# Documentação de Apoio aos Diagramas

## Diagrama de Casos de Uso (UML)

O diagrama de casos de uso representa as interações entre os diferentes tipos de usuários do sistema e suas funcionalidades. Ele contempla os seguintes atores:  
  
- Líder do Clã: possui acesso completo à plataforma. Pode gerenciar membros, eventos, builds, permissões e dados financeiros.  
- Membro do Clã: pode visualizar o dashboard, consultar builds e confirmar presença em eventos.  
- Sistema Externo: integrações com a API do jogo (Albion Online), Discord e serviços de autenticação como Firebase/Auth0.  
  
Esse diagrama facilita o entendimento funcional do sistema, separando claramente as responsabilidades de cada perfil de usuário.

## Modelo C4 – Nível 1: Diagrama de Contexto

Esse nível descreve como o sistema SaaS de Gerenciamento de Clãs se relaciona com os usuários e sistemas externos. Ele evidencia:  
  
- Os usuários principais: Líder do Clã e Membro do Clã;  
- A Plataforma SaaS, que centraliza o gerenciamento de funcionalidades internas;  
- As integrações com serviços externos como:  
 - Albion Online API: para puxar dados do jogo;  
 - Discord API: para envio de notificações automáticas;  
 - Firebase/Auth0: para autenticação de usuários.  
  
Esse nível estabelece os limites da aplicação e os pontos de comunicação externa.

## Modelo C4 – Nível 2: Diagrama de Contêineres

Aqui são apresentados os principais contêineres da aplicação e suas responsabilidades:  
  
- Frontend (Vue.js + Tailwind CSS): responsável pela interface do usuário;  
- Backend (NestJS + Express): camada de negócio e regras de validação;  
- Banco de Dados (PostgreSQL): persistência de dados do sistema;  
- Cache (Redis): otimização de dados temporários e sessões;  
- Serviço de Autenticação (Firebase/Auth0): gerenciamento de login e permissões;  
- Monitoramento (Prometheus + Grafana): análise de métricas e logs do sistema.  
  
As conexões entre os contêineres representam o fluxo de informações interno, reforçando o modelo arquitetural adotado.

## Modelo C4 – Nível 3: Diagrama de Componentes

Foca nos componentes internos do Backend (NestJS):  
  
- Cada módulo possui um Controller, Service e, quando necessário, um Repository;  
- Serviços auxiliares como NotificationService, IntegrationService e LoggerService são apresentados como elementos independentes reutilizáveis;  
- O diagrama evidencia as relações de dependência entre os componentes, como por exemplo:  
 - ReportService depende de FinanceService;  
 - NotificationService utiliza o IntegrationService.  
  
Isso proporciona uma visão mais granular da aplicação e reforça os princípios de baixa acoplamento e alta coesão.

## Modelo C4 – Nível 4: Diagrama de Código

Ilustra a estrutura interna do módulo EventModule, seguindo o padrão do NestJS:  
  
- event.controller.ts: define os endpoints da API.  
- event.service.ts: contém a lógica de negócio.  
- event.repository.ts: abstração da camada de acesso ao banco.  
- dto/: objetos de transferência de dados.  
- schemas/event.entity.ts: define o modelo da tabela.  
- event.module.ts: orquestra todos os componentes do módulo.  
  
As dependências entre arquivos são destacadas por setas, representando importações e chamadas diretas entre as camadas.