

# SET - Software Estimation Tool



Aluno: Inácio Moraes  
Orientador: Cleiton Silva

# Objetivos do Projeto

---

01

Redução da subjetividade nas estimativas de esforço

02

Otimização do tempo e precisão no planejamento.

03

Aumento da precisão utilizando dados históricos do GitHub.

04

Estimativa gerada bem contextualizada.

# Escopo

01

Estimar tarefas individualmente utilizando inteligência artificial

02

Processar todo o backlog em lote e de forma eficiente

03

Sincronizar e utilizar dados históricos reais do GitHub

# Escopo

04

Gerar estimativas em horas/pontos e com nível de confiança

05

Oferecer raciocínio detalhado para cada resultado obtido

# Fora do Escopo

## Interface



Interface web ou mobile não será desenvolvida; apenas CLI

## Integrações



Não há integração com Jira ou Azure DevOps, somente GitHub

## AI



Não utiliza modelo de Machine Learning próprio; depende de API OpenAI

# Fora do Escopo

## Dados



Não oferece autenticação multi-usuário ou análise de desempenho

## Integrações



Estimativas dependem da qualidade dos dados históricos e API Keys

## API



Existem limitações de rate limits nas APIs utilizadas (GitHub e OpenAI)

# Atores



**Ana Rodrigues**  
Desenvolvedor Full  
Stack Sênior

---

Ana é desenvolvedora sênior em uma consultoria de software atende múltiplos clientes. Especialista em React, Node.js e Go. Lidera tecnicamente 2 desenvolvedores juniores e frequentemente é consultada para estimativas de tarefas complexas. Preocupa-se com a qualidade técnica e a precisão das entregas.

---



**Carlos Mendes**  
Scrum Master

---

Carlos é Scrum Master em uma startup de tecnologia com 15 desenvolvedores. Formado em Ciência da Computação, possui certificação PSM I e PSM II. Trabalha com 3 times de Scrum simultaneamente e é apaixonado por métricas e melhoria contínua. Valoriza a transparência e busca constantemente formas de otimizar o processo de desenvolvimento.

---



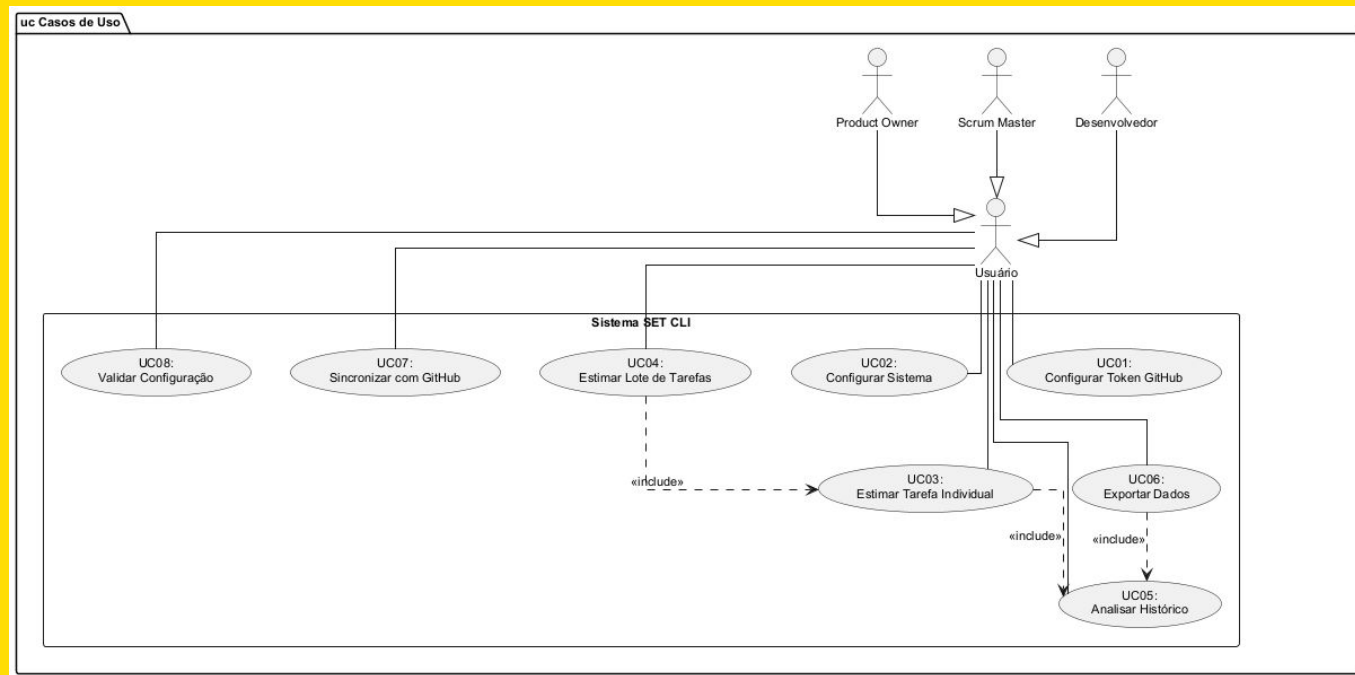
**Roberto Santos**  
Product Owner

---

Roberto é PO de um produto SaaS B2B com 50k+ usuários. Começou a carreira como desenvolvedor full-stack, trabalhou 10 anos com código antes de migrar para o time de produtos. Usa sua experiência técnica para fazer estimativas mais realistas e se comunicar melhor com a equipe de engenharia.

---

# Casos de Uso





# Funcionalidades

	Comando na CLI
Configurar Token GitHub	<i>set configure --github-token</i>
Configurar Sistema	<i>set configure --initial</i>
Estimar Tarifa Individual	<i>set estimate "&lt;descrição&gt;"</i>
Estimar Lote de Tarefas	<i>set batch --file &lt;arquivo&gt;</i>

# Funcionalidades

	Comando na CLI
Analisar Histórico	<i>set configure --github-token</i>
Exportar Dados	<i>set configure --initial</i>
Sincronizar com GitHub	<i>set estimate "&lt;descrição&gt;"</i>
Validar Configuração	<i>set batch --file &lt;arquivo&gt;</i>

# Necessidades e Funcionalidades

01

Configurar Token GitHub - *set configure --github-token*

02

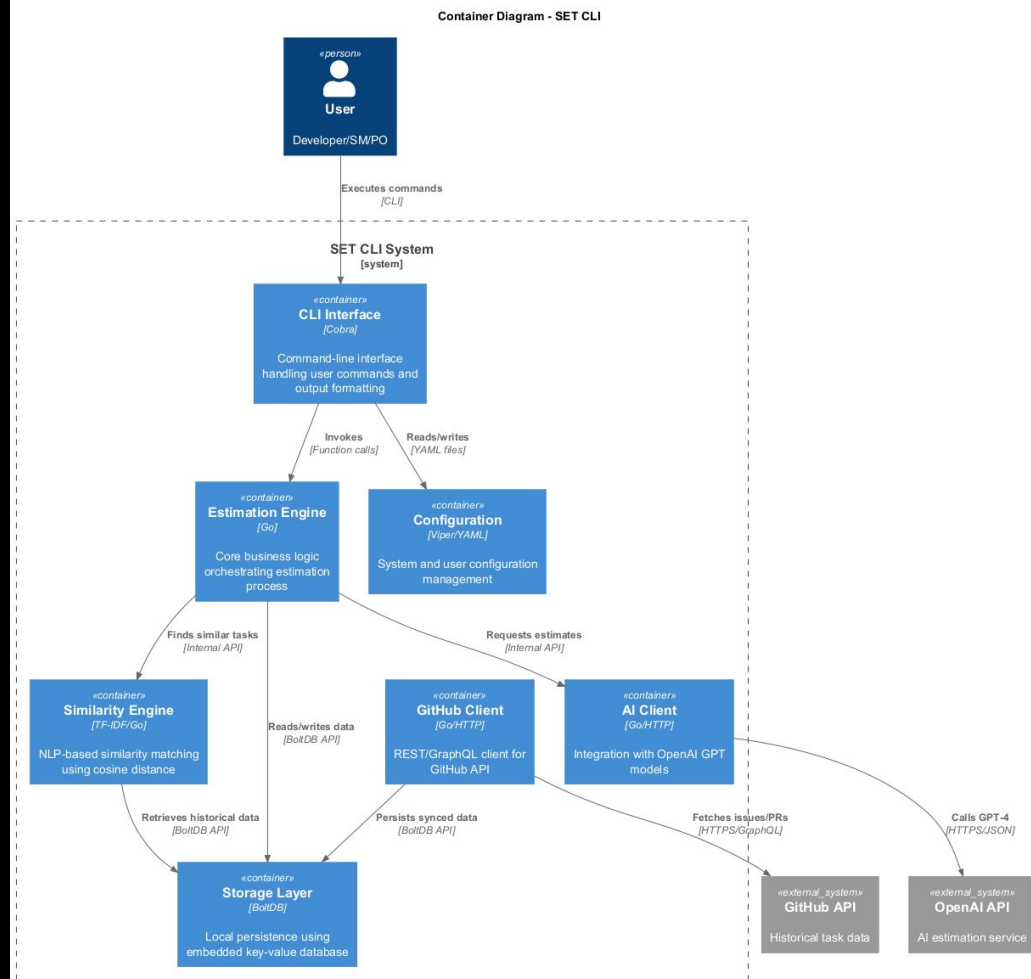
Processar todo o backlog em lote e de forma eficiente

03

Sincronizar e utilizar dados históricos reais do GitHub

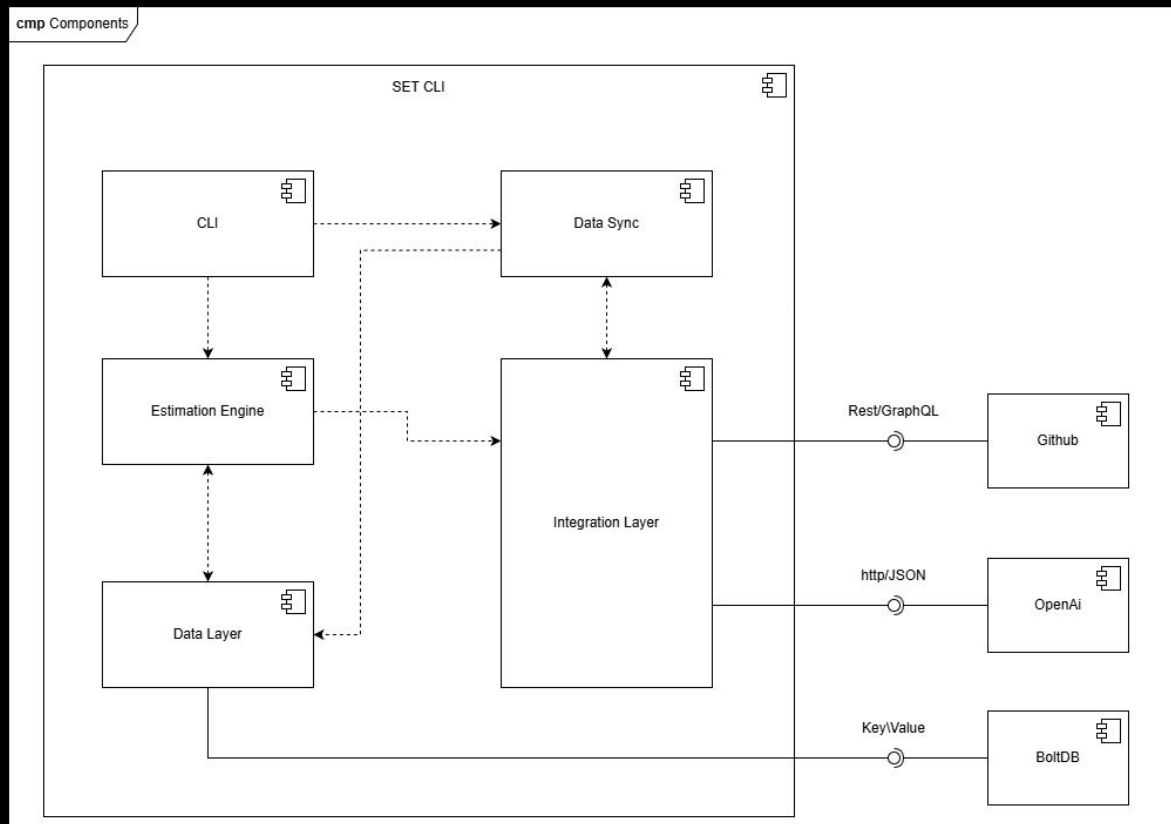
# Arquitetura

O diagrama de contêineres da decompõe o sistema SET CLI em seus principais componentes executáveis, mostrando como a CLI Interface, Estimation Engine, Similarity Engine, AI Client, GitHub Client, Storage Layer e Configuration trabalham em conjunto para fornecer funcionalidade de estimativa.



# Principais Componentes

O diagrama de componentes, demonstra a organização modular do sistema em componentes de alto nível, suas interfaces fornecidas e requeridas, e as dependências entre eles



# Interface

Comando  
Principal de  
estimar.

*set estimate "Implement OAuth login with Google"*

Task

Implement OAuth login with Google

Estimation

Time

16.0h

Size

L

Points

8

Conf

70%

Analysis

The task of implementing OAuth login with Google is a development task that involves integration with an external service (Google). This typically involves understanding the external service's API, coding the integration, handling different scenarios (success, failure, cancellation), and testing. The task is more complex than a simple feature but less complex than creating new components from scratch. The average duration for development tasks is 16 hours, which aligns with this estimate.

Assumptions

- The development environment is set up and ready for development
- The developer is familiar with OAuth and Google APIs
- There are no significant architectural changes required to support this feature

Risks

- ⚡ Potential delays if unfamiliar with OAuth or Google APIs
- ⚡ Possible complications or limitations with Google's API
- ⚡ Potential security implications of handling user authentication

Recommendation

Before starting the task, ensure the developer is familiar with OAuth and Google APIs. Consider a spike to investigate Google's API and any potential challenges. Also, ensure to follow best practices for handling user authentication to mitigate security risks.

ai · openai

Next Steps

- Review the assumptions and risks above
- Consider adding custom fields to GitHub Projects

# Interface

Comando  
Principal de  
estimar.

```
set batch --file .\examples\sprint-backlog.json
```

🔥 Batch Estimation

Input File: .\examples\sprint-backlog.json  
Tasks: 5  
Workers: 1  
Model: gpt-4

Processing tasks...

☒ Batch Processing Complete  
Duration: 56s  
Success: 5/5 tasks

Batch Estimation Report

Overall Statistics

Total Hours	96.0	
Average	19.2	
Median	19.2	
Range	8.0	- 40.0
Avg Confidence	73	%

Size Distribution

Size	Count	Total Hours	Avg Hours
M	1	8.0	8.0
L	4	88.0	22.0

Confidence Distribution

High (≥70%)	5 tasks
Medium (50-69%)	0 tasks
Low (<50%)	0 tasks

Category Distribution

Category	Count	Total Hours
backend	3	72.0
security	1	40.0
frontend	3	40.0
critical	1	8.0
test	1	16.0
feature	2	56.0
bug	1	8.0
refactor	1	16.0
performance	1	16.0

Task Results

ID	Title	Hours	Size	Points	Confidence
	Implementar autenticação ...	40.0	L	13	80%
	Corrigir bug no formulário...	8.0	M	5	70%
	Adicionar paginação na li...	16.0	L	13	75%
	Escrever testes unitários ...	16.0	L	8	70%
	Refatorar componente de car...	16.0	L	13	70%

---

# Demo

---



# Post-mortem

01

Experiência  
Positivas



**Escolha da Linguagem Go:** O uso de Go foi acertado devido à facilidade de criar binário único multiplataforma, excelente performance nativa, suporte eficiente a concorrência para processamento paralelo e a familiaridade da equipe com a linguagem, acelerando o desenvolvimento.

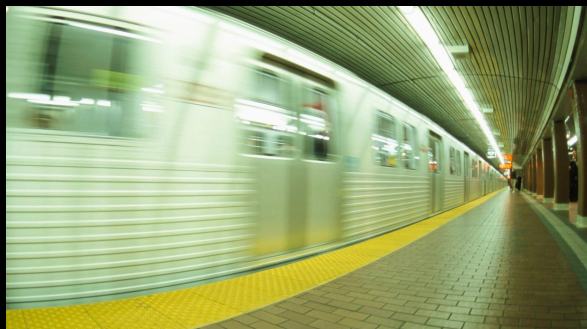
**Integração com IA (OpenAI):** A API OpenAI superou expectativas na geração de estimativas baseadas em linguagem natural, agregando valor ao produto pela capacidade da IA de compreender contexto e gerar raciocínios explicativos.

**Arquitetura Clean Architecture:** A adoção dos princípios de Clean Architecture facilitou a manutenção, testabilidade e evolução do código, com uma separação clara entre as camadas de domínio, casos de uso e infraestrutura.

# Post-mortem

02

Desafios e  
Soluções



**Tratamento de Dados Inconsistentes do GitHub:** Validação rigorosa de dados na camada de integração. Implementação de normalização e sanitização de campos antes de persistir. Logs detalhados para debug de dados problemáticos.

**Balanceamento entre Custo de API e Precisão:** Sistema de cache inteligente para evitar chamadas repetitivas. Batch processing para reduzir overhead por requisição.

**Definição de Thresholds para Similaridade:**

Implementação de threshold adaptativo que se ajusta baseado na quantidade e qualidade de dados históricos disponíveis. Múltiplos níveis de confiança informados ao usuário.

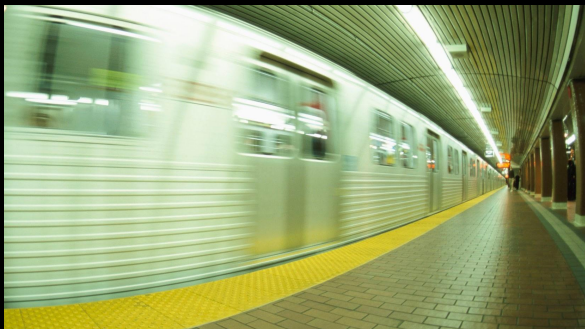
**Sincronização com GitHub Rate Limits:**

Implementação de rate limiting awareness com backoff exponencial. Cache agressivo de dados já sincronizados. Sync incremental baseado em timestamps.

# Post-mortem

03

Lições  
Aprendidas



## **Importância de dataset de qualidade:**

A qualidade das estimativas está diretamente relacionada à qualidade e quantidade dos dados históricos. Projetos com menos de 50 issues têm estimativas significativamente menos confiáveis. Recomenda-se aguardar o acúmulo de histórico antes de confiar cegamente nas estimativas.

**Validação de inputs é essencial:** Diversos bugs em produção foram evitados graças a validações rigorosas de entrada. Especialmente importante para dados vindos de APIs externas que podem ter formatos inconsistentes.

## **CLI UX Requer Feedback Claro:**

Para ferramentas CLI, feedback visual claro é crucial. Implementação de progress bars, colorização de output e mensagens de erro descritivas melhorou significativamente a interpretação dos dados.

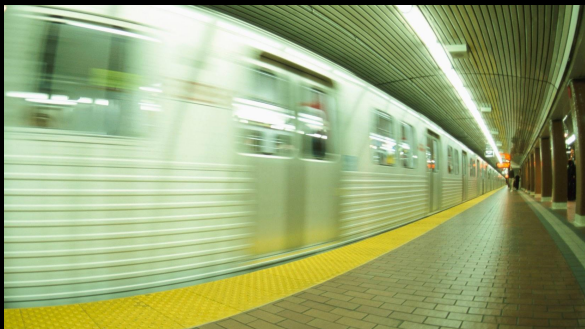
## **Trade-offs de Performance vs Precisão:**

Nem sempre vale a pena buscar a precisão máxima se isso compromete muito a performance. Encontrar o ponto de equilíbrio requer experimentação e métricas claras.

# Post-mortem

04

Melhorias  
Futuras



## **Suporte a Múltiplos Provedores de IA:**

Implementar interfaces agnósticas para permitir uso de Anthropic Claude, Google Gemini, ou outros modelos além de OpenAI. Isso reduziria vendor lock-in e permitiria comparação de resultados.

## **Interface Web Complementar:**

Embora a CLI seja o foco, uma interface web leve para visualização de métricas e dashboards agregaria valor para Product Owners e stakeholders não-técnicos.

## **Integração com Jira/Azure DevOps:**

Expandir além do GitHub para suportar outras ferramentas de gestão de projetos amplamente usadas.

## **Machine Learning Local:**

Implementar um modelo de ML local (treinado nos dados históricos do time) como fallback quando a API está indisponível ou para reduzir custos.

## **Suporte a Múltiplos Repositórios:**

Permitir análise consolidada de múltiplos repositórios para empresas que trabalham com micro serviços.

## **Notificações e Alertas:**

Sistema de alertas quando accuracy começa a degradar ou quando padrões suspeitos são detectados.

---

# Obrigado!