

Generative AI e LLMs: ChatGPT e Copilot

Gabriel Oliveira do Espírito Santo



Pair Programming

- Prática comum entre devs;
- Devs trabalhando juntos no mesmo computador;
- Traz novas perspectivas de resolução de problemas;
- Promove aprendizagem, transferência de conhecimento e criatividade;
- Muitas vezes consegue gerar código de melhor qualidade e com menos erros;
- Consegue ajudar em comportamentos transversais como comunicação;



Ferramentas

Pair Programming com IA

ChatGPT



Bing Chat



Github Copilot



—

Benefícios de pairar com uma IA

Pair Programming com IA

- Aumento de produtividade
- Soluções mais específicas para o problema
- Agilidade no desenvolvimento, dentre elas:
 - ◆ Solução de bugs
 - ◆ Redução da curva de aprendizado
 - ◆ Geração de código



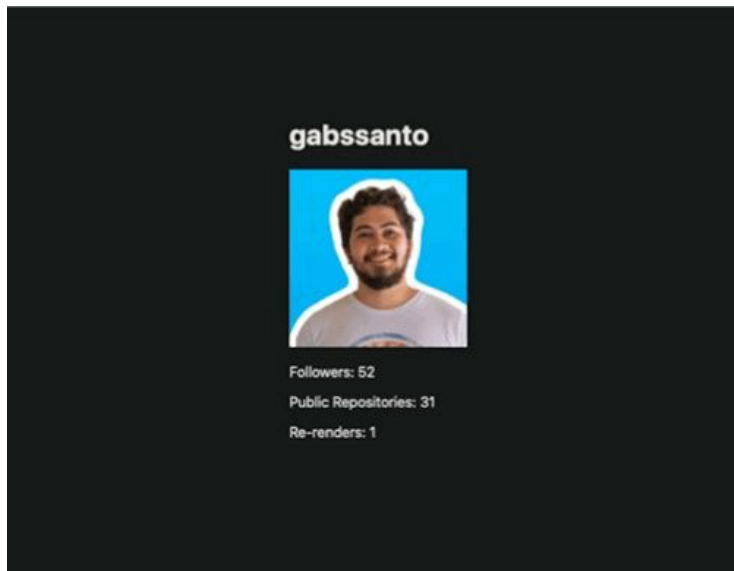
Exemplos de Pair Programming com IA



Exemplo de Pair Programming com IA

1. Solução de Bugs

- **Problema:** Loop Infinito, o contador soma várias vezes por segundo



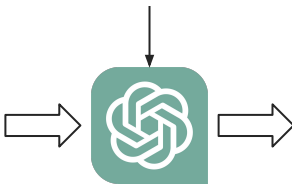
```
1 import React, { useState, useEffect } from 'react';
2
3 export function App() {
4   const [counter, setCounter] = useState(0);
5   const [userData, setUserData] = useState(null);
6
7   useEffect(() => {
8     const fetchData = async () => {
9       try {
10         const response = await fetch('https://api.github.com/users/gabssanto');
11         const data = await response.json();
12         setUserData(data);
13         setCounter(counter + 1);
14       } catch (error) {
15         console.log('Erro ao buscar dados:', error);
16       }
17     };
18
19     fetchData();
20
21     const interval = setInterval(fetchData, 500);
22
23     return () => clearInterval(interval);
24   });
25
26   if (!userData) {
27     return <div>Carregando ... </div>;
28   }
29
30   return (
31     <div>
32       <h1>{userData.login}</h1>
33       <img src={userData.avatar_url} alt={userData.login} width="200" />
34       <p>Seguidores: {userData.followers}</p>
35       <p>Repositórios Públicos: {userData.public_repos}</p>
36       Renderizações: {counter}
37     </div>
38   );
39 }
```

```

1 import React, { useState, useEffect } from 'react';
2
3 export function App() {
4   const [counter, setCounter] = useState(0);
5   const [userData, setUserData] = useState(null);
6
7   useEffect(() => {
8     const fetchData = async () => {
9       try {
10         const response = await fetch('https://api.github.com/users/gabssanto');
11         const data = await response.json();
12         setUserData(data);
13         setCounter(counter + 1);
14       } catch (error) {
15         console.log('Erro ao buscar dados:', error);
16       }
17     };
18     fetchData();
19
20     const interval = setInterval(fetchData, 500);
21
22     return () => clearInterval(interval);
23   });
24
25   if (!userData) {
26     return <div>Carregando ... </div>;
27   }
28
29   return (
30     <div>
31       <h1>{userData.login}</h1>
32       <img src={userData.avatar_url} alt={userData.login} width="200" />
33       <p>Seguidores: {userData.followers}</p>
34       <p>Repositórios Públicos: {userData.public_repos}</p>
35       Renderizações: {counter}
36     </div>
37   );
38 }

```

*This code I'll send
has a bug in it, can
you fix it for me?*



```

1 export function App() {
2   const [counter, setCounter] = useState(0);
3   const [userData, setUserData] = useState(null);
4
5   useEffect(() => {
6     const fetchData = async () => {
7       try {
8         const response = await fetch('https://api.github.com/users/gabssanto');
9         const data = await response.json();
10        setUserData(data);
11        setCounter((prevCounter) => prevCounter + 1); // Fix: Update counter using functional form
12      } catch (error) {
13        console.log('Erro ao buscar dados:', error);
14      }
15    };
16    fetchData();
17
18    const interval = setInterval(fetchData, 500);
19
20    return () => clearInterval(interval);
21  }, []); // Fix: Empty dependency array to run effect only once
22
23  if (!userData) {
24    return <div>Carregando ... </div>;
25  }
26
27  return (
28    <div>
29      <h1>{userData.login}</h1>
30      <img src={userData.avatar_url} alt={userData.login} width="200" />
31      <p>Seguidores: {userData.followers}</p>
32      <p>Repositórios Públicos: {userData.public_repos}</p>
33      Renderizações: {counter}
34    </div>
35  );
36 }

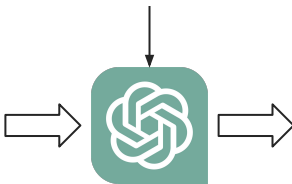
```

```

1 import React, { useState, useEffect } from 'react';
2
3 export function App() {
4   const [counter, setCounter] = useState(0);
5   const [userData, setUserData] = useState(null);
6
7   useEffect(() => {
8     const fetchData = async () => {
9       try {
10         const response = await fetch('https://api.github.com/users/gabssanto');
11         const data = await response.json();
12         setUserData(data);
13         setCounter(counter + 1);
14       } catch (error) {
15         console.log('Erro ao buscar dados:', error);
16       }
17     };
18     fetchData();
19
20     const interval = setInterval(fetchData, 500);
21
22     return () => clearInterval(interval);
23   });
24
25   if (!userData) {
26     return <div>Carregando ... </div>;
27   }
28
29   return (
30     <div>
31       <h1>{userData.login}</h1>
32       <img src={userData.avatar_url} alt={userData.login} width="200" />
33       <p>Seguidores: {userData.followers}</p>
34       <p>Repositórios Públicos: {userData.public_repos}</p>
35       Renderizações: {counter}
36     </div>
37   );
38 }

```

*This code I'll send
has a bug in it, can
you fix it for me?*



```

1 export function App() {
2   const [counter, setCounter] = useState(0);
3   const [userData, setUserData] = useState(null);
4
5   useEffect(() => {
6     const fetchData = async () => {
7       try {
8         const response = await fetch('https://api.github.com/users/gabssanto');
9         const data = await response.json();
10        setUserData(data);
11        setCounter((prevCounter) => prevCounter + 1); // Fix: Update counter using functional form
12      } catch (error) {
13        console.log('Erro ao buscar dados:', error);
14      }
15    };
16    fetchData();
17
18    const interval = setInterval(fetchData, 500);
19
20    return () => clearInterval(interval);
21  });
22
23  if (!userData) {
24    return <div>Carregando ... </div>;
25  }
26
27  return (
28    <div>
29      <h1>{userData.login}</h1>
30      <img src={userData.avatar_url} alt={userData.login} width="200" />
31      <p>Seguidores: {userData.followers}</p>
32      <p>Repositórios Públicos: {userData.public_repos}</p>
33      Renderizações: {counter}
34    </div>
35  );
36 }

```

Here's what has been fixed:

The counter state is updated using the functional form `setCounter((prevCounter) => prevCounter + 1)` instead of `setCounter(counter + 1)`. This ensures that the updated value is based on the previous value of the counter state correctly.

The dependency array in the `useEffect` hook is now empty (`[]`). This ensures that the effect runs only once when the component mounts, instead of repeatedly running every time the component re-renders. The interval setup in the effect will still work as intended.

2. Geração de Código – Copilot

- Alimentado pelo Codex da OpenAI
- Ajuda na produtividade gerando pedaços de códigos
 - ◆ Facilita geração de regexes
 - ◆ Agiliza a tipagem em Typescript
- Desenvolvedor utiliza de forma mais imperativa a depender da linguagem/framework
 - ◆ Em ReactJS/Typescript é mais propositivo
 - ◆ Em RubyOnRails/Ruby é mais imperativo

```
1 # Gerar expectativa para uma URL que corresponda a users/gabs  
  santo?secret_key=, mas a chave secreta é variável  
2 expect(url).to match(/users\/gabssanto\?secret_key=(.*)/)
```

```
1 /**  
2  * Gerar uma interface para um componente React, que recebe  
  as seguintes props  
3  * O tipo pode ser primary, secondary ou tertiary  
4  * O tamanho pode ser small, medium ou large  
5  * O onClick pode ser indefinido  
6  */  
7 interface ButtonProps {  
8   type: 'primary' | 'secondary' | 'tertiary';  
9   size: 'small' | 'medium' | 'large';  
10  onClick?: () => void;  
11 }
```

2. Geração de Código – Chat GPT

- **Prompt 1:** Gere para mim um exemplo de uso do FabricJS para implementar um Quadrado e um Círculo usando o padrão Factory

```
1 // Primeiro, definimos a classe Factory
2 class ShapeFactory {
3   createShape(type, options) {
4     let shape;
5     switch (type) {
6       case 'square':
7         shape = new fabric.Rect(options);
8         break;
9       case 'circle':
10        shape = new fabric.Circle(options);
11        break;
12       default:
13         throw new Error('Tipo de forma inválido: ${type}');
14     }
15     return shape;
16   }
17 }
18
19 // Inicializamos nossa fábrica
20 const shapeFactory = new ShapeFactory();
```

2. Geração de Código – Chat GPT

→ **Prompt 2:** Estenda a classe Factory para incluir um triângulo



```
1 class ShapeFactory {  
2     ...  
3     case 'triangle':  
4         shape = new fabric.Triangle(options);  
5         break;  
6     ...  
7 }
```

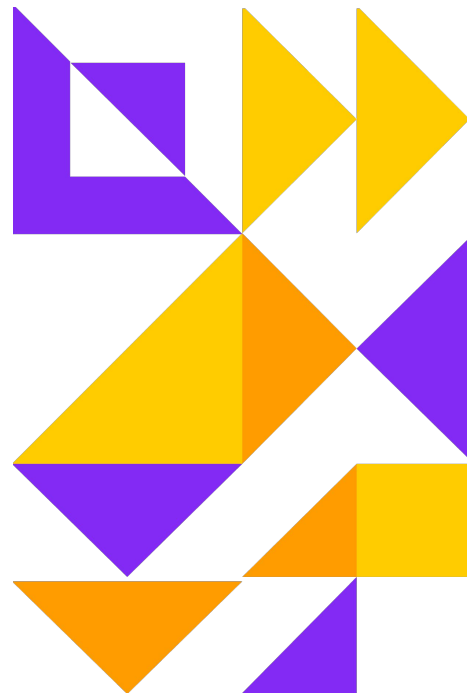
3. Redução da Curva de Aprendizado

- Mudança de projeto
 - ◆ Não estava familiarizado com a base de código, linguagem e framework.
 - ◆ Era o único desenvolvedor trabalhando no projeto.
 - ◆ Projeto antigo, difícil de encontrar soluções para alguns problemas.
- Uso de Chat GPT e Bing
 - ◆ Boa fonte de conhecimento para entender bases de códigos antigas.
 - ◆ Facilita a compreensão de convenções como de Ruby On Rails.
- Algumas vezes falam algo errado de forma muito convincente.
 - ◆ De volta a leitura de documentação
 - ◆ Stack Overflow
 - ◆ Pair programming.



Benefícios do Pair Programming com IA

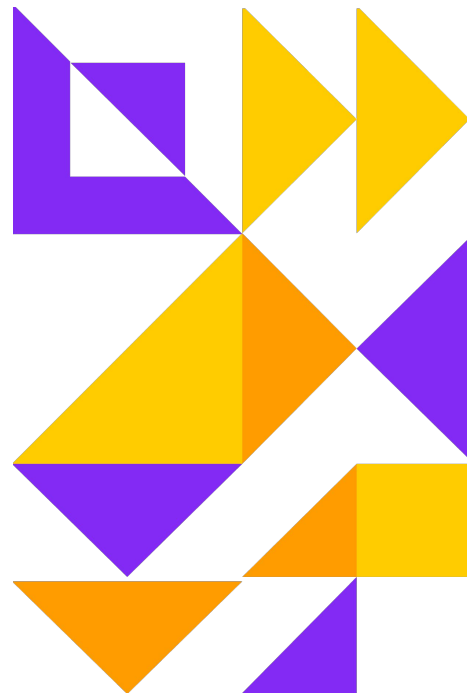
- GitHub Copilot é muito útil por estar já integrado ao editor de texto.
 - ◆ Muito útil na sugestão de tipos e regexes.
 - ◆ Atualmente um pouco mais limitado quando comparado a ferramentas como Chat GPT e Bing.
- Chat GPT e Bing Chat são mais versáteis
 - ◆ Tem potencial de reduzir a curva de aprendizado.
 - ◆ Facilitam a correção de bugs, fornecendo soluções mais específicas e muitas vezes corretas.
 - ◆ Podem produzir códigos complexos, seguindo inclusive Padrões de Design de Código.
 - ◆ Ferramentas muito eficazes para se debater sobre uma solução e melhorias em códigos.



“Com grandes poderes, vem grandes responsabilidades”

Considerações Éticas e Responsabilidade

- Desenvolvedores devem utilizar as ferramentas de IA de forma ética e com responsabilidade.
- O desenvolvedor é o piloto e a IA é o copiloto, com o desenvolvedor sempre no controle do código gerado.
- Copiar e colar código de empresas é dar acesso a código privado de uma empresa para outra.
- O desenvolvedor deve ser capaz de explicar a AI o contexto e a intenção pretendida, sem violar códigos privados.



Aplicações Futuras

- Chat GPT Plus, a versão paga do Chat GPT
 - ◆ Acesso antecipado ao GPT 4, com acesso a plugins e acesso a internet por parte do GPT 4 via Bing.
- Amazon CodeWhisperer
 - ◆ Assistente de codificação projetado para ajudar a desenvolver aplicações de forma rápida e segura.
- Google Bard
 - ◆ Projetado para ser a resposta do Google ao Bing Chat, usa uma versão leve e otimizada do LaMDA.
- Llama
 - ◆ LLM Open Source da Meta.
- GitHub Copilot X
 - ◆ Uma futura suite de aplicações fornecidas pelo GitHub que devem ajudar na criação de Pull Requests, geração de testes, etc.



Conclusão

- IA's são uma ótima ferramenta para ajudar no desenvolvimento
- Altíssimo potencial de melhorias para o futuro, muitas possibilidades de aplicação
- Cuidado com o uso ético das ferramentas
- O Desenvolvedor sempre deve ser o piloto, e a IA o copiloto



Valeu!

