# Documentação Completa do Projeto: Sistema de Gerenciamento para a Academia XYZ

**KAYKY MATOS SANTANA, RGM: 33345945**
**LUIZ HENRIQUE SOUZA PRUDENTE, RGM:32920598**
**THIAGO APARECIDO GOMES, RGM:4933768391**
**GUILHERME FALCK DO CARMO, RGM:4934721193**
**LUCAS MATHEUS DE LIRA VASQUEZ, RGM: 33232083**
**NICOLAS GUSTAVO PRADO ARLANDIS MORAES, RGM: 4933434875**

# 1. Escopo do Projeto

O projeto visa desenvolver um **Sistema de Gerenciamento de Academia** para a **Academia XYZ**, uma rede de academias de ginástica que busca otimizar suas operações. O sistema permitirá o gerenciamento de membros, agendamento de aulas, controle de pagamentos e geração de relatórios detalhados de uso e desempenho. O foco será criar uma solução robusta, segura e amigável ao usuário, garantindo alta performance e compatibilidade com dispositivos móveis.

# 2. Objetivos do Projeto

**Objetivo Geral**: Desenvolver um sistema de software que facilite o gerenciamento de operações da Academia XYZ, proporcionando maior eficiência administrativa e uma experiência melhorada para os membros.

**Objetivos Específicos**:

- Permitir o cadastro, atualização e exclusão de membros.
- Implementar funcionalidades de agendamento, cancelamento e reagendamento de aulas.
- Monitorar e registrar pagamentos, oferecendo alertas de inadimplência.
- Fornecer relatórios detalhados sobre a utilização das modalidades e frequência dos membros.
- Assegurar uma interface amigável e de fácil uso, compatível com smartphones.
- Garantir a segurança de dados pessoais e financeiros.

# 3. Requisitos do Sistema

### 3.1 Requisitos Funcionais (RF)

- **RF1**: Cadastro de membros – O sistema deve permitir o cadastro, atualização e exclusão de informações pessoais dos membros (nome, idade, endereço, detalhes de contato).
- **RF2**: Agendamento de aulas – Membros devem ter a capacidade de agendar, cancelar e reagendar aulas.
- **RF3**: Controle de pagamento – O sistema deve registrar todos os pagamentos, acompanhar o status de pagamento e gerar alertas para inadimplências.
- **RF4**: Geração de relatórios – O sistema deve gerar relatórios semanais e mensais sobre a frequência dos membros e utilização das modalidades.

### 3.2 Requisitos Não Funcionais (RNF)

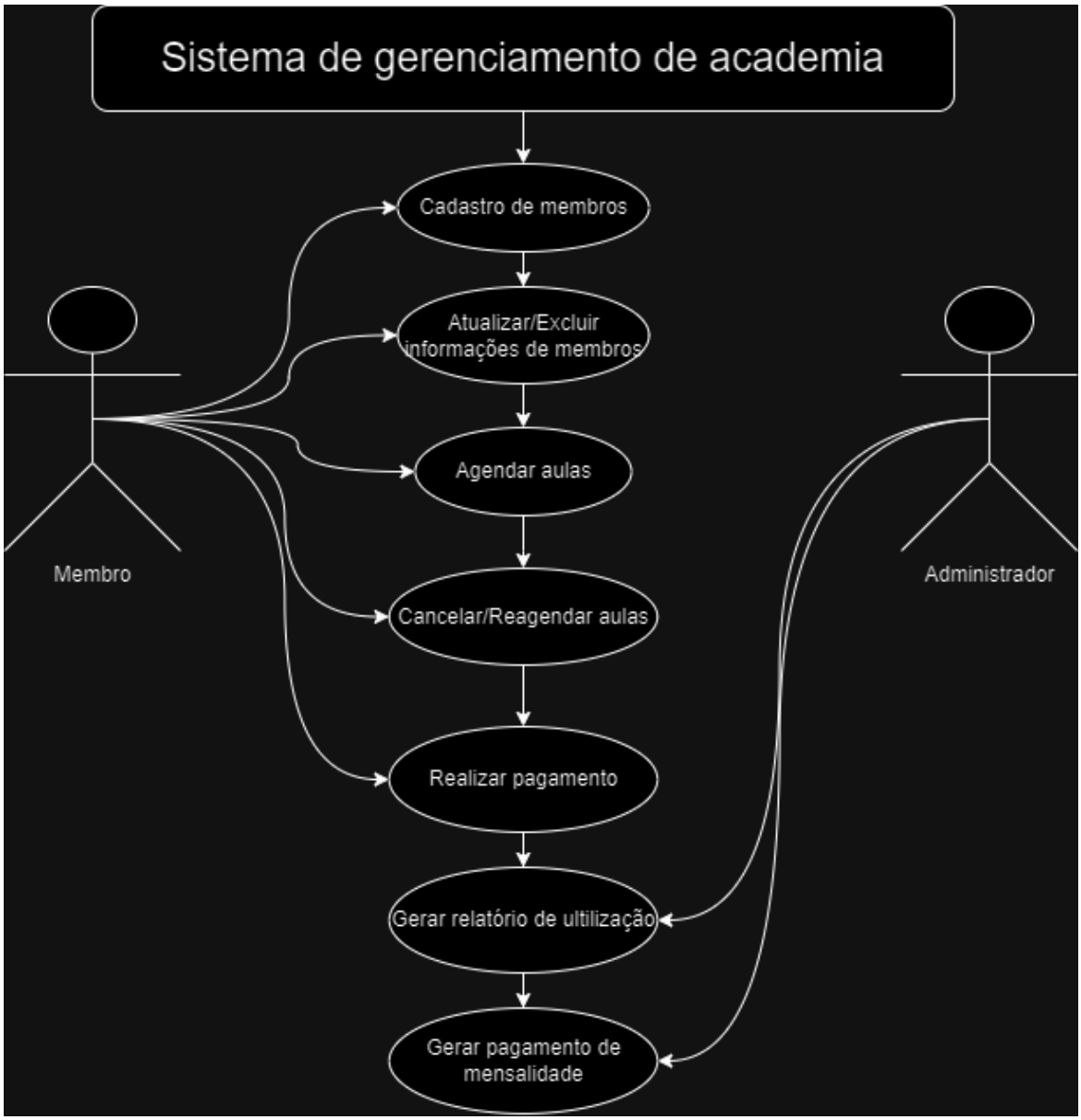
- **RNF1**: Desempenho – O sistema deve suportar até 1000 membros simultaneamente, garantindo que operações como o agendamento sejam processadas em menos de 2 segundos.
- **RNF2**: Segurança – Informações pessoais e financeiras devem ser protegidas por criptografia, e o acesso deve ser seguro e restrito a usuários autorizados.
- **RNF3**: Usabilidade – A interface deve ser intuitiva, facilitando a navegação mesmo para usuários com pouca experiência tecnológica. Deve ser compatível com dispositivos móveis.

### 3.3 Regras de Negócio (RN)

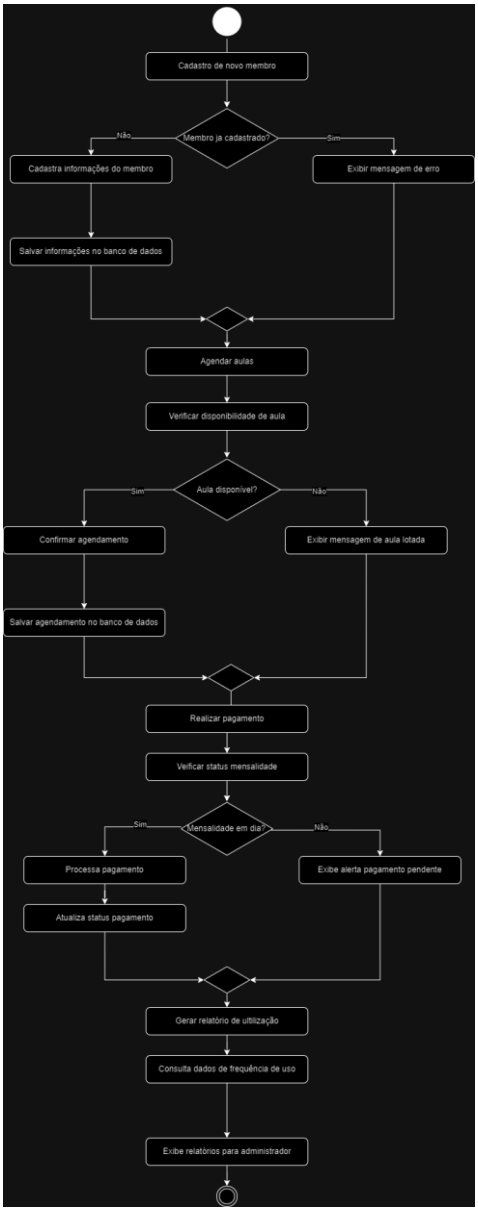
- **RN1**: Somente membros com mensalidades em dia podem agendar novas aulas.
- **RN2**: Novos cadastros requerem a assinatura de um contrato inicial.
- **RN3**: Cancelamentos de aulas podem ser realizados sem custos até 24 horas antes da aula. Cancelamentos com menos de 24 horas resultarão na perda da aula.
- **RN4**: Aplicação automática de descontos – 10% de desconto para pagamento anual adiantado e 5% de desconto para alunos universitários.

# 4. Diagramas

- **4.1. Caso de uso**

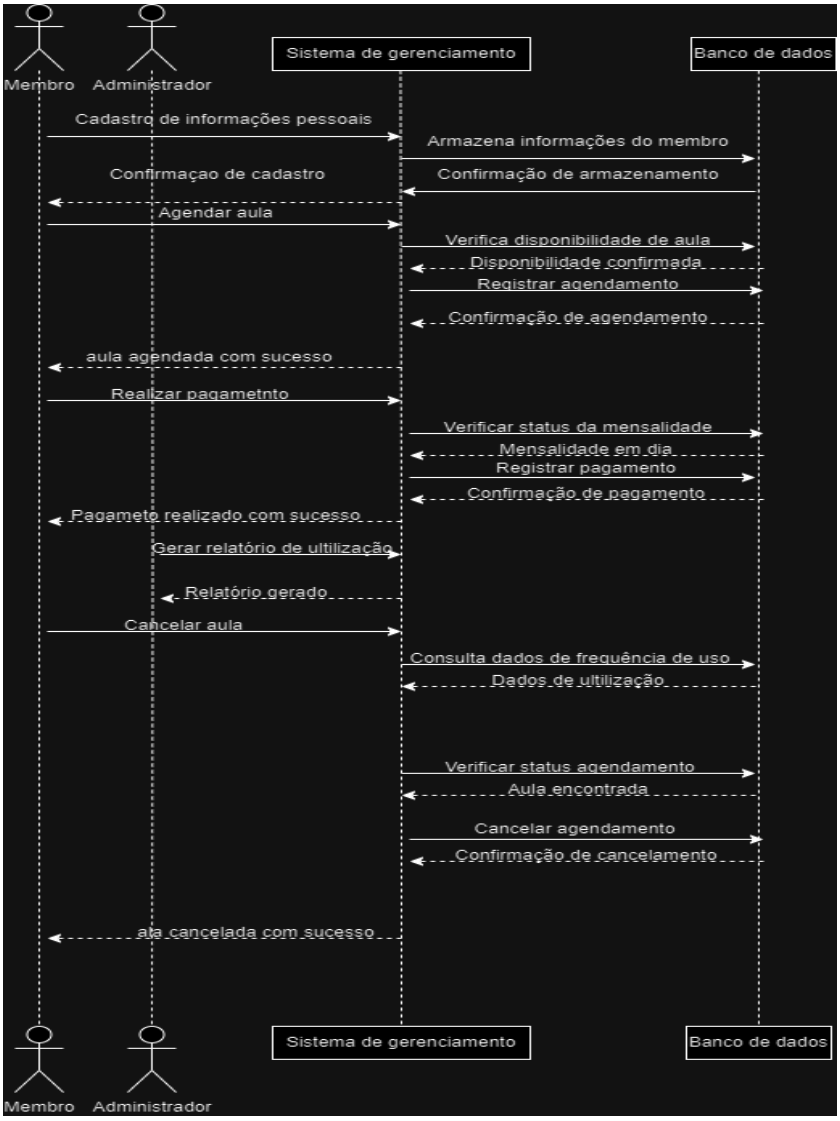


- **4.2. Diagrama de Atividades**

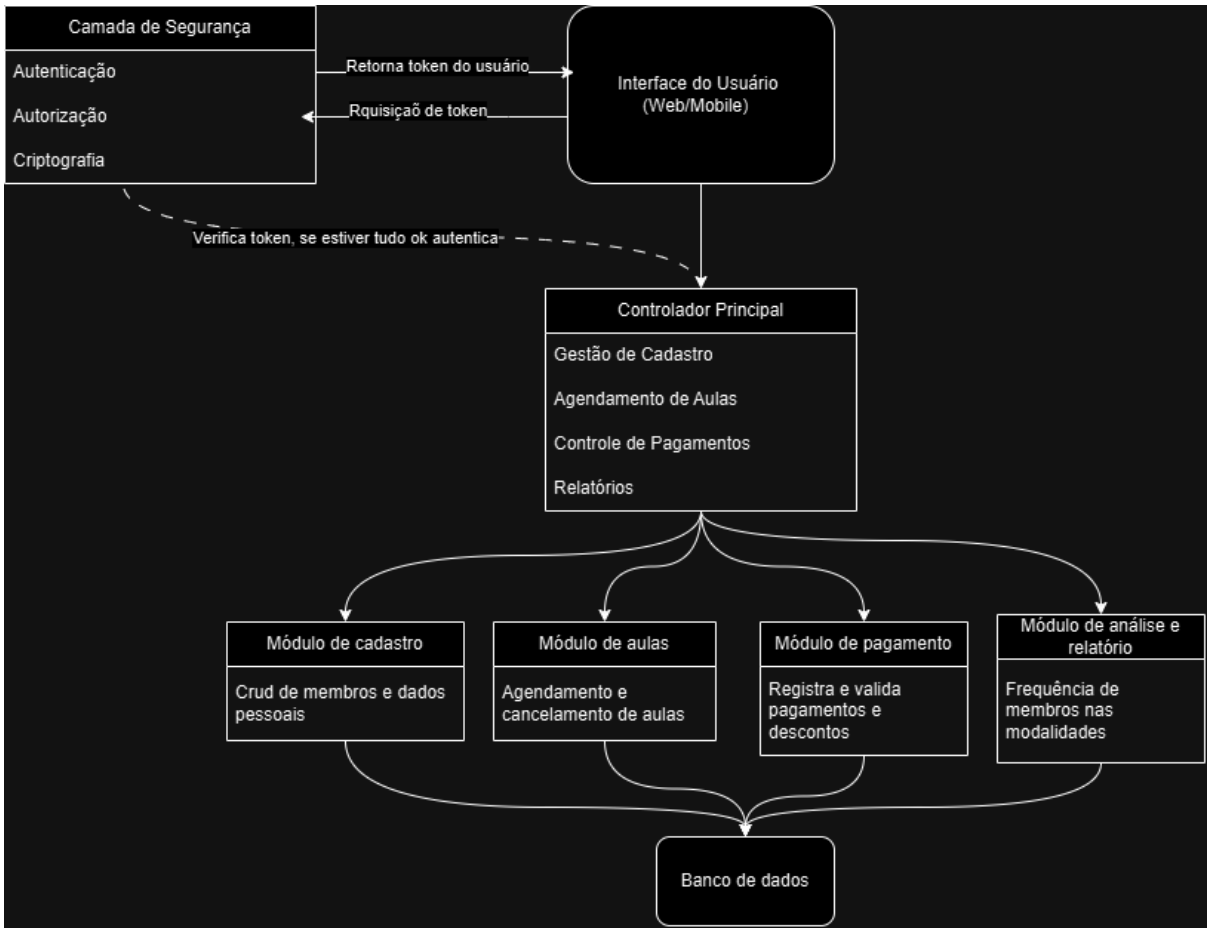


- **4.3. Diagrama de classes**


- **4.4. Diagrama sequencial**



- ## 4.4. Diagrama de componentes



# 5. Design Patterns

## 5.1. Singleton

- **Onde aplicar:**
    - Ideal para gerenciar conexões com bancos de dados ou qualquer recurso caro em termos de desempenho, onde uma única instância é suficiente para a aplicação.

- **Justificativa para aplicar:**
    - O padrão Singleton garante que apenas uma única instância da conexão seja criada, melhorando a eficiência e evitando conflitos de dados. Em ambientes multi-thread, uma implementação adequada pode incluir mecanismos de controle de acesso, aumentando a segurança e a performance da aplicação.

## 5.2. Factory Method

- **Onde aplicar:**
    - Criação de diferentes tipos de contas de usuário (ex.: conta de membro regular, conta de personal trainer, conta de administrador).

- **Justificativa para aplicar:**
    - O Factory Method pode ser usado para instanciar diferentes tipos de contas de usuário com atributos específicos (por exemplo, permissões distintas). Isso facilita a manutenção da lógica de criação e permite a adição de novos tipos de contas sem modificar o código principal.

## 5.2. Observer

- **Onde aplicar:**
    - Sistema de monitoramento da capacidade das aulas.

- **Justificativa para aplicar:**
    - O Observer pode notificar automaticamente os membros sobre a disponibilidade de vagas em uma aula, seja por lotação iminente ou cancelamentos, para que possam se inscrever ou reagendar.

## 5.3. Strategy

- **Onde aplicar:**
    - Políticas de controle de acesso baseadas em diferentes tipos de membros (ex.: acesso total, restrito a determinados horários).

- **Justificativa para aplicar:**
    - O Strategy pode ser usado para definir diferentes políticas de controle de acesso a recursos, como a reserva de aulas apenas em horários específicos para determinados membros. Essa abordagem facilita a implementação de políticas de acesso que podem ser trocadas dinamicamente conforme o tipo de membro ou regras da academia.

## 5.4. Command

- **Onde aplicar:**
    - Processamento de pagamentos e geração de comprovantes.

- **Justificativa para aplicar:**
    - Cada operação de pagamento (ex.: pagamento mensal, pagamento com desconto) pode ser encapsulada em um comando (ProcessarPagamentoCommand), que é então executado para registrar o pagamento e gerar um comprovante. Isso permite manter um histórico de transações e facilita o processamento em lote ou o desfazer de pagamentos em caso de erros.

## 5.5. Decorator

- **Onde aplicar:**
    - Customização de planos de assinatura dos membros.

- **Justificativa para aplicar:**
    - O Decorator pode ser usado para adicionar benefícios ou recursos extras aos planos de assinatura, como aulas VIP ou acesso a equipamentos exclusivos. Isso permite que a academia crie planos de assinatura de forma flexível, decorando um plano base com funcionalidades adicionais (ex.: PlanoBase + AcessoVIPDecorator + DescontoEspecialDecorator).

## 5.6. Facade

- **Onde aplicar:**
    - Interface simplificada para gestão de agendamentos e notificações.

- **Justificativa para aplicar:**
    - O Facade pode ser aplicado para criar uma interface unificada que coordena a interação entre o módulo de agendamento, o sistema de notificações e a verificação de capacidade. Isso ajuda a encapsular a complexidade do processo de agendamento e facilita o uso do sistema para desenvolvedores e usuários.

## 5.7. Template Method

- **Onde aplicar:**
    - Processos de check-in para diferentes tipos de aulas (ex.: check-in para aulas regulares, check-in para aulas especiais com regras adicionais).

- **Justificativa para aplicar:**
    - O Template Method pode ser usado para definir a estrutura geral do processo de check-in, com etapas comuns como verificação de pagamento, confirmação de identidade e atualização de presença. Subclasses podem implementar variações específicas, como um processo de check-in que inclui verificação extra para aulas especiais ou eventos.

# 6. Metodologia de Desenvolvimento

Para o desenvolvimento do **Sistema de Gerenciamento de Academia XYZ**, será utilizada a metodologia ágil **Scrum**, permitindo flexibilidade e entregas incrementais.

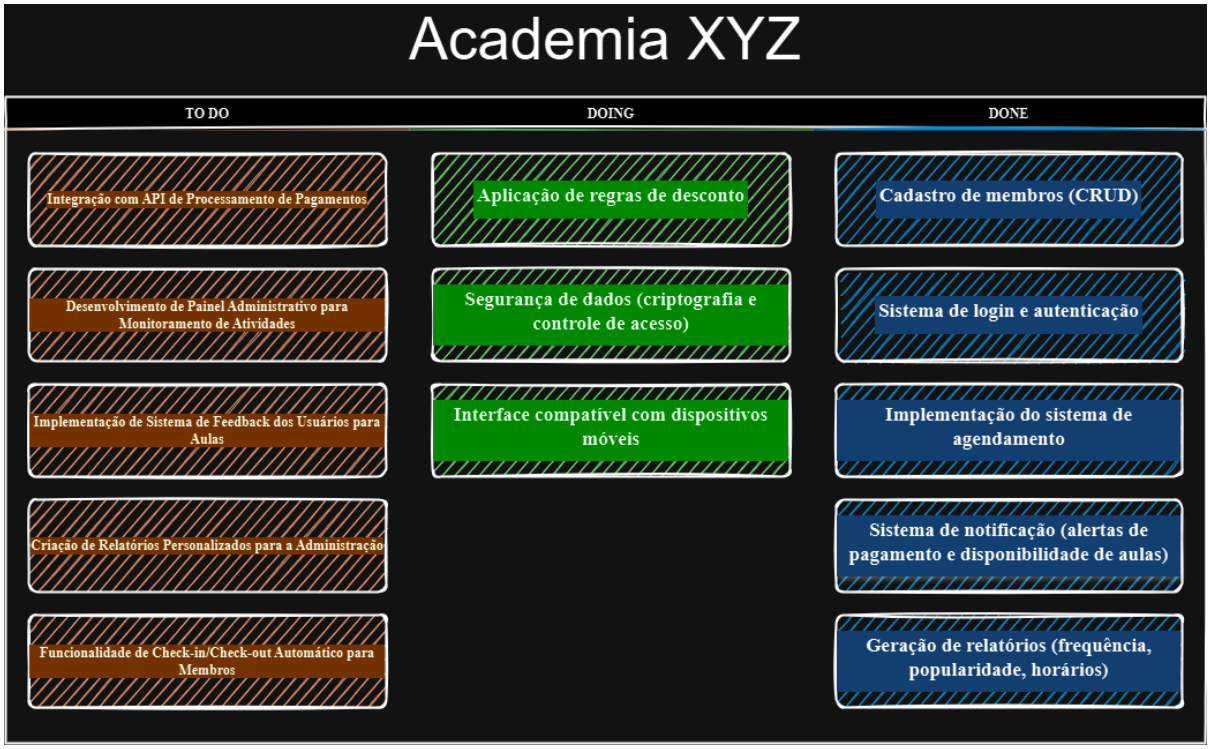
**Papéis e Responsabilidades**:

- **Product Owner**: Representante da Academia XYZ, responsável por priorizar os requisitos.
- **Scrum Master**: Garantirá que a equipe siga as práticas ágeis e removerá impedimentos.
- **Equipe de Desenvolvimento**: Composta por desenvolvedores e testadores, encarregada da implementação.

**Sprints e Ciclo de Desenvolvimento**:

- **Sprints** de 2 semanas com entregas incrementais.
- **Backlog** dividido em **histórias de usuário**, como:
    - "Como membro, quero me cadastrar com minhas informações pessoais para acessar o sistema."
    - "Como administrador, quero gerar relatórios para otimizar a alocação de recursos."

# 7. Quadro Kanban




Este quadro Kanban foi projetado para gerenciar o desenvolvimento do sistema de software da "Academia XYZ", ajudando a equipe a monitorar o progresso das tarefas essenciais. Ele está dividido em 3 colunas principais:

1. **Backlog**: Contém todas as tarefas planejadas que ainda não foram iniciadas.
2. **Em Progresso**: Lista as tarefas que estão sendo ativamente trabalhadas pela equipe.
3. **Concluído**: Exibe as tarefas finalizadas e prontas para uso.

Esse quadro oferece uma visão clara e estruturada do status do projeto, facilitando a priorização de tarefas e a identificação de gargalos no fluxo de trabalho. É uma ferramenta eficaz para manter a equipe alinhada, melhorar a produtividade e garantir que o desenvolvimento do sistema atenda aos prazos e expectativas de qualidade.