**CENTRO UNIVERSITÁRIO CATÓLICA DE SANTA CATARINA**

**ENGENHARIA DE SOFTWARE**

**LEONARDO PEREIRA BORGES**

**Plataforma de Crowdfunding Simplificada**

**Joinville**

**2025**

**Resumo**

Este documento apresenta o desenvolvimento de uma plataforma web de crowdfunding que permite aos usuários criarem campanhas de financiamento coletivo e apoiar projetos de terceiros. A solução combina tecnologias modernas como Vue.js (Quasar), Node.js, PostgreSQL e integração com Stripe para pagamentos, oferecendo uma experiência intuitiva e segura para criadores e apoiadores. O projeto demonstra a aplicação prática de princípios de Engenharia de Software, incluindo arquitetura escalável, segurança da informação, design responsivo e integração com serviços externos.

1. **Introdução**

**Contexto**

O crowdfunding emergiu como uma técnica revolucionária de financiamento coletivo, permitindo que indivíduos e organizações obtenham recursos financeiros através da colaboração voluntária de uma comunidade online. Com o avanço tecnológico e a crescente conectividade digital, este modelo se consolidou globalmente, democratizando o acesso ao financiamento para projetos inovadores, causas sociais e empreendimento criativos.

**Justificativa**

A criação de uma plataforma web de crowdfunding representa uma excelente oportunidade para aplicar na prática os fundamentos da Engenharia de Software. Este tipo de sistema exige a integração de diversos conceitos técnicos essenciais: arquitetura de software robusta, segurança da informação, gerenciamento eficiente de dados, design responsivo focado na experiência do usuário, e integração confiável com serviços externos como gateways de pagamento. Além disso, o projeto permite explorar aspectos de escalabilidade, performance e conformidade com regulamentações de proteção de dados.

**Objetivos**

**Objetivo Principal:**

* Desenvolver uma aplicação web completa que facilite o cadastro, visualização e apoio financeiro a projetos de forma simples, segura e acessível.

**Objetivos Específicos:**

* Implementar sistema de autenticação seguro com gerenciamento de usuários via Clerk
* Criar interface moderna e responsiva utilizando Quasar Framework (Vue.js)
* Desenvolver API RESTful robusta com Node.js e Express
* Integrar sistema de pagamentos confiável através do Stripe
* Implementar funcionalidades de comentários e interação social
* Garantir persistência segura de dados com PostgreSQL
* Estabelecer pipeline de CI/CD com testes automatizados

1. **Descrição do Projeto**

**Linha de Projeto**

**Web Apps -** Aplicação web completa com frontend e backend integrados.

**Tema do Projeto**

Plataforma de crowdfunding que conecta criadores de projetos com potenciais apoiadores, facilitando o financiamento coletivo através de uma interface web intuitiva e segura. A solução permite que usuários registrem campanhas com metas financeiras específicas e que a comunidade contribua financeiramente para projetos de seu interesse.

**Propósito e Uso Prático**

A plataforma resolve a necessidade de uma solução acessível para financiamento coletivo, permitindo que:

* **Criadores**lancem campanhas detalhadas com metas claras e prazos definidos
* **Apoiadores** descubram e contribuam para projetos alinhados com seus interesses
* **Comunidade** acompanhe o progresso das campanhas e interaja através de comentários

O sistema será utilizado por empreendedores, artistas, desenvolvedores de jogos, criadores de conteúdo e organizações que buscam financiamento alternativo para seus projetos.

**Público-Alvo**

* **Criadores de Projetos:** Empreendedores, artistas, desenvolvedores, organizações sem fins lucrativos
* **Apoiadores:** Indivíduos interessados em apoiar inovação, arte, tecnologia e causas sociais
* **Faixa Etária:** 18-55 anos, com conhecimento básico de tecnologia
* **Perfil:** Usuários com acesso à internet e métodos de pagamento digital

**Problemas a Resolver**

* **Baixa visibilidade:** Projetos pequenos têm dificuldade para alcançar audiência
* **Barreira técnica:** Criadores sem conhecimento técnico enfrentam dificuldades para criar campanhas
* **Falta de controle:** Ausência de ferramentas para gerenciar campanhas pessoais
* **Desconfiança:** Receio de fraudes ou promessas não cumpridas
* **Transparência limitada:** Falta de clareza sobre uso dos recursos arrecadados

**Diferenciação/Ineditismo**

* **Interface Simplificada:** Design intuitivo que reduz a curva de aprendizado
* **Integração Nativa:** Autenticação via Clerk e pagamentos via Stripe integrados nativamente
* **Transparência Total:** Sistema de comentários público e acompanhamento em tempo real
* **Performance Otimizada:** Arquitetura moderna com foco em velocidade e responsividade
* **Código Aberto:** Desenvolvimento transparente com documentação completa

**Limitações**

* Sistema de gerenciamento financeiro básico (sem análises avançadas)
* Segurança limitada a boas práticas padrão (sem autenticação multifator)
* Disponível inicialmente apenas em português
* Sem versão mobile nativa (apenas web responsiva)
* Moeda única (Real brasileiro)
* Sem suporte internacional inicial

**Normas e Legislações Aplicáveis**

* **LGPD (Lei Geral de Proteção de Dados):** Implementação de políticas de privacidade e controle de dados pessoais
* **Marco Civil da Internet:** Conformidade com princípios de neutralidade e privacidade
* **Código de Defesa do Consumidor:** Transparência nas transações financeiras
* **Regulamentação do Banco Central:** Conformidade com normas de pagamentos eletrônicos

**Métricas de Sucesso**

* **Performance:** Tempo de carregamento < 2 segundos
* **Disponibilidade:** Uptime ≥ 99.5%
* **Usabilidade:** Taxa de conclusão de cadastro de campanha ≥ 85%
* **Conversão:** Taxa de conversão visitante → apoiador ≥ 5%
* **Qualidade:** Cobertura de testes ≥ 80%
* **Segurança:** Zero incidentes críticos de segurança

**3. Especificação Técnica**

Esta aplicação seguirá uma **arquitetura monolítica**, usando **Node.js (Express)** no backend, **Quasar Framework (Vue 3)** no frontend e **PostgreSQL** como banco de dados relacional.

* **Autenticação e gerenciamento de usuários com Clerk (Vue.js)**, integrando o estado de sessão no frontend e a validação de token no backend;
* **Banco de dados PostgreSQL,** mantendo acesso pelo Prisma ORM.

O **Quasar** foi escolhido por sua competência em criar interfaces responsivas voltadas para desempenho e UX. O **Node.js** oferece uma estrutura leve e eficiente para as regras de negócio e integração com o banco. O **Clerk** centraliza a autenticação/gestão de contas de usuário e o com PostgreSQL.

**Requisitos do Tema Proposto**

A seção de Requisitos de Software apresenta os componentes funcionais e não funcionais do software. Esses requisitos são fundamentais para assegurar que o aplicativo satisfaça as demandas dos usuários.

**Requisitos Funcionais (RF)**

|  |  |  |
| --- | --- | --- |
| **Código** | **Prioridade** | **Requisito Funcional** |
| RF01 | Must | **Cadastro de usuários:** O sistema deve permitir registro via interface Clerk com validação de campos e integração automática com PostgreSQL para dados de perfil essenciais. |
| RF02 | Must | |  | | --- | | **Autenticação segura:** Sistema de login via Clerk com geração de JWT para sessões e validação de token no backend para rotas protegidas. |  |  | | --- | |  | |
| RF03 | Must | **Criação de campanhas:** Usuários autenticados podem criar campanhas com título, descrição, meta financeira, prazo, categoria e imagem ilustrativa. |
| RF04 | Should | **Gerenciamento de campanhas:** Criadores podem editar ou excluir suas campanhas ativas, com controle de permissões baseado em propriedade. |
| RF05 | Must | **Listagem pública:** Exibição de todas as campanhas ativas com informações resumidas, filtros por categoria e busca textual. |
| RF06 | Must | |  | | --- | | **Visualização detalhada:** Página completa da campanha com descrição, progresso, criador, prazo e botão de apoio. |  |  | | --- | |  | |
| RF07 | Must | **Sistema de apoio:** Usuários autenticados podem realizar contribuições financeiras com valores personalizados. |
| RF08 | Must | **Registro de contribuições:** Persistência de todos os apoios relacionando usuário, campanha, valor e status da transação. |
| RF09 | Could | **Dashboard pessoal:** Painel individual com campanhas criadas, valores arrecadados e histórico de apoios realizados. |
| RF10 | Could | **Checkout Stripe:** Integração com Stripe Checkout para processamento seguro de pagamentos. |
| RF11 | Could | **Sistema de comentários:** Comentários públicos em campanhas para usuários autenticados com moderação básica. |
| RF12 | Could | **Categorização:**Sistema de categorias para organização e filtragem de campanhas. |

**Requisitos Não-Funcionais (RNF)**

|  |  |  |
| --- | --- | --- |
| **Código** | **Prioridade** | **Requisito Não-Funcional** |
| RNF01 | Must | **Responsividade:** Interface adaptável para desktops, tablets e smartphones com breakpoints otimizados. |
| RNF02 | Must | **Segurança:** Autenticação via Clerk, validação de JWT no backend, sanitização de inputs e proteção contra-ataques comuns. |
| RNF03 | Must | |  | | --- | | **Usabilidade:** Interface intuitiva com navegação fluida, feedback visual e acessibilidade básica (WCAG 2.1 AA). |  |  | | --- | |  | |
| RNF04 | Must | |  | | --- | | **Performance:** Carregamento inicial < 2s, lazy loading de imagens, otimização de bundle e cache adequado. |  |  | | --- | |  | |
| RNF05 | Should | |  | | --- | | **Escalabilidade:** Arquitetura preparada para crescimento com índices de banco otimizados e estrutura modular. |  |  | | --- | |  | |
| RNF06 | Should | |  | | --- | | **Disponibilidade:** Uptime de 99.5% com monitoramento ativo e recuperação automática de falhas. |  |  | | --- | |  | |
| RNF07 | Must | |  | | --- | | **Compatibilidade:** Suporte aos navegadores modernos (Chrome, Firefox, Safari, Edge) nas últimas 2 versões. |  |  | | --- | |  | |

**Representação dos Requisitos**

Os requisitos são representados através de três diagramas de casos de uso:

1. **Acesso Público:**Funcionalidades disponíveis para visitantes não autenticados
2. **Usuário Autenticado:**Operações completas para usuários logados
3. **Apoio a Campanhas:**Fluxo específico de contribuições financeiras

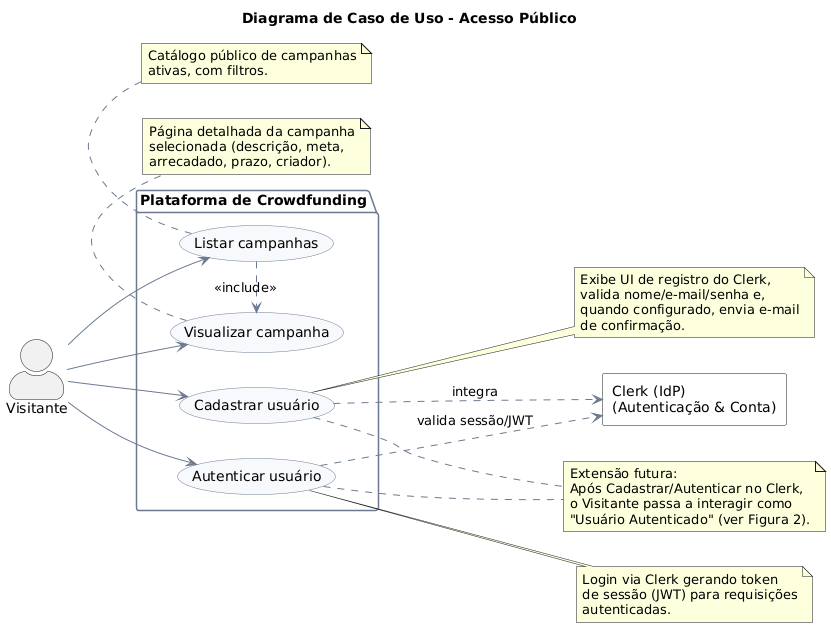


Figura 1: Casos de Uso para Visitante no Sistema Crowdfunding

Na **Figura 1** é apresentado o conjunto mínimo de funcionalidades disponíveis a qualquer usuário anônimo (Visitante) que acessa a Plataforma sem efetuar login. Agora, o **cadastro** e a **autenticação** são realizados pela **UI do Clerk** integrada ao frontend (Vue/Quasar).

* **Cadastrar usuário:** Exibe a interface de registro do **Clerk**, validando nome, e-mail e senha; conclui com a criação de conta e envio de e-mail de confirmação (quando configurado).
* **Autenticar Usuário:** Permite login via **Clerk**, gerando **token de sessão (JWT)** para as requisições autenticadas.
* **Listar Campanhas:** Apresenta catálogo público de campanhas ativas com filtros.
* **Visualizar Campanha:** Mostra a página detalhada da campanha selecionada.

**Relações entre casos de uso**

* **«include»** listar campanhas → Visualizar campanha.
* **Extensão futura:** Após Cadastrar/Autenticar no **Clerk**, o ator passa a interagir como Usuário Autenticado (Figura 2).

**Visão geral**

Mesmo sem autenticação, a plataforma oferece valor permitindo explorar campanhas e incentiva o cadastro para liberar funcionalidades avançadas, em conformidade com os RF01 a RF06.

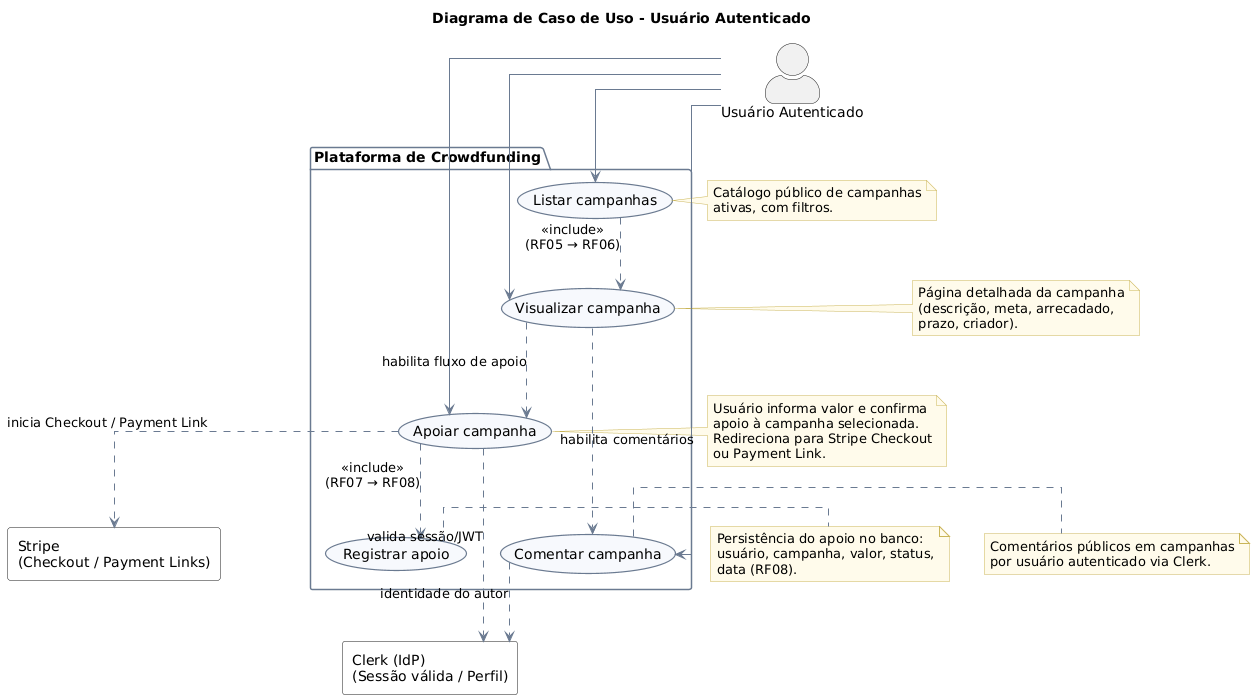


Figura 2: Casos de Uso para Usuário Autenticado no Sistema Crowdfunding

**Na Figura 2**é apresentado o diagrama com as funcionalidades para quem já realizou login (sessão do **Clerk** válida). Os casos concentram-se em navegação, apoio a campanhas e comentários, mantendo a integração de pagamento **via Stripe Checkout/Payment Links.**

* **Listar campanhas**→ «include»**Visualizar campanha (**RF05 → RF06).
* **Visualizar campanha:** ponto de partida para**Apoiar campanha**(RF07) **e Comentar campanha.**
* **Apoiar campanha**(RF07) inclui:
  + **Registrar apoio**(RF08).
  + **Iniciar Checkout/Payment Link (Stripe)**(RF10).
* **Comentar campanha:** comentários públicos (usuário autenticado via Clerk).

**Dependências externas**

* **Clerk**(autenticação e perfil do usuário).
* **Stripe**(Checkout/Payment Links).

**Visão geral**

Após autenticado pelo **Clerk**, o usuário controla o ciclo de vida de suas campanhas e interage com a comunidade, atendendo integralmente os requisitos funcionais.

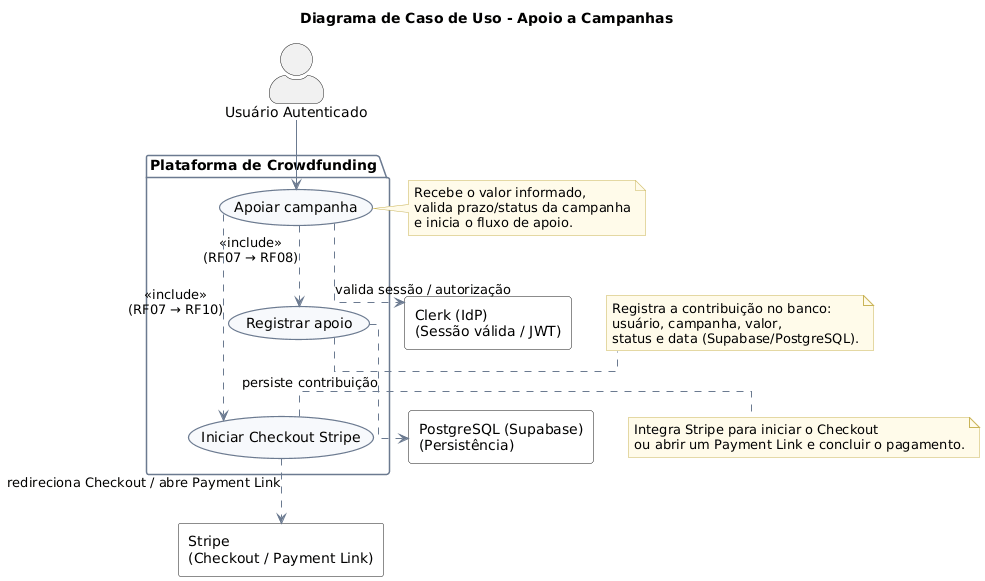


Figura 3: Caso de Uso para Apoio à Campanhas e Pagamento na Plataforma Crowdfunding

Na **Figura 3** é apresentado o fluxo completo quando um **Usuário Autenticado** (sessão **Clerk** válida) decide contribuir financeiramente para uma campanha. A **persistência** é feita no **PostgreSQL.**

**A persistência é feita no PostgreSQL.**

* **Apoiar campanha:**recebe o valor, valida prazo/status e inicia registro e abertura do **Checkout/Payment Link.**
* **Registrar apoio**: persiste contribuição(usuário, campanha, valor, status) no**PostgreSQL.**
* **Iniciar Checkout/Payment Link:**integra Stripe para criar/abrir o link de pagamento **(Checkout/Payment Link)** e concluir o pagamento**.**

**Relações**

* **«include»**Apoiar campanha → Registrar apoio.
* **«include»**Apoiar campanha → Iniciar Checkout/Payment Link.

**Visão geral**

O sistema garante rastreabilidade dos apoios e transparência dos valores, delegando o pagamento ao provedor externo.

Com esses três diagramas, temos uma visão completa dos Requisitos Funcionais do Sistema Crowdfunding, desde o acesso público até as operações mais específicas de criação e apoio a campanhas.

**Aderência aos Requisitos da Linha de Projeto**

**Web Apps Critérios Obrigatórios Atendidos:**

* Interface web responsiva (Quasar Framework)
* Backend com API RESTful (Node.js + Express)
* Banco de dados relacional (PostgreSQL)
* Autenticação e autorização (Clerk + JWT)
* Integração com serviços externos (Stripe)
* Testes automatizados (Jest + Vitest)
* Deploy automatizado (CI/CD via GitHub Actions)

**3.2. Considerações de Design**

**Visão Inicial da Arquitetura**

A aplicação segue arquitetura em três camadas:

**Camada de Apresentação (Frontend):**

* Single Page Application (SPA) com Quasar Framework (Vue.js 3)
* Gerenciamento de estado com Pinia
* Autenticação integrada via Clerk Vue SDK
* Design system responsivo com componentes reutilizáveis

**Camada de Aplicação (Backend):**

* API RESTful com Node.js e Express
* Middleware de autenticação com validação JWT
* Integração com Stripe para pagamentos
* Validação de dados com Zod

**Camada de Dados (Persistência):**

* PostgreSQL como banco principal
* ORM Prisma para modelagem e queries
* Índices otimizados para performance
* Migrations controladas e visionadas

**Diagrama

O conteúdo gerado por IA pode estar incorreto.**

Figura 4: Visão Arquitetura Simples

**Padrões de Arquitetura**

* RESTful API: Endpoints padronizados com verbos HTTP semânticos
* MVC (Model-View-Controller): Separação clara de responsabilidades
* Repository Pattern: Abstração da camada de dados
* Middleware Pattern: Processamento de requisições em pipeline
* Component-Based Architecture: Interface modular e reutilizável
* Service Layer: Lógica de negócio centralizada

**Modelos C4**

**Nível 1 - Contexto:** Sistema de crowdfunding conectando usuários, integrando Clerk (autenticação), Stripe (pagamentos) e banco de dados (PostgreSQL).

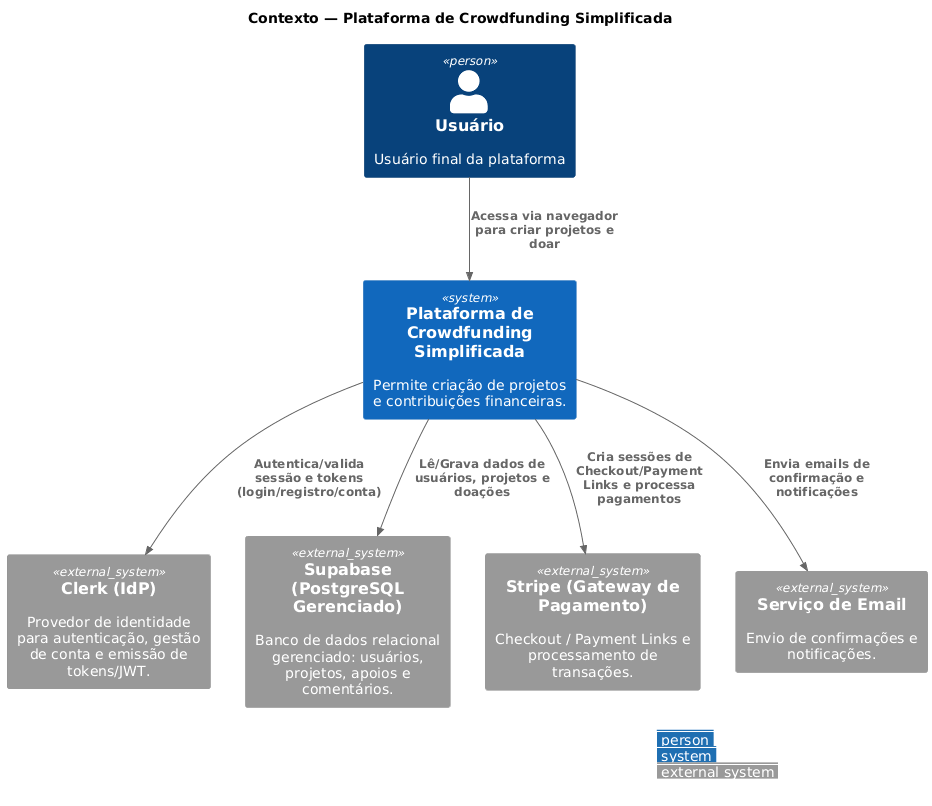
****

Figura 5:Diagrama de Contexto da Plataforma de Crowdfunding

**Nível 2 - Contêineres:**

* **Frontend Web:**Quasar/Vue + Clerk SDK
* **API Backend:**Express + Prisma + validação JWT
* **Banco PostgreSQL:**banco de dados relacional
* **Serviços Externos:**Stripe + Clerk + Email

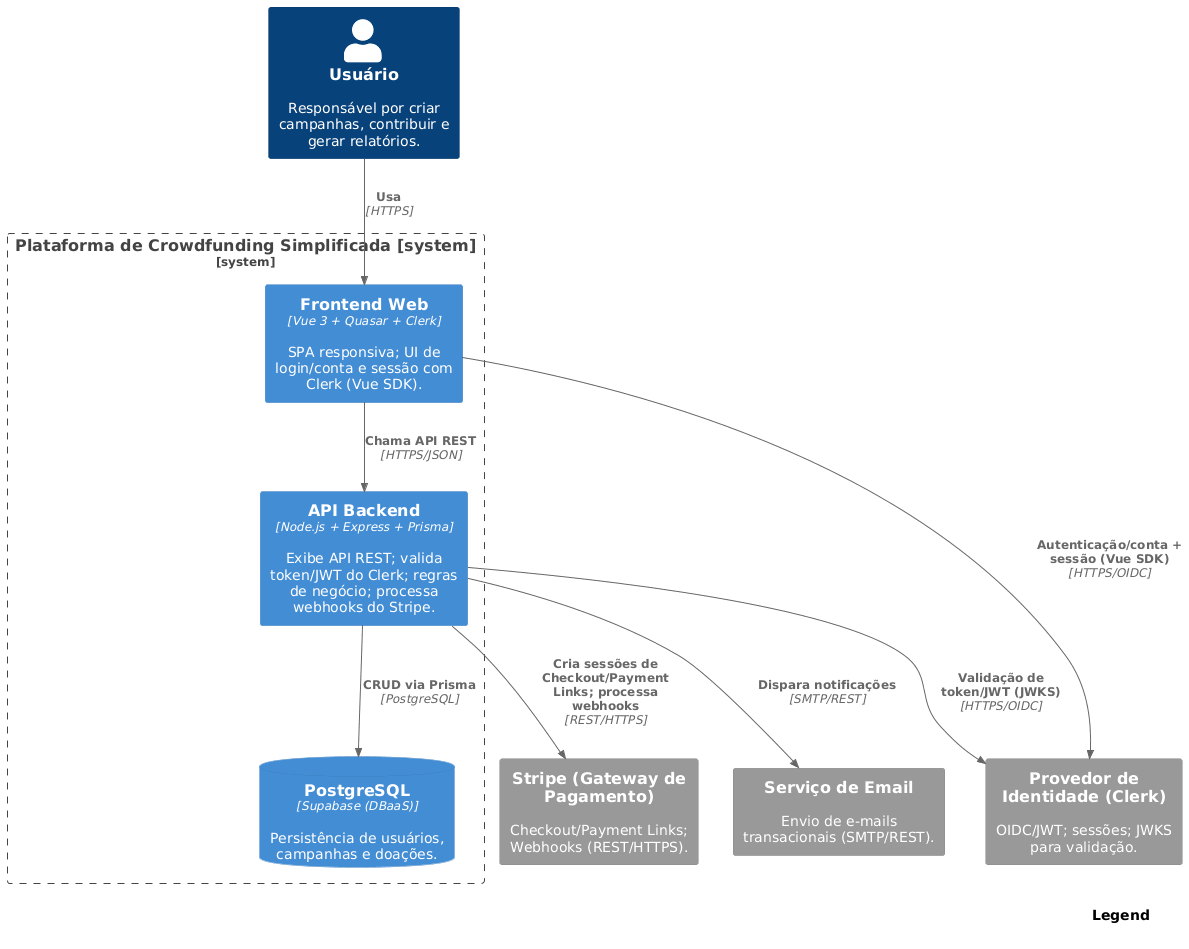
****

Figura 6: Diagrama de Contêineres da Plataforma de Crowdfunding

**Mockups das Telas Principais**

**Tela Inicial:**

* Hero section com call-to-action
* Grid de campanhas em destaque
* Seção "Como funciona" com passos ilustrados
* Estatísticas da plataforma em tempo real

**Lista de Campanhas:**

* Grid responsivo com filtros por categoria
* Busca textual avançada
* Ordenação por relevância, data, progresso
* Paginação otimizada

**Detalhes da Campanha:**

* Layout em duas colunas (desktop) / stack (mobile)
* Galeria de imagens com zoom
* Barra de progresso animada
* Seção de comentários interativa
* Call-to-action destacado para apoio

**Dashboard do Usuário:**

* Visão geral com cards de estatísticas
* Lista de campanhas criadas com status
* Histórico de apoios realizados
* Gráficos de performance (quando aplicável)

**Decisões e Alternativas Consideradas**

**Frontend Framework:**

* Escolhido: Quasar (Vue.js) - Componentes prontos, performance, comunidade ativa
* Alternativas: React + Material-UI, Angular + Angular Material
* Justificativa: Produtividade de desenvolvimento e qualidade visual

**Backend Framework:**

* Escolhido: Node.js + Express - Ecossistema JavaScript unificado, performance adequada
* Alternativas: Python + FastAPI, Java + Spring Boot
* Justificativa: Consistência tecnológica e time-to-market

**Banco de Dados:**

* Escolhido: PostgreSQL - ACID, performance, recursos avançados
* Alternativas: MySQL, MongoDB
* Justificativa: Consistência transacional crítica para pagamentos

**Autenticação:**

* Escolhido: Clerk - Solução completa, segura, fácil integração
* Alternativas: Auth0, Firebase Auth, implementação própria
* Justificativa: Foco no core business, segurança garantida

**Critérios de Escalabilidade, Resiliência e Segurança**

**Escalabilidade:**

* Arquitetura stateless permitindo scaling horizontal
* Índices de banco otimizados para queries frequentes
* CDN para assets estáticos
* Lazy loading e paginação para grandes datasets

**Resiliência:**

* Tratamento de erros padronizado com códigos HTTP apropriados
* Retry logic para integrações externas
* Graceful degradation para funcionalidades não críticas
* Monitoring e alertas automatizados

**Segurança:**

* Validação rigorosa de inputs (sanitização + validação)
* Rate limiting para prevenção de ataques
* HTTPS obrigatório em produção
* Secrets management adequado
* Logs de auditoria para ações críticas

**3.3. Stack Tecnológica**

**Linguagens de Programação**

**TypeScript:**Linguagem principal para frontend, oferecendo tipagem estática, melhor tooling e redução de bugs em runtime.

**JavaScript (Node.js):**Backend com runtime otimizado, vasto ecossistema de packages e performance adequada para I/O intensivo.

**Frameworks e Bibliotecas**

**Frontend:**

* **Quasar Framework 2.16+:**Framework Vue.js com componentes Material Design, build otimizado e suporte PWA
* **Vue.js 3.4+:**Framework reativo com Composition API, performance superior e TypeScript nativo
* **Pinia 3.0+:**Store management moderno, type-safe e com DevTools integrado
* **Vue Router 4.0+:**Roteamento SPA com guards, lazy loading e meta fields
* **Clerk Vue SDK 1.13+:**Autenticação completa com UI components e session management
* **Axios 1.11+:**Cliente HTTP com interceptors, retry logic e TypeScript support

**Backend:**

* **Node.js 20+ LTS:**Runtime JavaScript com performance otimizada e security patches
* **Express 4.18+:**Framework web minimalista, extensível e com vasto ecossistema
* **Prisma 5.7+:**ORM type-safe com schema-first approach e migrations automatizadas
* **Clerk Express 1.0+:**Middleware de autenticação com validação JWT automática
* **Stripe SDK 18.5+:**Integração oficial com Stripe para pagamentos seguros
* **Zod 3.22+:**Validação de schemas com TypeScript inference automático

**Ferramentas de Desenvolvimento e Gestão**

**Desenvolvimento:**

* **Visual Studio Code:**IDE com extensões específicas para Vue, TypeScript e Prisma
* **Vite**: Build tool com HMR ultra-rápido e otimizações automáticas
* **ESLint + Prettier:**Linting e formatação de código padronizada
* **TypeScript Compiler:**Type checking e transpilação

**Testes:**

* **Jest 30.1+:**Framework de testes para backend com coverage reporting
* **Vitest 1.0+:**Test runner para frontend com compatibilidade Vite
* **Supertest 7.1+:**Testes de API com assertions HTTP
* **Vue Test Utils 2.4+:**Utilitários para testes de componentes Vue

**DevOps e Monitoramento:**

* **Git + GitHub:**Controle de versão com GitHub Flow
* **GitHub Actions:**CI/CD automatizado com deploy contínuo
* **SonarQube:**Análise de qualidade e cobertura de código
* **Postman:**Documentação e testes manuais de API

**Licenciamento**

**Dependências Principais:**

* **MIT License:**Vue.js, Quasar, Express, Prisma, Jest, Axios
* **Apache 2.0:**TypeScript, ESLint
* **ISC License:**Node.js packages diversos
* **Proprietário:**Clerk (SaaS), Stripe (SaaS), PostgreSQL (SaaS com tier gratuito)

**Projeto:**

* **MIT License: Código fonte disponível publicamente no GitHub**

**3.4. Considerações de Segurança**

**Riscos Identificados**

**Críticos:**

* **Injeção SQL:**Queries maliciosas via inputs não sanitizados
* **Cross-Site Scripting (XSS**): Execução de scripts maliciosos no browser
* **Vazamento de dados:**Exposição de informações pessoais ou financeiras
* **Falhas de autenticação:**Bypass de controles de acesso

**Altos:**

* **Cross-Site Request Forgery (CSRF):**Ações não autorizadas em nome do usuário
* **Ataques de força bruta:**Tentativas massivas de login
* **Man-in-the-Middle:**Interceptação de comunicações
* **Denial of Service (DoS):**Sobrecarga intencional do sistema

**Médios:**

* **Exposição de informações:**Logs com dados sensíveis
* **Configurações inseguras:**Defaults de segurança inadequados
* **Dependências vulneráveis:**Packages com falhas conhecidas

**Medidas de Mitigação**

**Prevenção de Injeção:**

* **Prisma ORM:**Queries parametrizadas automáticas
* **Zod Validation:**Sanitização e validação rigorosa de inputs
* **Prepared Statements:**Separação entre código e dados

**Proteção XSS:**

* **Content Security Policy (CSP):**Headers restritivos para scripts
* **Sanitização HTML:**Escape de caracteres especiais em outputs
* **Vue.js Protection:**Template engine com escape automático

**Segurança de Autenticação:**

* **Clerk Integration:**Provedor especializado em auth security
* **JWT Validation:**Verificação de assinaturas e expiração
* **HTTPS Enforcement:**Criptografia em trânsito obrigatória
* **Secure Cookies:**Flags HttpOnly e Secure ativadas

**Proteção contra-ataques:**

* **Rate Limiting:**Throttling de requests por IP/usuário
* **CORS Configuration:**Whitelist de origens permitidas
* **Input Size Limits:**Prevenção de payloads excessivos
* **Error Handling:**Mensagens genéricas sem exposição de stack traces

**Normas e Boas Práticas Seguidas**

**OWASP Top 10 2021:**

* **A01:** Broken Access Control - Controle via JWT + Clerk
* **A02:** Cryptographic Failures - HTTPS + Hashing adequado
* **A03:** Injection - Prisma ORM + validação Zod
* **A04:** Insecure Design - Security by design desde arquitetura
* **A05:** Security Misconfiguration - Configurações padronizadas
* **A06:** Vulnerable Components - Dependabot + audits regulares
* **A07:** Authentication Failures - Clerk + JWT validation
* **A08:** Software Integrity Failures - Package lock files + checksums
* **A09:** Logging Failures - Structured logging sem dados sensíveis
* **A10:** Server-Side Request Forgery - Validação de URLs externas

**ISO/IEC 27001 Principles:**

* **Confidencialidade:**Criptografia de dados sensíveis
* **Integridade:**Checksums e validações de consistência
* **Disponibilidade:**Monitoring e recovery procedures

**Responsabilidade Ética**

**Transparência:**

* Política de privacidade clara e acessível
* Termos de uso em linguagem simples
* Comunicação proativa sobre uso de dados

**Consentimento:**

* Opt-in explícito para coleta de dados
* Granularidade de permissões
* Direito de revogação facilitado

**Minimização:**

* Coleta apenas de dados necessários
* Retenção limitada por tempo definido
* Pseudonimização quando possível

**3.5. Conformidade e Normas Aplicáveis**

**LGPD - Lei Geral de Proteção de Dados**

**Coleta Minimizada:**

* Apenas dados essenciais: nome, email, dados de campanha
* Evitar dados sensíveis desnecessários
* Justificativa clara para cada campo coletado

**Consentimento Explícito:**

* Checkbox específico para termos de privacidade
* Linguagem clara e não ambígua
* Possibilidade de revogação simples

**Direitos do Titular:**

* **Acesso:** API para consulta de dados pessoais
* **Correção:** Interface para edição de perfil
* **Exclusão:** Funcionalidade de delete account
* **Portabilidade:** Export de dados em formato estruturado

**Segurança Técnica:**

* Criptografia em trânsito (HTTPS)
* Pseudonimização de dados não críticos
* Logs de auditoria para acesso a dados pessoais
* Backup com mesmas proteções dos dados originais

**Marco Civil da Internet**

**Neutralidade:**

* Tratamento equitativo de todos os usuários
* Sem discriminação por conteúdo ou origem
* Transparência em algoritmos de ordenação

**Privacidade:**

* Comunicação criptografada
* Não compartilhamento com terceiros sem consentimento
* Logs mínimos necessários para operação

**Código de Defesa do Consumidor**

**Transparência Financeira:**

* Exibição clara de taxas e comissões
* Detalhamento do processo de pagamento
* Política de reembolso explícita

**Informações Claras:**

* Descrições precisas de campanhas
* Status de projetos atualizados
* Canais de contato acessíveis

**Regulamentação do Banco Central**

**PCI DSS Compliance:**

* Não armazenamento de dados de cartão
* Delegação completa ao Stripe
* Certificação SSL/TLS adequada

**Prevenção à Lavagem de Dinheiro:**

* Limites de transação apropriados
* Logs de transações para auditoria
* Cooperação com autoridades quando necessário

1. **Próximos Passos**

**Cronograma Portfólio I (Concluído)**

* **Semanas 1-2:** Análise de requisitos e documentação técnica
* **Semanas 3-4:** Setup do ambiente e arquitetura base
* **Semanas 5-6:** Implementação do MVP (autenticação + CRUD campanhas)
* **Semanas 7-8:** Integração Stripe e sistema de comentários
* **Semanas 9-10:** Testes, documentação e deploy

**Cronograma Portfólio II (Em Planejamento)**

* **Semanas 11-12:** Otimizações de performance e UX
* **Semanas 13-14:** Sistema de notificações e analytics básico
* **Semanas 15-16:** Melhorias de acessibilidade e SEO
* **Semanas 17-18:** Testes de carga e monitoramento
* **Semanas 19-20:** Documentação final e apresentação

**Definição de Marcos**

**Marco 1 - MVP Funcional (Concluído)**

* Sistema de autenticação operacional
* CRUD completo de campanhas
* Interface responsiva básica
* Integração com banco de dados

**Marco 2 - Integração Completa (Concluído)**

* Sistema de pagamentos via Stripe
* Comentários em campanhas
* Dashboard de usuário
* Deploy automatizado

**Marco 3 - Otimização (Em Andamento)**

* Performance otimizada (< 2s loading)
* Cobertura de testes ≥ 80%
* Monitoramento em produção
* Documentação técnica completa

**Marco 4 - Entrega Final (Planejado)**

* Todas as funcionalidades testadas
* Conformidade com normas aplicáveis
* Apresentação e demonstração
* Código documentado e versionado

**Riscos e Contingências**

**Riscos Técnicos:**

* Integração Stripe complexa → Documentação detalhada + sandbox
* Performance em escala → Testes de carga antecipados
* Compatibilidade cross-browser → Testes automatizados

**Riscos de Prazo:**

* Scope creep → Backlog priorizado e timeboxing
* Dependências externas → Mocks e fallbacks
* Debugging complexo → Logging estruturado

1. **Referências**

**Frameworks e Bibliotecas**

* Vue.js Documentation. Disponível em: <https://vuejs.org/>
* Quasar Framework Documentation. Disponível em: <https://quasar.dev/>
* Express.js Documentation. Disponível em: <https://expressjs.com/>
* Prisma Documentation. Disponível em: <https://www.prisma.io/docs>
* Stripe API Documentation. Disponível em: <https://stripe.com/docs/api>
* Clerk Documentation. Disponível em: <https://clerk.com/docs>

**Padrões e Boas Práticas**

* OWASP Top 10 2021. Disponível em: <https://owasp.org/Top10/>
* REST API Design Guidelines. Microsoft. 2023
* Vue.js Style Guide. Disponível em: <https://vuejs.org/style-guide/>
* Node.js Best Practices. Goldbergyoni. GitHub. 2023

**Regulamentações**

* Lei Geral de Proteção de Dados (LGPD). Lei nº 13.709/2018
* Marco Civil da Internet. Lei nº 12.965/2014
* Código de Defesa do Consumidor. Lei nº 8.078/1990
* Regulamentação PCI DSS. PCI Security Standards Council

**Artigos e Estudos**

* "Crowdfunding: Geography, Social Networks, and the Timing of Investment Decisions." Agrawal, Catalini, Goldfarb. Journal of Economics & Management Strategy. 2015
* "The Economics of Crowdfunding Platforms." Belleflamme, Lambert, Schwienbacher. Information Economics and Policy. 2014

1. **Apêndices**

**Apêndice A - Diagramas de Arquitetura**

*Referência às imagens já incluídas no documento: Contexto, Contêineres, Casos de Uso*

**Apêndice B - Schemas de Banco de Dados**

*-- Schema principal disponível em: backend/prisma/schema.prisma*

*-- Inclui tabelas: User, Project, Category, Contribution, Comment*

**Apêndice C - Configurações de Deploy**

*# CI/CD Pipeline disponível em: .github/workflows/ci.yml*

*# Inclui: build, test, deploy automatizado*

**Apêndice D - Análise de Cobertura**

* Relatórios SonarQube disponíveis no repositório
* Coverage reports: backend/coverage/ e frontend/coverage/

**7. Avaliações de Professores**

**Considerações Professor/a 1:**

**Assinatura: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Considerações Professor(a):

**Considerações Professor/a 2:**

**Assinatura: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Considerações Professor(a):

**Considerações Professor/a 3:**

**Assinatura: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Considerações Professor(a):