Documentação de Projeto

para o sistema

SET (Software Estimation Tool)

Versão 0.1

Projeto de sistema elaborado pelo aluno **Inácio Moraes da Silva** e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo e orientação acadêmica do professor **Cleiton Silva Tavares**.

1 de agosto 2025

Tabela de Conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Inacio Moraes	10/09/2025	Documentação inicial	0.1

1. Introdução

Este documento agrega: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o sistema **SET** (Software Estimation Tool). A referência principal para a descrição geral do problema, domínio e requisitos do sistema é o Documento de Visão que descreve a visão de domínio do sistema.

O SET é uma ferramenta **CLI** (Command Line Interface) inteligente desenvolvida em **Go** que utiliza Inteligência Artificial para revolucionar o processo de estimativa de esforço em projetos de software. A escolha da linguagem Go proporciona alta performance, facilidade de distribuição multiplataforma através de binários únicos, e excelente suporte para integração com APIs REST.

1.1 Público-Alvo e Natureza CLI

Importante: O SET é uma ferramenta de linha de comando (CLI) projetada especificamente para profissionais com background técnico em desenvolvimento de software. O público-alvo inclui:

- Desenvolvedores que trabalham diariamente com terminal e ferramentas CLI
- Scrum Masters com formação técnica e experiência em desenvolvimento
- Product Owners com background como desenvolvedores

Todos os usuários do sistema possuem familiaridade com:

- Terminal/console (Linux, macOS, Windows PowerShell)
- Git e versionamento de código
- Ferramentas CLI comuns (npm, docker, kubectl, etc.)
- Conceitos de desenvolvimento de software

Esta escolha deliberada por CLI permite integração natural no workflow de desenvolvedores, automação via scripts, e uso em pipelines CI/CD, mantendo a ferramenta enxuta e eficiente.

2. Modelos de Usuário e Requisitos

2.1 Descrição de Atores

Scrum Master: Facilitador de metodologias ágeis com formação técnica em desenvolvimento de software, responsável por utilizar a ferramenta para obter estimativas mais precisas durante o planejamento de sprints e auxiliar na tomada de decisões sobre a capacidade da equipe. Confortável com terminal e ferramentas CLI.

Desenvolvedor: Membro da equipe de desenvolvimento que utilizará a ferramenta diariamente para validar estimativas próprias e do time, comparando com dados históricos e obtendo sugestões baseadas em IA. Usuário avançado de terminal e ferramentas de linha de comando.

Product Owner: Responsável pelo backlog do produto com experiência prévia como desenvolvedor, que usará as estimativas geradas para priorização de features e planejamento de releases. Mantém familiaridade com ferramentas técnicas e CLI para entender melhor a complexidade das tarefas.

2.2 Modelos de Usuários (Personas)

Esta seção apresenta os modelos de usuários desenvolvidos por meio da implementação de personas que representam os principais usuários da ferramenta CLI SET. As personas foram criadas baseando-se nas necessidades identificadas no Documento de Visão e nos padrões observados em equipes ágeis de desenvolvimento.

Perfil Técnico das Personas: Todas as personas possuem formação técnica e experiência com desenvolvimento de software, refletindo a natureza CLI da ferramenta. O SET é projetado para profissionais que trabalham diariamente com terminal, versionamento de código (Git), e outras ferramentas de linha de comando. A diferença entre as personas está nos seus papéis atuais dentro da equipe ágil, não no background técnico.

Mapeamento de Necessidades vs. Personas

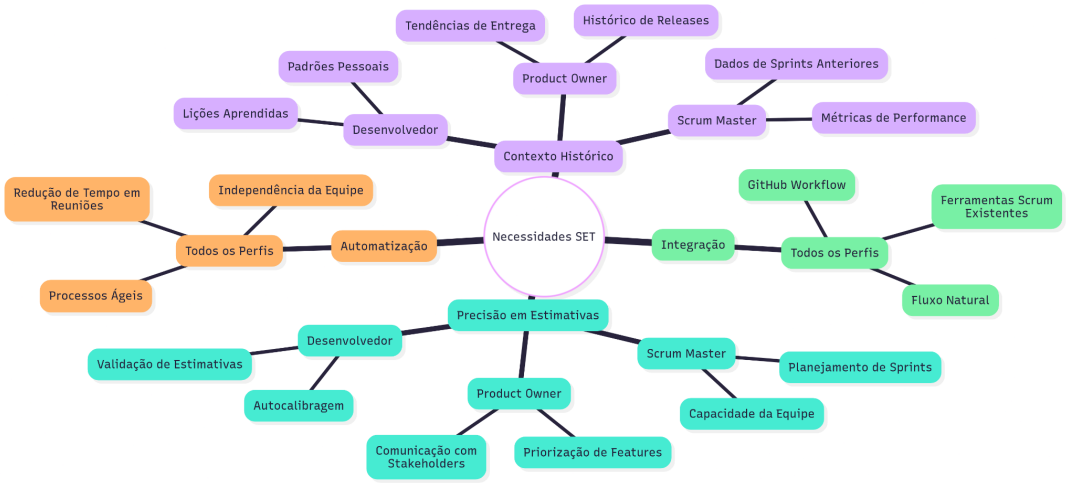

Figura 2.1 - Personas e suas necessidades

Nota: O diagrama acima mostra o mapeamento entre as principais necessidades do sistema SET e as personas envolvidas.

Persona 1: Scrum Master

Aspecto	Detalhes
Nome	Carlos Mendes
Idade e Perfil	35 anos, Scrum Master com 8 anos de experiência em metodologias ágeis
Descrição	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 transparência e busca constantemente formas de otimizar o processo de desenvolvimento.
Contexto Técnico	• Utiliza Jira, Azure DevOps e GitHub diariamente • Familiarizado com CLI tools (git, docker, kubectl) • Experiência com ferramentas de métricas (SonarQube, New Relic) • Confortável com terminal Linux/Mac
Dores Principais	• **Planning Poker Subjetivo**: "As estimativas do planning poker variam muito entre os desenvolvedores e nem sempre refletem a realidade" • **Falta de Dados Históricos**: "Não temos uma base sólida de dados para justificar nossas estimativas para o PO" • **Tempo Gasto em Reuniões**: "Gastamos 2-3 horas por sprint só em estimativas e nem sempre acertamos" • **Inconsistência Entre Times**: "Cada time estima de forma diferente, dificultando comparações"
Objetivos	• Reduzir tempo de planning meetings em 50% • Aumentar precisão das estimativas para >80% • Ter dados concretos para apresentar ao management • Melhorar previsibilidade de entrega dos times • Identificar padrões de super/subestimação

Cenários de Uso	**Preparação de Sprint Planning:**
	$ set analyze --team backend --sprint-history 6
	$ set estimate --batch --file backlog.json
	Revisão de Sprint:
	$ set report --type accuracy --team frontend --period last-sprint
	$ set compare --estimated vs-actual --export dashboard.csv

Métricas	• Redução de 50% no tempo de planning

de Sucesso	• Acurácia > 80% nas estimativas • Velocity mais previsível • Menos re-estimativas durante o sprint

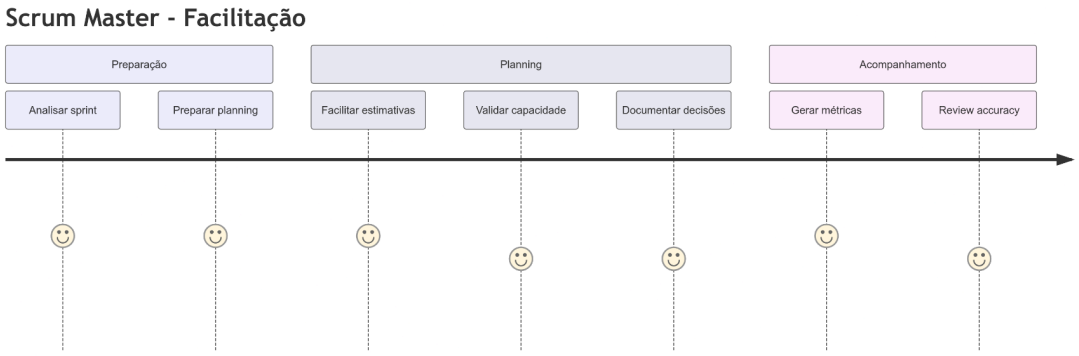

Figura 2.2 - Jornada do Scrum Master

Persona 2: Desenvolvedor Sênior

Aspecto	**Detalhes**
Nome	Ana Rodrigues
Idade e Perfil	29 anos, Desenvolvedor Full Stack Sênior com 6 anos de experiência
Descrição	Ana é desenvolvedora sênior em uma consultoria de software que atende múltiplos clientes. Especialista em React, Node.js e Go. Lidera tecnicamente 2 desenvolvedores juniors e frequentemente é consultada para estimativas de tarefas complexas. Preocupa-se com a qualidade técnica e a precisão das entregas.
Contexto Técnico	• Expert em Git, GitHub Actions, Docker • Usuária avançada de terminal (Zsh, Vim) • Experiência com ferramentas CLI (make, npm, go build) • Trabalha com múltiplos repositórios simultaneamente
Dores Principais	• **Pressão por Estimativas**: "Sempre me pedem para estimar tarefas complexas e a responsabilidade é grande" • **Esquecimento de Detalhes**: "Às vezes esqueço de considerar testes, refactoring ou edge cases" • **Comparação Difícil**: "É difícil lembrar quanto tempo levou uma tarefa similar há 3 meses" • **Estimativas de Juniores**: "Os desenvolvedores juniors sempre subestimam

	por falta de experiência"
Objetivos	• Ter base de dados para fundamentar estimativas • Ajudar desenvolvedores juniors com referências • Melhorar a própria auto calibragem • Reduzir ansiedade nas estimativas • Documentar padrões de desenvolvimento

Cenários de Uso	**Estimativa Individual:**
	$ set estimate --task "Implementar auth JWT" --similar-analysis
	$ set history --my-tasks --pattern authentication
	Mentoring:
	$ set estimate --developer junior --confidence-boost
	$ set recommend --task-type crud --experience-level junior

Métricas de Sucesso	• Menor variação entre estimado vs real • Melhoria na mentoria de juniores • Redução de stress em estimativas

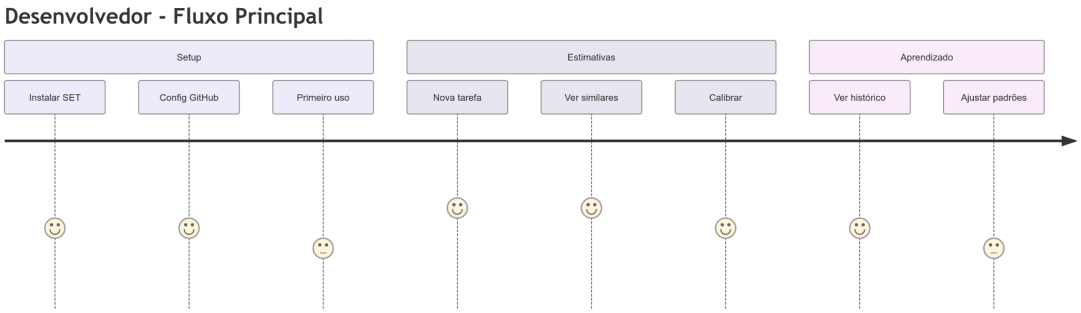

Figura 2.3 - Jornada do Desenvolvedor

Persona 2: Product Owner

Aspecto	**Detalhes**
Nome	Roberto Santos
Idade e Perfil	42 anos, Product Owner com background em negócios e 4 anos em produtos digitais
Descrição	Roberto é PO de um produto SaaS B2B com 50k+ usuários. Vem de área comercial mas aprendeu metodologias ágeis. Precisa constantemente priorizar features e comunicar prazos para C-level. Foco em ROI e time-to-market. Não é técnico mas entende o desenvolvimento.

Contexto Técnico	• Usuário básico de terminal (comandos simples) • Confortável com ferramentas SaaS (Notion, Miro, Figma) • Acessa GitHub ocasionalmente • Prefere interfaces visuais mas aceita CLI com boa UX
Dores Principais	• **Pressão de Prazos**: "O CEO sempre quer saber quando a feature vai estar pronta" • **Falta de Visibilidade**: "Não sei se as estimativas são realistas até ser tarde demais" • **Comunicação com Stakeholders**: "É difícil explicar por que uma 'simples' feature leva 2 sprints" • **Priorização Sem Dados**: "Preciso escolher entre features sem saber o real esforço de cada uma"
Objetivos	• Ter previsões confiáveis para planning de release • Dados para justificar priorizações • Comunicação clara com stakeholders • Melhor ROI nas decisões de produto • Reduzir riscos de atraso em releases

Cenários de Uso	**Planejamento de Release:**
	$ set estimate --epic user-management --breakdown
	$ set report --roadmap --export stakeholder-report.pdf
	Acompanhamento:
	$ set progress --release 2.4 --timeline
	$ set risk-analysis --features high-priority

Métricas de Sucesso	• Releases entregues no prazo (+90%) • Melhor comunicação com stakeholders • Decisões de priorização data-driven • Redução de escopo creep

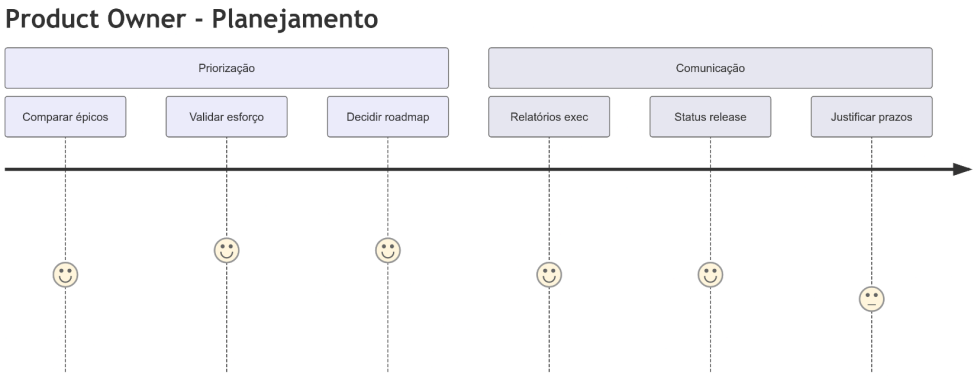

Figura 2.4 - Jornada do Product Owner

Matriz de Funcionalidades vs. Personas

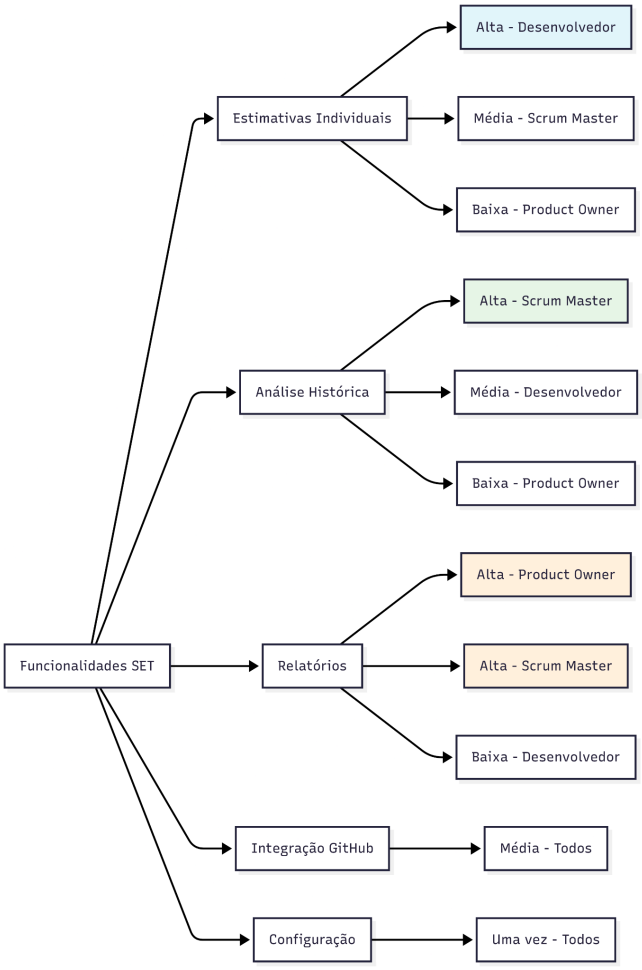

Figura 2.5 - Funcionalidade vs Personas

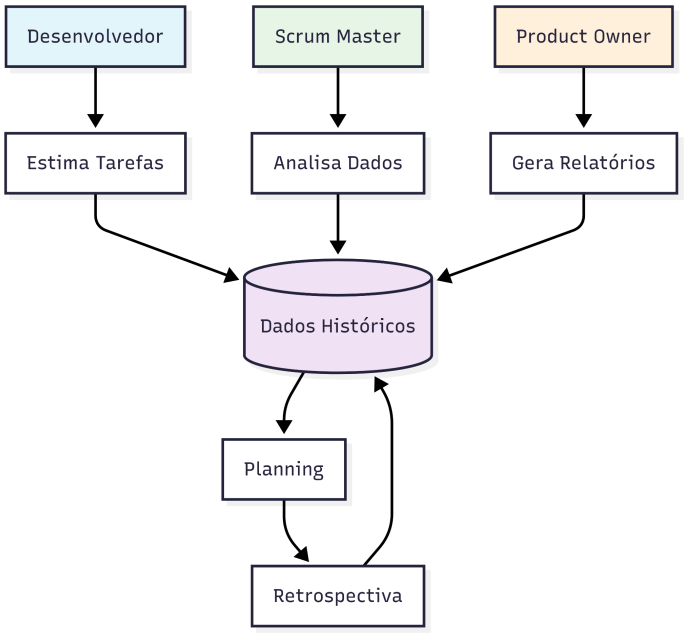

Figura 2.6 - Fluxo entre Personas

Padrões de Uso Identificados

Padrão 1: Desenvolvedor Solo

- Usa *set estimate* para tarefas individuais
- Foca em *--similar-analysis* e *--confidence*
- Histórico pessoal é prioritário

Padrão 2: Scrum Master Facilitador

- Usa *set analyze* para preparar planning
- Focado em métricas de time: *--team, --sprint-history*
- Exporta dados para apresentações

Padrão 3: Product Owner Estratégico

- Usa *set report* para comunicação
- Interessado em timeline e roadmap
- Prefere outputs visuais: *--chart, --export pdf*

2.3 Modelo de Casos de Uso e Histórias de Usuários

Esta seção apresenta o diagrama de casos de uso e as histórias de usuários que descrevem as principais interações dos atores com o sistema SET CLI. O diagrama ilustra as funcionalidades

essenciais e suas relações, enquanto as histórias de usuários detalham os requisitos funcionais na perspectiva de cada tipo de usuário.

2.3.1 Diagrama de Casos de Uso

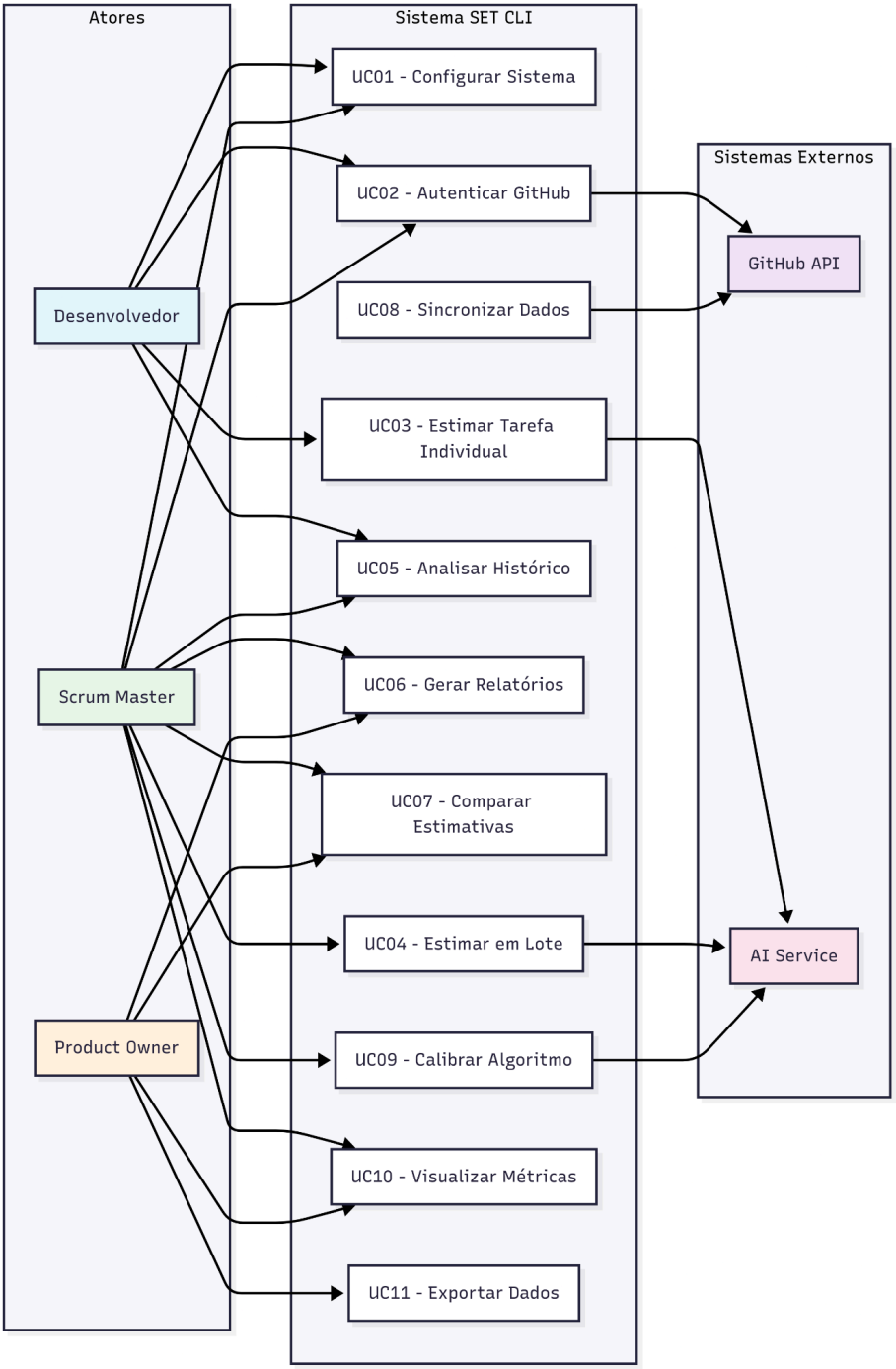

Figura 2.7 - Casos de Uso

2.3.2 Histórias de Usuários

As histórias de usuário apresentadas a seguir estão organizadas por ator e relacionadas aos casos de uso identificados. Para organização, utiliza-se o formato US#ID, onde US refere-se a User Story.

Histórias do Desenvolvedor:

US01 [UC01, UC02] - Como **desenvolvedor**, eu quero **configurar a CLI SET com meu token GitHub** para que o sistema possa acessar dados dos meus repositórios e gerar estimativas personalizadas.

US02 [UC03] - Como **desenvolvedor**, eu quero **estimar uma tarefa específica via CLI** para obter uma estimativa baseada em dados históricos e IA, economizando tempo em planning.

US03 [UC03] - Como **desenvolvedor**, eu quero **ver tarefas similares encontradas** durante a estimativa para validar se a sugestão da IA faz sentido com minha experiência.

US04 [UC05] - Como **desenvolvedor**, eu quero **analisar meu histórico pessoal de estimativas** para identificar padrões e melhorar minha precisão ao estimar.

US05 [UC03] - Como **desenvolvedor**, eu quero **especificar o nível de confiança desejado** na estimativa para adaptar a ferramenta ao meu perfil de risco.

US06 [UC05] - Como **desenvolvedor**, eu quero **filtrar tarefas por tipo ou tecnologia** para encontrar referências mais precisas para estimativa.

Histórias do Scrum Master:

US07 [UC04] - Como **Scrum Master**, eu quero **estimar um lote de tarefas do backlog** para preparar o planning meeting com dados concretos e acelerar as discussões.

US08 [UC05, UC10] - Como **Scrum Master**, eu quero **analisar a performance de estimativas do time** para identificar onde a equipe tem mais dificuldades e focar em melhorias.

US09 [UC06] - Como **Scrum Master**, eu quero **gerar relatórios de accuracy por sprint** para demonstrar a evolução do time e justificar investimentos em processo.

US10 [UC07] - Como **Scrum Master**, eu quero **comparar estimativas entre desenvolvedores** para identificar discrepâncias e promover calibração da equipe.

US11 [UC09] - Como **Scrum Master**, eu quero **ajustar os parâmetros do algoritmo** com base no feedback da equipe para melhorar a precisão das estimativas futuras.

US12 [UC06] - Como **Scrum Master**, eu quero **visualizar tendências de velocity** baseadas em estimativas vs execução para melhorar o planejamento de capacity.

US13 [UC10] - Como **Scrum Master**, eu quero **identificar tipos de tarefas com maior variação** para focar treinamentos específicos da equipe.

Histórias do Product Owner:

US14 [UC06, UC11] - Como **Product Owner**, eu quero **gerar relatórios executivos de estimativas** para comunicar timelines realistas aos stakeholders e management.

US15 [UC07] - Como **Product Owner**, eu quero **comparar o esforço estimado entre diferentes épicos** para priorizar features com melhor ROI.

US16 [UC10] - Como **Product Owner**, eu quero **visualizar o progresso de releases** baseado em estimativas para antecipar possíveis atrasos.

US17 [UC11] - Como **Product Owner**, eu quero **exportar dados para ferramentas de roadmap** para integrar as estimativas ao planejamento estratégico do produto.

US18 [UC06] - Como **Product Owner**, eu quero **ver previsões de entrega** baseadas no histórico da equipe para comunicar datas mais assertivas aos clientes.

2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Esta seção apresenta os Diagramas de Sequência do Sistema (DSS) e os Contratos de Operações que descrevem as principais interações entre os usuários e o sistema SET CLI.

DSS 01: Configurar Sistema

A Figura 2.8 representa o primeiro contato do usuário com o sistema SET CLI, onde é necessário realizar a configuração inicial para que a ferramenta possa funcionar adequadamente. Este fluxo é fundamental pois estabelece a conexão com o GitHub API e define os parâmetros básicos de funcionamento.

O processo inicia quando qualquer usuário (Desenvolvedor, Scrum Master ou Product Owner) executa o comando de configuração inicial. O sistema solicita o token de acesso do GitHub, que é essencial para acessar os repositórios e coletar dados históricos. Após receber o token, o sistema realiza uma validação junto à API do GitHub para garantir que as credenciais são válidas e possuem as permissões necessárias.

Em caso de sucesso na validação, o sistema prossegue solicitando a definição do repositório padrão, que será usado como base para análises quando não especificado outro repositório. O sistema verifica se o usuário tem acesso ao repositório informado, e ao confirmar, finaliza a configuração inicial. Se houver falha na validação do token, o sistema retorna uma mensagem de erro e solicita um novo token válido.

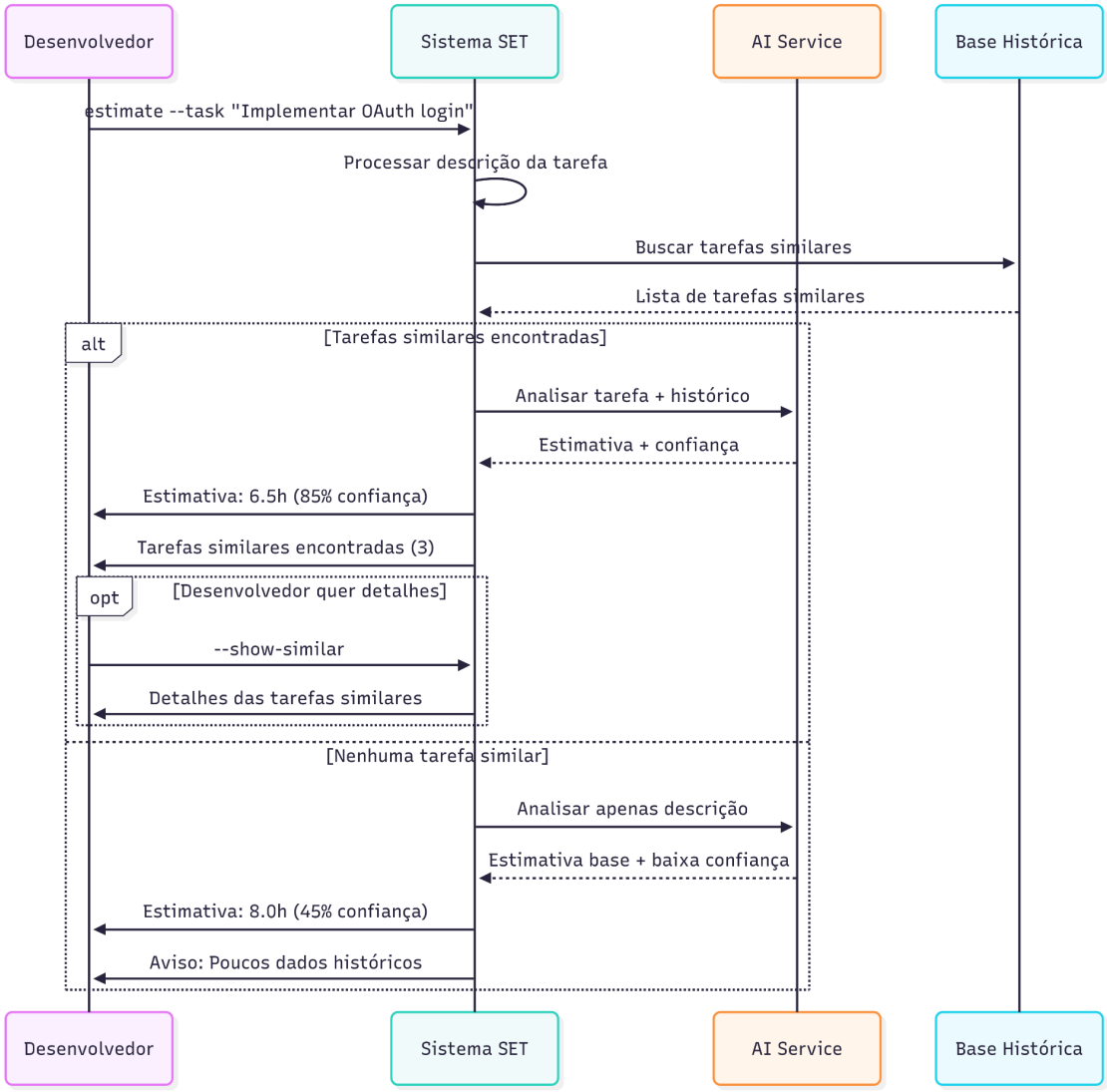

Figura 2.8 - Diagrama de Sequência - Configurar Sistema

Contrato	Configurar Sistema
Operação	configure(githubToken: string, defaultRepo: string)
Referências cruzadas	UC01, US01
Pré-condições	• Sistema não configurado anteriormente • Token GitHub válido • Usuário tem acesso ao repositório
Pós-condições	• Sistema configurado com credenciais • Token GitHub armazenado de forma segura • Repositório padrão definido • Sistema pronto para uso

DSS 02: Estimar Tarefa Individual

A figura 2.9 ilustra o fluxo central da aplicação SET CLI, representando um desenvolvedor que obtém estimativas para tarefas individuais. É o caso de uso mais frequente e demonstra a integração entre análise histórica e inteligência artificial para gerar estimativas precisas.

O desenvolvedor inicia o processo fornecendo uma descrição da tarefa que deseja estimar. O sistema processa esta descrição, extraindo palavras-chave e contexto relevante, e então consulta a base histórica em busca de tarefas similares já executadas. Esta busca utiliza algoritmos de similaridade textual e análise semântica para identificar tarefas com características semelhantes.

Quando tarefas similares são encontradas, o sistema envia tanto a descrição da nova tarefa quanto os dados históricos relevantes para o serviço de IA, que utiliza estas informações para gerar uma estimativa mais precisa e um nível de confiança associado. O resultado é apresentado ao desenvolvedor junto com as tarefas similares encontradas, permitindo validação humana da sugestão.

Nos casos onde não há histórico suficiente, o sistema ainda consegue gerar uma estimativa baseada apenas na análise da descrição, mas com um nível de confiança significativamente menor, alertando o usuário sobre a limitação dos dados disponíveis.

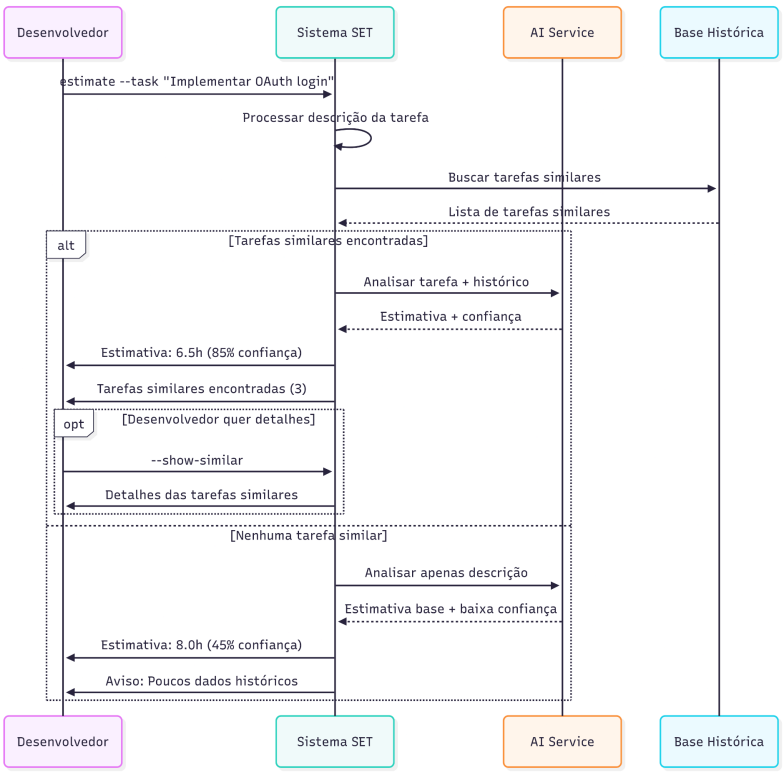

Figura 2.9 - Diagrama de Sequência - Estimar Tarefa Individual

Contrato	Estimar Tarefa Individual
Operação	estimateTask(taskDescription: string, options: EstimationOptions)
Referências cruzadas	UC03, US02, US03
Pré-condições	• Sistema configurado • Usuário autenticado • Descrição da tarefa fornecida
Pós-condições	• Estimativa gerada em horas • Nível de confiança calculado • Tarefas similares identificadas (se existirem) • Resultado exibido para o usuário

DSS 03: Estimar em Lote (Scrum Master)

A figura 2.10 representa uma funcionalidade avançada destinada principalmente aos Scrum Masters, que precisam processar múltiplas tarefas simultaneamente durante a preparação de planning meetings. O processo de estimativa em lote otimiza significativamente o tempo necessário para estimar um backlog completo.

O fluxo inicia quando o Scrum Master fornece um arquivo JSON contendo a lista de tarefas a serem estimadas. O sistema valida e lê este arquivo, extraindo cada tarefa individual. Em seguida, executa um loop de processamento onde cada tarefa é submetida ao mesmo algoritmo de estimativa individual, mas de forma otimizada para processamento em massa.

Durante o processamento, o sistema mantém estatísticas consolidadas e, ao final, compila todos os resultados em um relatório abrangente que inclui não apenas as estimativas individuais, mas também totalizações, médias e outras métricas úteis para o planejamento de sprints. O Scrum Master pode opcionalmente exportar os resultados em formatos adequados para apresentação (CSV, PDF) ou importação em outras ferramentas de gestão de projetos.

Este fluxo é fundamental para acelerar o processo de planning poker e fornecer uma base de dados objetiva para as discussões da equipe, reduzindo o tempo gasto em reuniões e aumentando a precisão do planejamento.

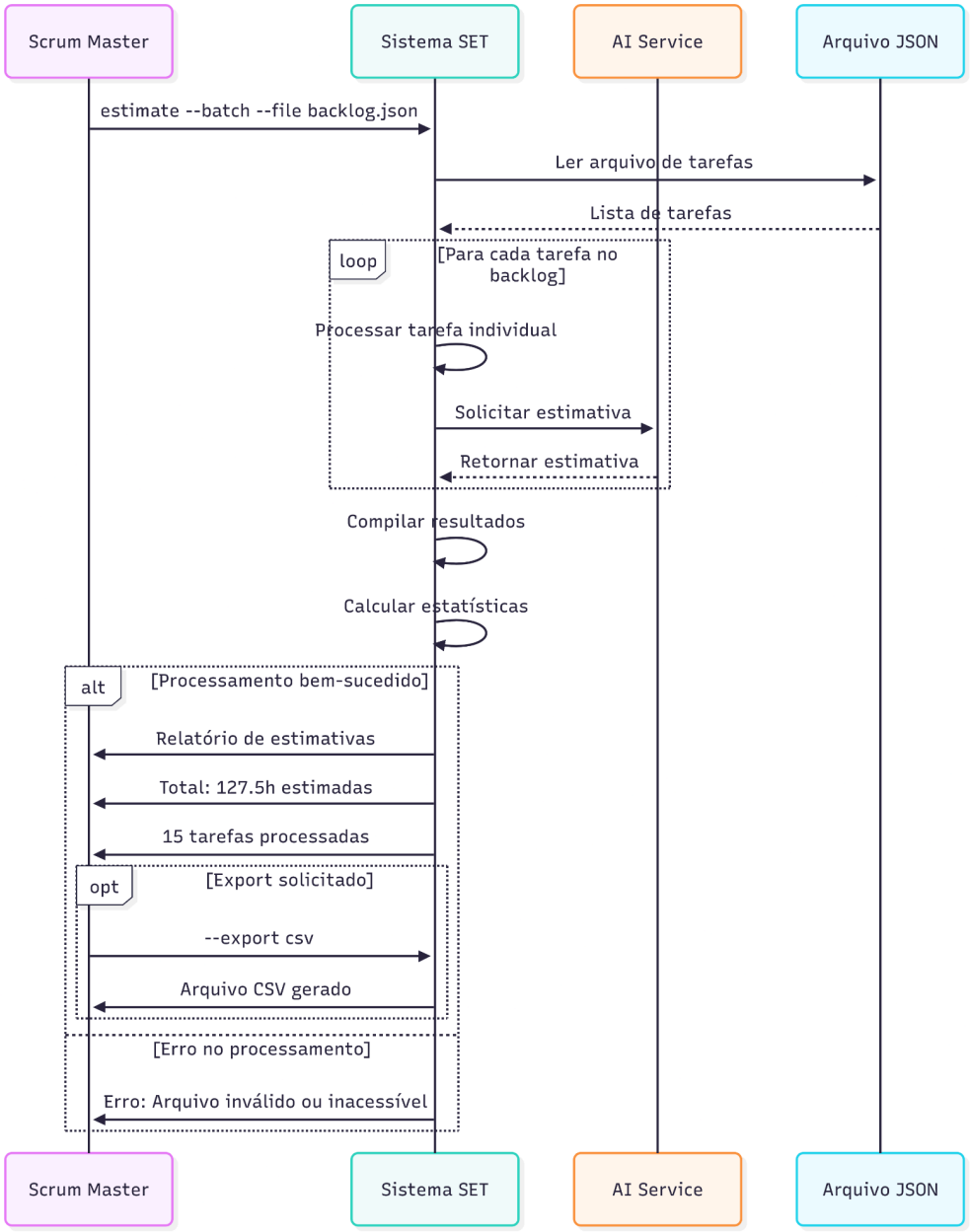

Figura 2.10 - Diagrama de Sequência - Estimar em Lote

Contrato	Estimar em Lote
Operação	estimateBatch(filePath: string, exportFormat?: string)
Referências cruzadas	UC04, US07
Pré-condições	• Sistema configurado • Usuário é Scrum Master • Arquivo JSON válido com lista de tarefas
Pós-condições	• Todas as tarefas estimadas

	• Relatório consolidado gerado • Estatísticas calculadas • Arquivo de exportação criado (opcional)

DSS 04: Analisar Histórico

A figura 2.11 representa uma das funcionalidades mais valiosas para melhoria contínua do processo de estimativas. A análise histórica permite que Scrum Masters identifique padrões, tendências e oportunidades de melhoria na precisão das estimativas da equipe.

O processo inicia com a especificação de filtros para a análise, como período temporal, time específico, tipo de tarefa ou desenvolvedor. O sistema primeiro sincroniza os dados mais recentes do GitHub para garantir que a análise inclua informações atualizadas sobre commits, pull requests e fechamento de issues.

Após a sincronização, o sistema consulta a base histórica aplicando os filtros especificados e realiza uma série de cálculos estatísticos complexos. Isso inclui o cálculo da accuracy (precisão das estimativas), identificação de padrões de super ou subestimação, análise de performance individual dos desenvolvedores e detecção de tendências ao longo do tempo.

Os insights gerados incluem identificação dos tipos de tarefa mais problemáticos, desenvolvedores que consistentemente estimam melhor, evolução da precisão ao longo do tempo e recomendações específicas para melhoria do processo. Esta análise é fundamental para que os Scrum Masters possam tomar decisões data-driven sobre como otimizar o processo de estimativas da equipe.

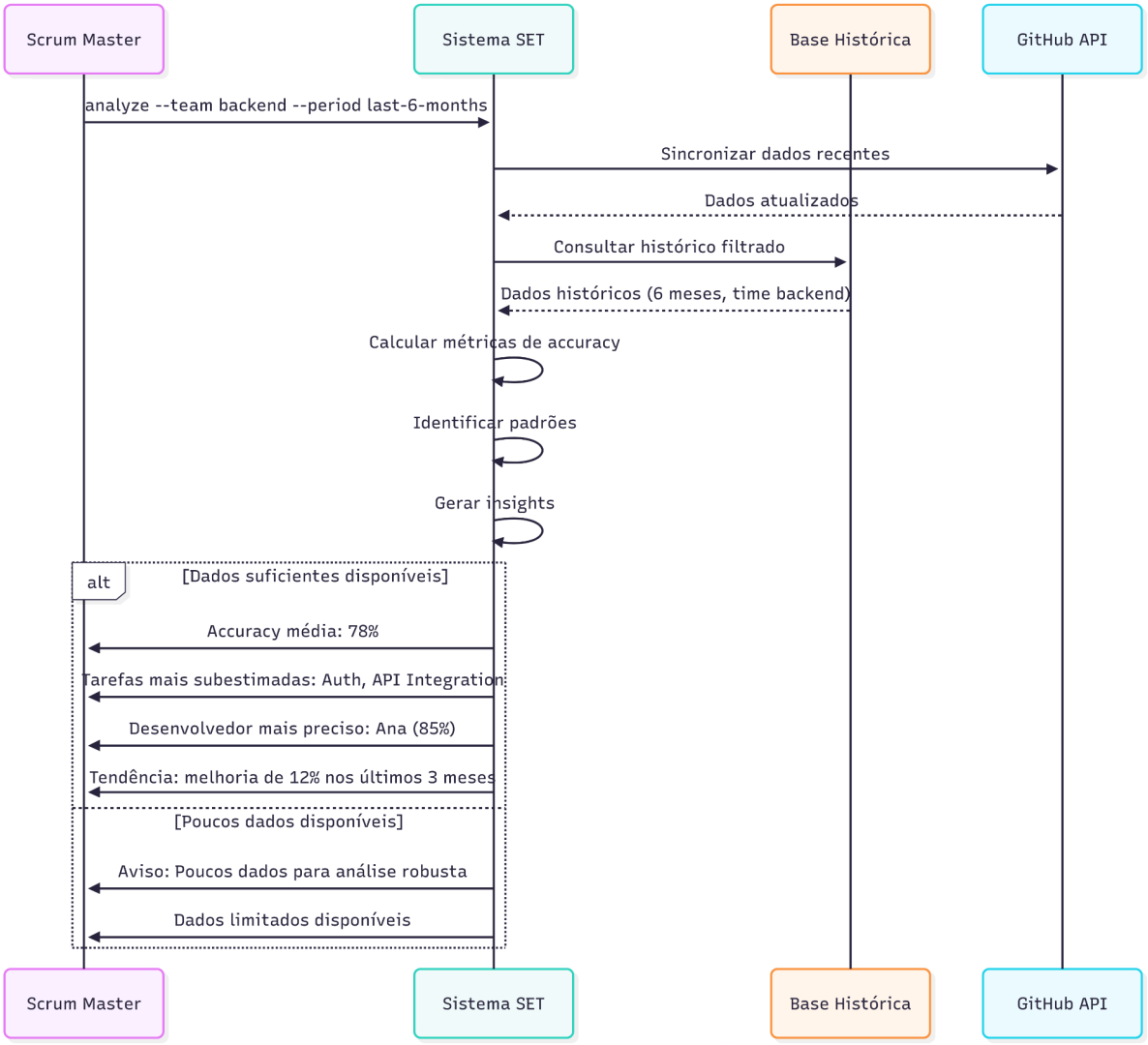

Figura 2.11 - Diagrama de Sequência - Analisar Histórico

Contrato	Analisar Histórico
Operação	analyzeHistory(filters: AnalysisFilters)
Referências cruzadas	UC05, US08, US04
Pré-condições	• Sistema configurado • Dados históricos disponíveis • Filtros válidos aplicados
Pós-condições	• Métricas de accuracy calculadas • Padrões identificado • Insights gerados • Relatório de análise apresentado

DSS 05: Gerar Relatório Executivo

A figura 2.12 ilustra como Product Owners podem gerar relatórios formatados e adequados para apresentação a stakeholders e management. Os relatórios executivos transformam dados técnicos de estimativas em informações de negócio relevantes para tomada de decisão estratégica.

O Product Owner especifica o tipo de relatório desejado e o formato de saída (PDF, PowerPoint, etc.). O sistema então consulta a base histórica coletando dados de múltiplos sprints e projetos, consolidando informações que podem incluir velocity da equipe, precisão das entregas, tendências de produtividade e métricas de qualidade.

O processamento dos dados envolve agregações complexas e cálculos de KPIs relevantes para o negócio, como tempo médio de entrega de features, variação entre estimado e real em releases, e projeções de capacity para futuros desenvolvimentos. O sistema também gera visualizações gráficas que facilitam a compreensão dos dados por audiências não técnicas.

O resultado final é um documento profissional que o Product Owner pode usar em apresentações para diretoria, reuniões de planejamento estratégico ou comunicação com clientes sobre cronogramas de entrega. Este fluxo é essencial para demonstrar o valor das práticas de estimativa e justificar investimentos em melhoria de processos.

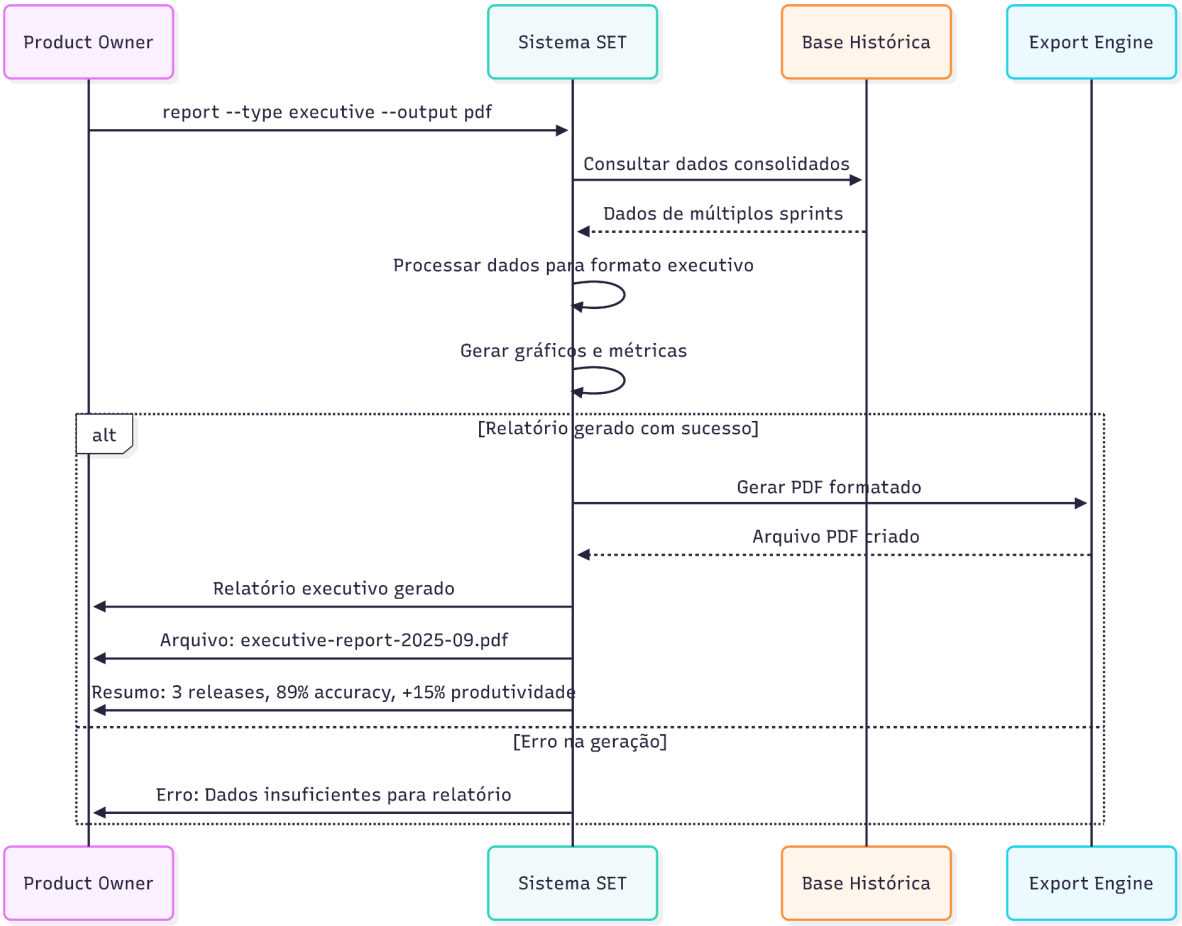

Figura 2.12 - Diagrama de Sequência - Gerar Relatório Executivo

Contrato	Gerar Relatório Executivo
Operação	generateReport(type: ReportType, outputFormat: string)
Referências cruzadas	UC06, US14, US15
Pré-condições	• Sistema configurado • Dados históricos suficientes • Usuário é Product Owner • Formato de saída suportado
Pós-condições	• Relatório formatado gerado • Métricas executivas calculadas • Arquivo de saída criado • Insights de negócio apresentados

DSS 06: Calibrar Algoritmo

A figura 2.13 representa o processo de melhoria contínua do sistema através da incorporação de novos dados de feedback baseados na execução real das tarefas. É uma funcionalidade avançada que

permite que o sistema tenha uma base de dados cada vez mais rica e precisa para futuras estimativas.

O Scrum Master inicia o processo fornecendo dados de feedback de sprints recentes, que incluem as estimativas originais versus o tempo realmente gasto na execução das tarefas. O sistema carrega estes dados de feedback e os valida, verificando consistência e qualidade das informações fornecidas.

O processo de calibração consiste em organizar e integrar estes novos dados à base histórica existente. O sistema pode ajustar pesos e relevância de diferentes fatores nas futuras estimativas, como dar maior importância a dados mais recentes, considerar características específicas do desenvolvedor que executou a tarefa, ou identificar padrões em tipos específicos de tarefas.

Os novos dados são processados e categorizados de forma a enriquecer o contexto disponível para o algoritmo de IA. Isso inclui a criação de novas associações entre descrições de tarefas e tempos de execução, atualização de perfis de desenvolvedores baseados em performance real, e refinamento dos critérios de similaridade entre tarefas.

Após a calibração, o sistema confirma a incorporação dos novos dados e informa sobre a melhoria esperada na precisão das próximas estimativas. Os dados ficam imediatamente disponíveis para serem utilizados em estimativas futuras, tornando o sistema progressivamente mais inteligente e adaptado às características específicas da equipe e projeto.

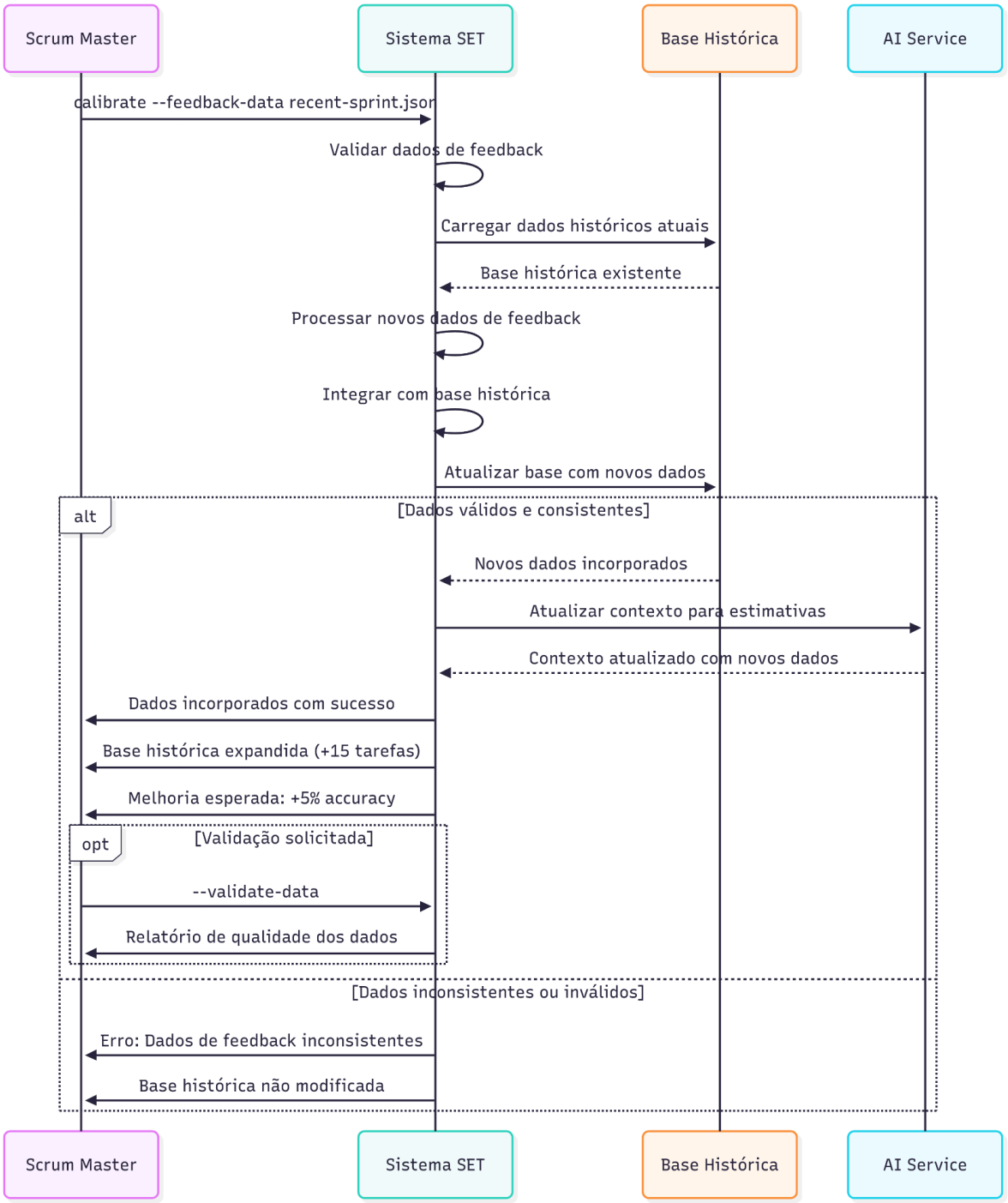

Figura 2.13 - Diagrama de Sequência - Calibrar Algoritmo

Contrato	Calibrar Algoritmo
Operação	calibrateModel(feedbackData: FeedbackData, options: CalibrationOptions)

Referências cruzadas	UC09, US11
Pré-condições	• Sistema configurado • Usuário é Scrum Master • Dados de feedback válidos disponíveis • Base histórica existente
Pós-condições	• Novos dados incorporados à base histórica • Contexto de estimativas enriquecido • Dados disponíveis para futuras estimativas • Log de incorporação registrado

3. Modelos de Projeto

3.1 Diagrama de Classes

Vou montar junto com desenvolvimento

Descrição das Classes Principais

Entidades do Domínio:

- **User**: Representa os usuários do sistema (Desenvolvedor, Scrum Master, Product Owner) com suas configurações e permissões
- **Task**: Modelagem de tarefas de software com metadados para estimativa e análise
- **Estimation**: Resultado de uma estimativa, incluindo confiança, variância e recomendações
- **HistoricalData**: Dados históricos de execução de tarefas para análise e calibração

Serviços de Negócio:

- **EstimationService**: Lógica central para estimativas individuais e em lote
- **AnalysisService**: Análise de dados históricos e identificação de padrões
- **ReportService**: Geração de relatórios e dashboards para diferentes usuários
- **ConfigurationService**: Gerenciamento de configurações e preferências

Interfaces de Integração:

- **Repository**: Interface para persistência de dados (implementada por BoltRepository)
- **AIClient**: Interface para serviços de IA (implementada por OpenAIClient)
- **GitHubClient**: Cliente para integração com a API do GitHub

Comandos CLI:

- **CLIApplication**: Aplicação principal que coordena todos os comandos
- **EstimateCommand, AnalyzeCommand, ReportCommand**: Comandos específicos da CLI

3.2 Diagramas de Sequência

Diagramas de sequência para realização de casos de uso.

3.3 Diagramas de Comunicação

Diagramas de comunicação para realização de casos de uso.

3.4 Arquitetura

Esta seção apresenta a modelagem da arquitetura do sistema SET CLI utilizando o modelo C4 (Context, Containers, Components, Code), que fornece uma abordagem hierárquica para visualizar a arquitetura de software em diferentes níveis de abstração. O modelo C4 foi escolhido por sua clareza na comunicação da arquitetura tanto para audiências técnicas quanto não técnicas, facilitando o entendimento desde a visão de alto nível até os detalhes de implementação.

A arquitetura do sistema SET CLI foi projetada seguindo os princípios de Clean Architecture, garantindo separação de responsabilidades, baixo acoplamento e alta coesão. O sistema utiliza Go como linguagem principal, aproveitando suas características de performance, concorrência e facilidade de distribuição multiplataforma.

Nível 1: Diagrama de Contexto

O diagrama de contexto apresenta a visão mais ampla do sistema, mostrando como o SET CLI se relaciona com seus usuários e sistemas externos.

Figura 3.xx - C4 *TODO*

Nível 2: Diagrama de Containers

O diagrama de componentes detalha a estrutura interna do Core Engine, o container mais crítico do sistema.

Figura 3.xx - C4 *TODO*

Arquitetura de Software Detalhada

Padrões Arquiteturais Adotados

Clean Architecture: O sistema segue os princípios da Clean Architecture, organizando o código em camadas concêntricas onde as dependências apontam sempre para dentro. As regras de negócio estão isoladas das implementações específicas de infraestrutura.

Repository Pattern: Utilizado para abstrair o acesso a dados, permitindo diferentes implementações de persistência (BoltDB para dados locais, potencial extensão para bancos remotos).

Dependency Injection: Implementado através de interfaces Go, facilitando testes unitários e flexibilidade na troca de implementações.

Strategy Pattern: Aplicado nos algoritmos de estimativa, permitindo diferentes abordagens de cálculo conforme o contexto.

Tecnologias e Frameworks

Linguagem Principal: Go 1.21+

- **Vantagens**: Performance superior, concorrência nativa com goroutines, facilidade de distribuição (binário único), excelente suporte para CLI
- **Frameworks**: Cobra (CLI), Viper (configuração), BoltDB (persistência)

Persistência Local:

- **BoltDB**: Banco de dados embarcado para dados históricos e estimativas
- **JSON/YAML**: Arquivos de configuração e cache estruturado
- **Vantagens**: Zero configuração, performance local, portabilidade

Integração Externa:

- **GitHub API v4**: Para coleta de dados de repositórios via GraphQL/REST
- **OpenAI API**: Para análise de linguagem natural e geração de estimativas
- **Rate Limiting**: Implementado para respeitar limites das APIs externas

Fluxo de Dados e Processamento

Estimativa Individual:

1. CLI recebe comando e parâmetros do usuário
2. Estimation Service processa a descrição da tarefa
3. Similarity Engine busca tarefas similares no histórico local
4. AI Client envia contexto para análise de IA
5. Resultado consolidado é retornado e formatado para exibição

Análise em Lote:

1. CLI carrega arquivo de tarefas (JSON/CSV)
2. Core Engine processa cada tarefa usando goroutines para paralelização
3. Resultados são agregados e estatísticas calculadas
4. Export Engine gera arquivos de saída nos formatos solicitados

Sincronização de Dados:

1. GitHub Client busca dados incrementais do repositório
2. Cache Manager otimiza requisições repetitivas
3. Repository persiste novos dados no BoltDB local
4. Similarity Engine reindexiza dados para buscas futuras

Estratégias de Performance

Concorrência: Utilização de goroutines para processamento paralelo de estimativas em lote e requisições de API.

Caching:

- Cache em memória para dados frequentemente acessados
- Cache de arquivos para resultados de APIs externas
- TTL configurável para balancear freshness e performance

Indexação: Índices otimizados no BoltDB para buscas rápidas de tarefas similares baseadas em keywords e metadados.

Segurança e Configuração

Autenticação: Tokens GitHub armazenados com criptografia AES-256 local.

Configuração: Sistema hierárquico de configuração (defaults < arquivo config < variáveis ambiente < parâmetros CLI).

Validação: Validação rigorosa de entrada para prevenir injection attacks e garantir integridade dos dados.

Escalabilidade e Extensibilidade

Modularidade: Arquitetura permite fácil adição de novos provedores de IA, sistemas de versionamento ou formatos de exportação.

Plugins: Interface preparada para extensões futuras via plugins Go.

Multi-repositório: Suporte nativo para análise de múltiplos repositórios simultaneamente.

Observabilidade

Logging: Sistema estruturado de logs com níveis configuráveis (Debug, Info, Warn, Error).

Métricas: Coleta de métricas de performance internas (tempo de resposta, taxa de acerto do cache, accuracy das estimativas).

3.5 Diagramas de Estados

Vou montar junto com desenvolvimento

3.6 Diagrama de Componentes e Implantação.

Vou montar junto com desenvolvimento

4. Projeto de Interface com Usuário

4.1 Interface CLI Comum a Todos os Atores

set estimate --task "Implementar login" --repo "user/project"
set analyze --project "user/project" --timeframe "last-6-months"
set configure --github-token "token" --repo "user/project"
set report --type estimation --output csv
set help

5. Glossário e Modelos de Dados

***CLI**: Command Line Interface - Interface de linha de comando*
***IA**: Inteligência Artificial*
***API**: Application Programming Interface*
***Scrum**: Framework ágil para desenvolvimento de software*
***GitHub**: Plataforma de hospedagem de código*
***Sprint**: Período fixo de desenvolvimento no Scrum*

6. Casos de Teste

Esta seção apresenta os casos de teste elaborados para validar o sistema SET CLI. Na Seção 6.1, são detalhados os testes de aceitação, projetados para assegurar que as funcionalidades desenvolvidas atendam às necessidades dos usuários conforme especificado no documento de visão. A Seção 6.2 descreve os testes de integração, focando em verificar a interação entre o sistema e os componentes externos, como GitHub API e serviços de IA.
Os testes foram organizados para cobrir as quatro necessidades principais identificadas no documento de visão: precisão em estimativas de esforço, automatização do processo de estimativa, utilização de contexto histórico e integração com ferramentas existentes. Cada caso de teste inclui pré-condições, dados de entrada, ações a serem executadas e resultados esperados, garantindo cobertura completa das funcionalidades críticas.

6.1 Testes de Aceitação

Os testes de aceitação (TA) foram elaborados com base nas necessidades identificadas no documento de visão e nos casos de uso apresentados. As necessidades testadas são:

- **N1**: Precisão em Estimativas de Esforço
- **N2**: Automatização do Processo de Estimativa
- **N3**: Utilização de Contexto Histórico
- **N4**: Integração com Ferramentas Existentes

Para a escrita dos testes, utiliza-se uma tabela com seis campos: Identificador (padrão TAXX), Necessidade, Caso de teste, Pré-condições, Dados de entrada, Ações e Resultado esperado.

6.1.1 Necessidade 1 – Precisão em Estimativas de Esforço

Esta necessidade garante que o sistema forneça estimativas precisas baseadas em dados históricos e análise de IA, reduzindo significativamente os desvios entre estimado e executado.

Identificador	TA01
Necessidade	Precisão em Estimativas de Esforço
Caso de teste	Gerar estimativa com alta confiança para tarefa similar
Pré-condições	• Sistema configurado com GitHub token válido • Base histórica contém pelo menos 5 tarefas similares • Usuário autenticado
Dados de entrada	Descrição da tarefa: "Implementar autenticação OAuth com Google"
Ações	1. Executar comando set estimate --task "Implementar autenticação OAuth com Google" 2. Sistema processa a descrição 3. Sistema busca tarefas similares 4. IA analisa e gera estimativa
Resultado esperado	• Estimativa gerada entre 4-12 horas • Confiança ≥ 80% • Pelo menos 3 tarefas similares encontradas • Tempo de resposta ≤ 30 segundos

Identificador	TA02
Necessidade	Precisão em Estimativas de Esforço

Caso de teste	Gerar estimativa com baixa confiança para tarefa inédita
Pré-condições	• Sistema configurado com GitHub token válido • Base histórica não contém tarefas similares
Dados de entrada	Descrição da tarefa: "Implementar blockchain para rastreamento"
Ações	1. Executar comando com tarefa inédita 2. Sistema não encontra similaridades 3. IA faz estimativa baseada apenas na descrição
Resultado esperado	• Estimativa gerada com confiança ≤ 50% • Aviso sobre limitação de dados históricos • Recomendação para revisar estimativa manualmente

ToDO: Add more cases

6.1.2 Necessidade 2 – Automatização do Processo de Estimativa

Esta necessidade visa reduzir o tempo gasto em reuniões de planning e permitir estimativas independentes da disponibilidade da equipe completa.

Identificador	TA03
Necessidade	Automatização do Processo de Estimativa
Caso de teste	Processar lote de tarefas para planning sprint
Pré-condições	• Sistema configurado • Arquivo JSON com 15 tarefas do backlog
Dados de entrada	Arquivo sprint-backlog.json contendo lista de tarefas
Ações	1. Executar set estimate --batch --file sprint-backlog.json 2. Sistema processa todas as tarefas em paralelo 3. Sistema gera relatório consolidado
Resultado esperado	• 15 estimativas geradas com sucesso • Tempo total de processamento ≤ 2 minutos • Relatório com total de horas estimadas • Arquivo CSV de saída gerado

ToDO: Add more cases

6.1.3 Necessidade 3 – Utilização de Contexto Histórico

Esta necessidade garante que dados valiosos de projetos anteriores sejam aproveitados para melhorar a precisão das estimativas futuras.

Identificador	TA04
Necessidade	Utilização de Contexto Histórico
Caso de teste	Analisar performance histórica da equipe
Pré-condições	• Base histórica com dados de 6 meses

	• Múltiplos desenvolvedores no histórico
Dados de entrada	Filtros: --team backend --period last-6-months
Ações	1. Executar set analyze --team backend --period last-6-months 2. Sistema processa dados históricos 3. Sistema calcula métricas de performance
Resultado esperado	• Accuracy média da equipe exibida • Identificação de padrões de super/subestimação • Ranking de desenvolvedores por precisão • Tendências temporais de melhoria

ToDO: Add more cases

6.1.4 Necessidade 4 – Integração com Ferramentas Existentes

Esta necessidade assegura que o sistema se integre naturalmente com GitHub e ferramentas Scrum sem causar disrupção no fluxo de trabalho existente.

Identificador	TA05
Necessidade	Integração com Ferramentas Existentes
Caso de teste	Sincronizar dados do GitHub automaticamente
Pré-condições	• Sistema configurado com GitHub • Repositório ativo com commits recentes
Dados de entrada	Comando de sincronização manual
Ações	1. Executar set sync --repository user/project --since last-week 2. Sistema acessa GitHub API 3. Sistema coleta issues, commits, PRs
Resultado esperado	• Dados sincronizados com sucesso • Issues fechadas adicionadas à base • Commits analisados para métricas • Rate limits respeitados

ToDO: Add more cases

7. Cronograma e Processo de Implementação

Esta seção apresenta o cronograma detalhado para implementação do sistema SET CLI e descreve o processo que será seguido durante o desenvolvimento. O cronograma foi estruturado considerando as datas marco estabelecidas e organizando as atividades em sprints quinzenais para facilitar o acompanhamento e entregas incrementais.

7.1 Cronograma de Implementação

Sprint	Período	Marco	Atividades Principais	Entregáveis
Sprint 1	16/09 - 29/09	**A2 - Boilerplate**	• **Setup do Projeto e Estrutura Inicial** • Configuração do repositório GitSetup do ambiente Go com módulos • Implementação da estrutura CLI com Cobra	• Projeto Go inicializado • Comandos CLI básicos funcionais • README com instruções de setup
Sprint 2	30/09 - 13/10		• **Sistema de Configuração e GitHub Integration** • Implementação do Configuration Service • Desenvolvimento do GitHub Client • Sistema de autenticação com tokens • Comandos de configuração inicial • Validação de conectividade	• Comando set configure funcional • Integração GitHub API implementada • Validação de tokens funcionando • Testes unitários das integrações • Documentação da configuração
Sprint 3	14/10 - 20/10	**A3 - Core Funcional**	• **Core de Estimativas e Persistência Local** • Implementação do EstimationService • Desenvolvimento do BoltRepository • Sistema de busca de similaridade • Comando básico de estimativa individualEstrutura de dados históricos	• Comando set estimate funcional • Persistência local operacional • Algoritmo de similaridade básico • Testes de integração com BoltDB • Documentação das APIs internas

Sprint 4	21/10 - 03/11		• **Integração com IA e Análise Avançada** • Implementação do AI Client (OpenAI) • Desenvolvimento do Similarity Engine • Integração IA + dados históricos • Sistema de confiança das estimativas • Tratamento de erros e fallbacks	• IA integrada às estimativas • Sistema de confiança funcional • Tratamento robusto de erros • Testes com mocks da API OpenAIMétricas de accuracy implementadas
Sprint 5	04/11 - 10/11	**A4 - Implementação Finalizada**	• **Funcionalidades Avançadas e Otimização** • Implementação do comando batch • Desenvolvimento do AnalysisService • Sistema de relatórios básico • Cache e otimização de performance • Comando de calibração	• Comando set estimate --batch funcional • Análise histórica implementada • Sistema de cache operacional • Relatórios básicos funcionando • Performance otimizada
Sprint 6	11/11 - 24/11		• **Relatórios Executivos e Export** • Implementação do ReportService completo • Sistema de exportação (CSV, PDF)Dashboards e visualizações • Comandos de análise avançada • Testes de integração completos	• Sistema completo de relatórios • Exportação funcionando • Todos os comandos implementados • Suite completa de testes • Cobertura de testes >85%
Sprint 7	18/11 - 24/11	**A5 - Revisão Final**	• **Testes Finais e Documentação** • Execução de todos os casos de teste • Correção de bugs identificados • Documentação técnica completa • Guias de usuário e instalação	• Todos os testes passando • Documentação completa • Guias de instalação e uso • Binários para múltiplas plataformas

				• Revisão de código finalizada
Sprint 8	25/11 - 30/11		• **Finalização e Entrega** • Testes finais de aceitação • Preparação da apresentação • Release final do sistema • Empacotamento para distribuição • Documentação de deployment	• Sistema 100% funcional • Release v1.0.0 publicada • Apresentação preparada • Documentação final entregue • Projeto pronto para uso