo e IPv4 ou
              IPv6
11          family = socket.AF_INET6 if ':' in target else
              socket.AF_INET
12          for port in range(22, 91): # Escaneia as
              portas de 22 a 89
13              try:
14                  with socket.socket(family, socket.
                      SOCK_STREAM) as s:
15                      s.settimeout(0.5) # Define um
                          timeout para evitar travamentos
16                      if s.connect_ex((target, port)) ==
                          0:
17                          open_ports.append(port)
18                          print(f"[+] Porta {port}
                              aberta")
19                      except Exception as e:
20                          print(f"Erro ao escanear a porta {port
                              }: {e}")
21          except Exception as e:
22              print(f"Erro ao configurar o socket: {e}")
23          return open_ports
24
25 def scan_services_with_nmap(target):
26     print("\n[+] Identificando servicos ativos com
        nmap...")
27     nm = nmap.PortScanner()
28     try:
29         # Adiciona o suporte a IPv6 com o argumento -6
30         nm.scan(hosts=target, arguments='-sV --version
            -intensity 5 -p 22 -6')
31         for host in nm.all_hosts():
32             print(f"\nHost: {host} ({nm[host].hostname
                ()})")
33             print(f"Estado: {nm[host].state()}")
34             for proto in nm[host].all_protocols():
35                 print(f"\nProtocolo: {proto}")
36                 ports = nm[host][proto].keys()
37                 for port in ports:
38                     service = nm[host][proto][port]
39                     print(f"Porta: {port}\tEstado: {
                        service['state']}\tServico: {
                            service['name']}")
40         except Exception as e:
41             print(f"Erro ao usar nmap: {e}")

```



```

42
43 def main():
44     # Solicita ao usuario um endereco IP
45     target = input("Digite o endereco IP para escanear
46                   : ")
47
48     # Valida se o usuario forneceu uma entrada
49     if not target.strip():
50         print("Erro: Nenhum endereco IP foi fornecido.
51               ")
52         return
53
54     # Valida se o endereco fornecido e um IP valido
55     try:
56         ipaddress.ip_address(target)
57         print(f"Alvo recebido: {target}")
58
59         # Escaneia portas abertas com socket
60         open_ports = scan_ports_with_socket(target)
61
62         # Identifica servicos ativos com nmap
63         if open_ports:
64             scan_services_with_nmap(target)
65         else:
66             print("Nenhuma porta aberta encontrada
67                   para escanear servicos.")
68     except ValueError:
69         print("Erro: O endereco fornecido nao e um IP
70               valido.")
71
72 if __name__ == "__main__":
73     main()

```

3.10. Prompt: "Caso eu queira criar um arquivo .csv ou .json contendo os resultados da varredura o que eu teria que fazer?"

```

1     import json
2
3 def save_results_to_json(filename, results):
4     try:
5         with open(filename, mode='w', encoding='utf-8')
6             as file:
7             json.dump(results, file, indent=4)
8             print(f"Resultados salvos em {filename}")
9     except Exception as e:
10        print(f"Erro ao salvar resultados no JSON: {e}")

```

3.11. Prompt: "O que é o json.dump e para que serve?"

O json.dump() é uma função da biblioteca json do Python que serve para salvar dados em formato JSON dentro de um arquivo.

3.12. Prompt: *Erro: No module nmap* "O que poderia ser esse erro?"

Resposta: verificar instalação da lib python-nmap, usar pip show para checar, ou reinstalar.

3.13. Prompt: "O resultado das portas abertas está dando a 22 e a 90, porém os resultados dos serviços ativos não batem com as portas."

Resposta: Ele gerou sugestões, mas também causou problemas posteriores de leitura.

3.14. Prompt: No caso de eu querer scanear as portas com o socket para passar a lista de portas abertas para scanear os serviços com o nmap, o que eu devo fazer?"

Resposta: Copilot sugeriu código que recebe portas abertas via socket e passa para o Nmap o que chegou no código final elaborado.

4. Falhas no Código

4.1. Erro na Leitura de Serviços com Nmap

O scanner apresentou falhas ao identificar serviços utilizando o nmap, resultando no seguinte erro:

```
1 Nmap scan info: {'error': ['2804:360:40:2188:19ba:9b3:e4d5:fa89 looks like an IPv6
2 target specification -- you have to use the -6 option.\r
  \nWARNING: No targets were
3 specified, so 0 hosts scanned.\r\n'], 'warning': ['
  WARNING: No targets were specified,
4 so 0 hosts scanned.\r\n'], 'tcp': {'method': 'syn', '
  services': '22,90'}}
```

Causa: O script inicial não identificava corretamente se o alvo era um endereço IPv6, e o nmap exigia a flag -6 para escanear corretamente.

Solução: Foi adicionada uma verificação que identifica se o alvo é IPv6 e adiciona automaticamente a flag -6 ao comando nmap.

```
1 is_ipv6 = ':' in target
2 arguments = f"-sV --version-intensity 9 -p {
  ports_to_scan}"
3 if is_ipv6:
4     arguments += " -6" # Adiciona a flag para IPv6
5
6 nm.scan(hosts=target, arguments=arguments)
```

4.2. Tratamento de IPv6

O script inicial não tratava adequadamente alvos IPv6, resultando em falha na execução do nmap e no escaneamento de portas com socket.

Causa: O socket por padrão usa AF_INET (IPv4). Para suportar IPv6, era necessário modificar a família de endereço para AF_INET6 quando um endereço IPv6 fosse identificado.

Solução: O código foi ajustado para identificar automaticamente o tipo de IP e utilizar AF_INET6 para IPv6.

```
1 family = socket.AF_INET6 if ':' in target else socket.  
  AF_INET  
2  
3 with socket.socket(family, socket.SOCK_STREAM) as s:  
4     s.settimeout(0.5)  
5     if s.connect_ex((target, port)) == 0:  
6         open_ports.append(port)
```

4.3. Falha no Salvamento de Resultados

Inicialmente, o scanner não salvava os resultados do escaneamento rápido, apenas os do escaneamento detalhado.

Causa: O salvamento estava implementado apenas na função de escaneamento detalhado.

Solução: Foi adicionada a função save_results_to_json() também ao escaneamento rápido, armazenando as portas abertas em um arquivo JSON.

```
1 def quick_scan():  
2     open_ports = scan_ports_with_socket(target)  
3     if open_ports:  
4         results = [{"port": port, "state": "open"} for  
5                     port in open_ports]  
6         save_results_to_json("scan_results.json",  
7                               results)
```

4.4. Validação de Entrada do Usuário

Se o usuário fornecesse um IP inválido, o programa poderia travar ou retornar erros inesperados.

Causa: O script original não validava corretamente se o IP fornecido era válido antes de tentar escaneá-lo.

Solução: Foi adicionada uma função validate_ip() para verificar a validade do endereço antes de iniciar o escaneamento.

```

1 def validate_ip(target):
2     try:
3         ipaddress.ip_address(target)
4         return True
5     except ValueError:
6         print("Erro: O endereço fornecido não é um IP válido.")
7         return False
8
9 target = input("Digite o endereço IP: ").strip()
10 if not validate_ip(target):
11     exit()

```

4.5. Melhorias no Menu Interativo

Um menu interativo foi solicitado para permitir que o usuário escolha entre um escaneamento rápido e um escaneamento detalhado.

Solução: O menu foi implementado com as opções "Escaneamento rápido" e "Escaneamento detalhado". O escaneamento rápido detecta portas abertas, enquanto o detalhado também identifica serviços ativos nas portas encontradas.

```

1 def main():
2     while True:
3         print("\n=== Menu de Escaneamento ===")
4         print("1. Escaneamento rápido")
5         print("2. Escaneamento detalhado")
6         print("3. Sair")
7
8         choice = input("Escolha uma opção: ").strip()
9
10        if choice == "1":
11            quick_scan()
12        elif choice == "2":
13            detailed_scan()
14        elif choice == "3":
15            print("\n[+] Saindo do programa. Até logo!")
16            break
17        else:
18            print("\n[!] Opção inválida. Tente novamente.")

```

5. Problemas de Segurança

5.1. Uso do nmap sem Filtragem Adequada

O código abaixo não valida nem sanitiza a entrada antes de construir o comando nmap. Isso pode ser explorado por um atacante para injetar parâmetros maliciosos.

```

1 arguments = f'-sV --version-intensity 9 -p {
  ports_to_scan}'
2 if is_ipv6:
3     arguments += ' -6' # Add the IPv6 flag if the
      target is IPv6
4 print(f"Comando Nmap: {arguments}")
5 nm.scan(hosts=target, arguments=arguments)

```

5.2. Exposição de Resultados Sensíveis

O código imprime informações sensíveis sobre os hosts e serviços escaneados diretamente no console, o que pode expor dados críticos caso o sistema seja acessado por terceiros.

```

1 print(f"Nmap scan info: {nm.scaninfo()}")
2 print(f"Nmap all hosts: {nm.all_hosts()}")

```

5.3. Falta de Rate Limiting

O código escaneia portas rapidamente, sem implementar uma pausa entre as requisições. Isso pode ser interpretado como um ataque de varredura (brute force) e pode fazer com que o sistema seja bloqueado ou marcado como um atacante.

```

1 for port in range(22, 91): # Escaneia as portas de 22 a
  89
2     try:
3         with socket.socket(family, socket.SOCK_STREAM)
          as s:
4             s.settimeout(0.5) # Define um timeout para
              evitar travamentos
5             if s.connect_ex((target, port)) == 0:
6                 open_ports.append(port)
7                 print(f"[+] Porta {port} aberta")
8     except Exception as e:
9         print(f"Erro ao escanear a porta {port}: {e}")

```

5.4. Falta de Registro de Erros

Embora o código capture erros, ele simplesmente imprime as falhas na tela, sem um sistema de registro de logs adequado. Isso dificulta a depuração e o monitoramento do sistema em produção.

```

1 print(f"Erro ao usar nmap: {e}")
2 print(f"Erro ao salvar resultados no JSON: {e}")

```

6. Análise dos Resultados

6.1. Comportamento do Desenvolvedor Durante o Uso da IA

Ao avaliar os prompts, é possível observar que o desenvolvedor inicialmente demonstrava interesse em compreender o funcionamento do código gerado pela IA, sendo evidenciado

por perguntas como "O que significa e para que serve o `.strip()`?" e "O que seria a biblioteca `Socket`?", indicando uma tentativa inicial de aprendizado e entendimento das ferramentas e conceitos apresentados.

No entanto, à medida que o projeto avançava e se tornava mais complexo, o comportamento do desenvolvedor mudou, ele passou a confiar mais na IA para tomar decisões e resolver problemas, muitas vezes aceitando o código gerado sem questionar ou compreender completamente seu funcionamento. Isso é especialmente evidente em momentos em que o desenvolvedor enfrentava dificuldades, como na integração do suporte a IPv6 ou na identificação de serviços ativos com o `nmap`. Nessas situações, o desenvolvedor recorria à IA para obter soluções prontas, sem buscar entender os detalhes técnicos ou as implicações do código gerado.

Essa mudança de comportamento, mesmo que em uma pequena escala, reflete uma dependência crescente da ferramenta, o que pode ser problemático em projetos mais avançados, pois a falta de compreensão do código gerado pode levar a erros ocultos, dificuldades na manutenção e riscos de segurança. Além disso, a confiança excessiva na IA pode limitar o desenvolvimento das habilidades técnicas do desenvolvedor, criando um ciclo de dependência que dificulta a resolução de problemas de forma autônoma.

6.2. Vantagens da Utilização da IA no Desenvolvimento de Software

- **Aceleração do Desenvolvimento:** Ferramentas de IA podem gerar código rapidamente, economizando tempo dos desenvolvedores.
- **Redução de Erros Comuns:** Muitos erros sintáticos e padrões de código ruins podem ser evitados com sugestões baseadas em IA.
- **Auxílio na Aprendizagem:** Desenvolvedores iniciantes podem usar IA para compreender padrões e boas práticas de programação.
- **Automação de Tarefas Repetitivas:** Testes automatizados, geração de documentação e refatoração de código podem ser otimizados.
- **Melhoria na Manutenção do Código:** Ferramentas de IA podem sugerir melhorias para aumentar a eficiência e a legibilidade do código.

6.3. Desvantagens da Dependência Excessiva

- **Código sem Compreensão:** Desenvolvedores podem adotar código gerado sem entender seu funcionamento, levando a falhas ocultas.
- **Falsa Segurança:** Acreditar que a IA sempre gera código correto pode resultar em vulnerabilidades de segurança e falhas de execução.
- **Repetição de Padrões Ruins:** Se a IA for treinada em código de baixa qualidade, ela pode perpetuar más práticas e padrões inseguros.
- **Problemas de Generalização:** A IA pode não entender contextos específicos, gerando soluções inadequadas ou ineficazes.
- **Dependência Exagerada:** Equipes podem perder habilidades essenciais, confiando exclusivamente na IA para decisões técnicas.

7. Conclusão

A Inteligência Artificial é, sem dúvida alguma, uma ferramenta poderosa que está transformando o desenvolvimento de software e sua capacidade de gerar código rapidamente,

automatizar tarefas repetitivas e apoiar em soluções para problemas complexos pode agilizar significativamente o trabalho dos desenvolvedores. No entanto, o uso inconsequente e inconsciente dessa tecnologia pode criar uma divisão clara entre bons programadores e programadores dependentes.

Bons programadores, que possuem conhecimento técnico sólido e a capacidade de avaliar criticamente o código gerado pela IA, conseguem aproveitar ao máximo os benefícios dessa ferramenta, eles utilizam a IA como um suporte para aumentar sua produtividade, mantendo o controle sobre as decisões técnicas e garantindo a qualidade e a segurança do código, por outro lado, programadores que dependem exclusivamente da IA, sem compreender o que está sendo gerado, correm o risco de produzir sistemas frágeis, com falhas ocultas e vulnerabilidades de segurança.

Essa diferença será um fator determinante no mercado de trabalho e empresas não precisarão mais de programadores fracos para realizar atividades simples, pois um único bom programador, com a IA ao seu lado, pode realizar o trabalho de vários desses profissionais em um tempo muito menor, o que resultará em economia de tempo e dinheiro para as empresas, tornando o domínio técnico e a capacidade de usar a IA de forma consciente um diferencial competitivo essencial.

Portanto, o futuro do desenvolvimento de software não está apenas no uso da IA, mas na combinação de conhecimento técnico, pensamento crítico e habilidade para avaliar e validar o que a ferramenta gera. Programadores que dominarem essa combinação estarão melhor posicionados para se destacar em um mercado cada vez mais exigente e competitivo.