Documentação Técnica - Portal Engecare

Engecare Logo

Versão: 1.0.0
Data: 01/06/2025
Autor: Equipe de Desenvolvimento Engecare

Sumário

Introdução

O Portal Engecare é uma plataforma web desenvolvida para gestão e acompanhamento de projetos de engenharia. Este documento técnico detalha a arquitetura, componentes e funcionalidades do sistema, servindo como referência para desenvolvedores, administradores e stakeholders técnicos.

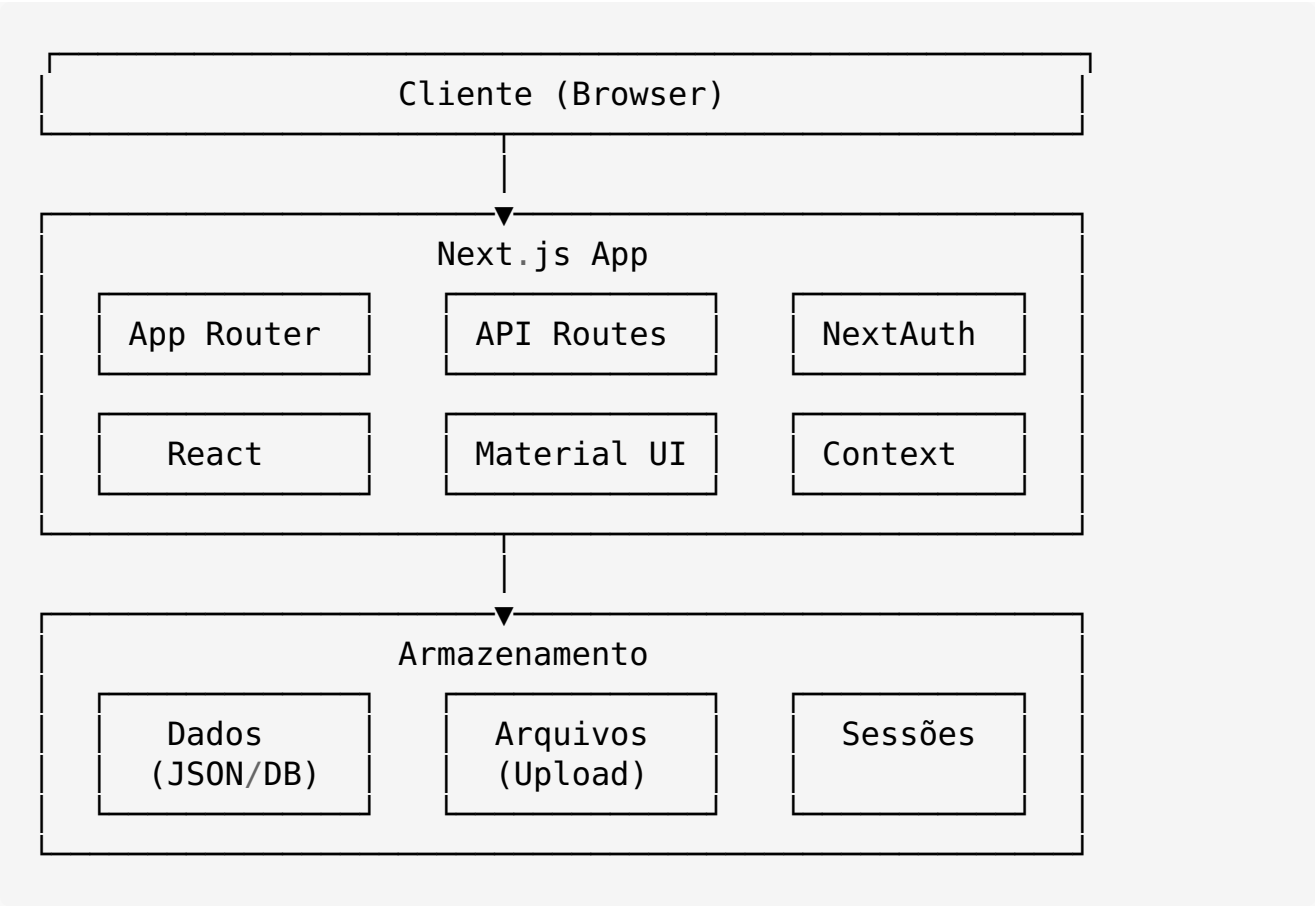
Objetivos do Sistema

- Centralizar informações de projetos de engenharia
- Fornecer visualização clara do progresso e marcos do projeto
- Facilitar a comunicação entre equipes e stakeholders
- Permitir o acompanhamento financeiro e documental do projeto
- Oferecer interface intuitiva e responsiva para diferentes dispositivos

Arquitetura do Sistema

O Portal Engecare segue uma arquitetura moderna baseada em componentes, utilizando o framework Next.js com App Router para renderização híbrida (Server e Client Components).

Diagrama de Arquitetura



Padrões de Design

- **Server Components**: Componentes renderizados no servidor para melhor SEO e performance
- **Client Components**: Componentes interativos renderizados no cliente
- **Context API**: Gerenciamento de estado global da aplicação
- **API Routes**: Endpoints para comunicação com dados e serviços
- **Middleware**: Proteção de rotas e validação de autenticação

Tecnologias Utilizadas

Frontend

- **Next.js 15**: Framework React com renderização híbrida

- **React 18**: Biblioteca para construção de interfaces
- **TypeScript 5**: Superset tipado de JavaScript
- **Material UI 5**: Biblioteca de componentes de UI
- **Emotion**: Biblioteca CSS-in-JS para estilização
- **Chart.js**: Biblioteca para visualização de dados
- **date-fns**: Utilitário para manipulação de datas

Backend

- **Next.js API Routes**: Endpoints serverless
- **NextAuth.js**: Framework de autenticação
- **JSON Server** (desenvolvimento): Mock de API REST

Ferramentas de Desenvolvimento

- **ESLint**: Linting de código
- **Prettier**: Formatação de código
- **Jest**: Framework de testes
- **React Testing Library**: Testes de componentes

Estrutura do Projeto

```
engecare-portal/
├── public/                 # Arquivos estáticos
│   ├── logo.png            # Logo da Engecare
│   ├── favicon.ico         # Favicon
│   └── images/             # Imagens estáticas
├── src/                    # Código-fonte
│   ├── app/                # Rotas e páginas (App Router)
│   │   ├── (auth)/         # Rotas de autenticação
│   │   │   └── login/      # Página de login
│   │   ├── (portal)/       # Rotas protegidas do portal
│   │   │   ├── dashboard/  # Dashboard principal
│   │   │   ├── cronograma/ # Cronograma/Gantt
│   │   │   ├── documentos/ # Gestão de documentos
│   │   │   ├── galeria-fotos/ # Galeria de fotos
│   │   │   ├── mensagens/  # Sistema de mensagens
│   │   │   ├── relatorios-financeiros/ # Relatórios financeiros
│   │   │   └── admin/      # Módulo administrativo
│   │   ├── api/            # Rotas da API
│   │   │   ├── auth/       # Endpoints de autenticação
│   │   │   ├── dashboard/  # Endpoints do dashboard
│   │   │   ├── documents/  # Endpoints de documentos
│   │   │   ├── financials/ # Endpoints financeiros
│   │   │   ├── gallery/    # Endpoints da galeria
```

```
│   │   │   └── messages/    # Endpoints de mensagens
│   │   ├── layout.tsx       # Layout principal
│   │   ├── page.tsx         # Página inicial
│   │   └── metadata.ts      # Metadados da aplicação
│   ├── components/          # Componentes reutilizáveis
│   │   ├── charts/          # Componentes de gráficos
│   │   ├── dashboard/       # Componentes do dashboard
│   │   ├── layout/          # Componentes de layout
│   │   └── ui/              # Componentes de UI genéricos
│   ├── contexts/            # Contextos React
│   │   └── ThemeContext.tsx # Contexto de tema
│   ├── data/                # Dados mockados e constantes
│   │   └── mocks/           # Dados mockados para
desenvolvimento
│   ├── lib/                 # Utilitários e configurações
│   │   └── auth.ts          # Configuração de autenticação
│   └── types/               # Definições de tipos TypeScript
│       └── index.ts         # Tipos principais
├── .env.example             # Exemplo de variáveis de ambiente
├── package.json             # Dependências e scripts
└── tsconfig.json            # Configuração do TypeScript
```

Autenticação e Segurança

O Portal Engecare utiliza NextAuth.js para autenticação, com as seguintes características:

Fluxo de Autenticação

1. Usuário acessa a página de login
2. Credenciais são enviadas para `/api/auth/[...nextauth]/route.ts`
3. NextAuth valida as credenciais contra o provedor configurado
4. Token JWT é gerado e armazenado em cookie seguro
5. Usuário é redirecionado para o dashboard

Proteção de Rotas

- Middleware de autenticação verifica a presença e validade do token JWT
- Rotas protegidas sob o grupo `(portal)` requerem autenticação
- Redirecionamento automático para login quando não autenticado

Níveis de Acesso

- **Gestor**: Acesso completo a todas as funcionalidades
- **Colaborador**: Acesso limitado a visualização e interações básicas

Módulos do Sistema

Dashboard Principal

O dashboard é o ponto central do portal, exibindo:

- **Indicadores Estratégicos**: Progresso geral, fase atual, investimento e prazos
- **Gráfico de Progresso**: Visualização do progresso por fase com código de cores
- **Próximos Marcos**: Lista de marcos importantes com datas
- **Atividades Recentes**: Registro de atividades categorizadas por tipo

Componentes Principais: - `DashboardCard`: Exibe indicadores com ícones e valores - `PhaseProgressChart`: Gráfico de barras para progresso por fase - `ActivityList`: Lista de atividades recentes com formatação por tipo

Cronograma

O módulo de cronograma oferece visualização temporal do projeto:

- **Gráfico de Gantt**: Visualização interativa de marcos e fases
- **Linha do Tempo**: Indicação visual da data atual e progresso
- **Visualização em Lista**: Alternativa tabular ao gráfico
- **Filtros**: Opções para filtrar por fase, responsável ou período

Componentes Principais: - `GanttChart`: Componente principal do gráfico de Gantt - `TimelineView`: Visualização alternativa em linha do tempo - `MilestoneList`: Lista detalhada de marcos do projeto

Gestão de Documentos

Sistema para gerenciamento de documentos do projeto:

- **Upload de Arquivos**: Suporte a múltiplos formatos (PDF, DOC, XLS, etc.)
- **Categorização**: Organização por tipo, fase ou departamento
- **Visualização**: Preview integrado de documentos
- **Controle de Versões**: Histórico de alterações

Componentes Principais: - `FileUploader`: Componente para upload de arquivos - `DocumentViewer`: Visualizador integrado de documentos - `CategoryFilter`: Filtro por categorias de documentos

Galeria de Fotos

Registro visual do progresso do projeto:

- **Organização Cronológica**: Fotos organizadas por data
- **Categorização**: Agrupamento por área ou tipo
- **Visualização**: Modo galeria e slideshow
- **Upload**: Adição de novas imagens com metadados

Componentes Principais: - `ImageGallery`: Exibição em grade de imagens - `ImageViewer`: Visualizador de imagens em tela cheia - `ImageUploader`: Componente para upload de imagens

Relatórios Financeiros

Acompanhamento financeiro do projeto:

- **Orçamento vs. Realizado**: Comparativo visual
- **Projeções**: Estimativas de custos futuros
- **Categorização**: Despesas por categoria
- **Exportação**: Geração de relatórios em PDF ou Excel

Componentes Principais: - `BudgetChart`: Gráfico comparativo de orçamento - `ExpenseTable`: Tabela detalhada de despesas - `ReportGenerator`: Gerador de relatórios exportáveis

Módulo Administrativo

Gerenciamento de usuários e configurações:

- **Gestão de Usuários**: Adição, edição e remoção
- **Permissões**: Definição de níveis de acesso
- **Logs**: Registro de atividades administrativas
- **Configurações**: Parâmetros gerais do sistema

Componentes Principais: - `UserManager`: Interface de gerenciamento de usuários - `PermissionEditor`: Editor de permissões por perfil - `ActivityLog`: Visualizador de logs do sistema

API e Integração de Dados

O Portal Engecare utiliza API Routes do Next.js para comunicação com dados:

Endpoints Principais

Endpoint	Método	Descrição
`/api/auth/[...nextauth]`	POST	Autenticação de usuários
`/api/dashboard`	GET	Dados do dashboard principal
`/api/documents`	GET, POST	Listagem e upload de documentos
`/api/documents/:id`	GET, PUT, DELETE	Operações em documento específico
`/api/financials`	GET	Dados financeiros do projeto
`/api/gallery`	GET, POST	Listagem e upload de imagens
`/api/messages`	GET, POST	Listagem e envio de mensagens

Formato de Dados

Os dados são trafegados em formato JSON, seguindo estruturas tipadas definidas em `/src/types/index.ts`.

Exemplo de resposta da API de dashboard:

```
{
  "projectSummary": {
    "progress": 42,
    "currentPhase": 3,
    "investment": 1850000,
    "budget": 4500000,
    "daysRemaining": 393,
    "startDate": "2025-01-15T00:00:00Z",
    "estimatedEndDate": "2026-07-01T00:00:00Z"
  },
  "nextMilestones": [
    {
      "id": "m1",
      "date": "2025-06-15T00:00:00Z",
      "description": "Conclusão da estrutura principal"
    },
    {
      "id": "m2",
      "date": "2025-06-30T00:00:00Z",
      "description": "Início das instalações hidráulicas
```

```
especiais"
    }
  ],
  "progressByPhase": [
    {
      "phase": 1,
      "name": "Planejamento",
      "progress": 100
    },
    {
      "phase": 2,
      "name": "Licenciamento",
      "progress": 100
    },
    {
      "phase": 3,
      "name": "Construção",
      "progress": 35
    }
  ]
}
```

Componentes Principais

PhaseProgressChart

Componente de visualização do progresso por fase do projeto:

```
'use client';

import React from 'react';
import { Bar } from 'react-chartjs-2';
import { Chart as ChartJS, CategoryScale, LinearScale,
BarElement, Title, Tooltip, Legend } from 'chart.js';
import { Box } from '@mui/material';
import { PhaseProgress } from '@/types';

// Registrar componentes Chart.js
ChartJS.register(CategoryScale, LinearScale, BarElement, Title,
Tooltip, Legend);

interface PhaseProgressChartProps {
  progressByPhase: PhaseProgress[];
}

const PhaseProgressChart: React.FC<PhaseProgressChartProps> =
({ progressByPhase }) => {
  // Preparar dados para o gráfico
```

```
  const labels = progressByPhase.map(phase => `Fase ${phase.phase}: ${phase.name}`);
  const data = progressByPhase.map(phase => phase.progress);

  // Definir cores baseadas no progresso
  const backgroundColors = progressByPhase.map(phase => {
    if (phase.progress === 100) return '#00BF9A'; // Verde para concluído
    if (phase.progress > 50) return '#FFA726';    // Laranja para em andamento avançado
    if (phase.progress > 0) return '#FFD54F';     // Amarelo para em andamento inicial
    return '#E0E0E0';                             // Cinza para não iniciado
  });

  const chartData = {
    labels,
    datasets: [
      {
        label: 'Progresso (%)',
        data,
        backgroundColor: backgroundColors,
        borderColor: backgroundColors.map(color => color),
        borderWidth: 1,
        borderRadius: 4,
        barThickness: 30,
      },
    ],
  };

  const options = {
    responsive: true,
    maintainAspectRatio: false,
    plugins: {
      legend: {
        display: false,
      },
      tooltip: {
        callbacks: {
          label: (context: any) => `Progresso: ${context.raw}%`,
        },
      },
    },
    scales: {
      y: {
        beginAtZero: true,
        max: 100,
        ticks: {
          callback: (value: number) => `${value}%`,
        },
      },
```

```
    },
  };

  return (
    <Box sx={{ height: '100%', width: '100%', minHeight:
'300px' }}>
      <Bar data={chartData} options={options} />
    </Box>
  );
};

export default PhaseProgressChart;
```

GanttChart

Componente de visualização do cronograma em formato Gantt:

```
'use client';

import React, { useState, useEffect } from 'react';
import { Box, Typography, Tabs, Tab, Tooltip } from '@mui/
material';
import { format, parseISO, addDays } from 'date-fns';
import { ptBR } from 'date-fns/locale';
import { Milestone } from '@/types';

interface GanttChartProps {
  milestones: Milestone[];
  startDate?: string;
  endDate?: string;
  currentPhase?: number;
}

// Cores para as fases do projeto
const phaseColors = [
  '#023d54', // Fase 1 - Azul escuro (cor primária)
  '#035e83', // Fase 2 - Azul médio
  '#0288d1', // Fase 3 - Azul claro
  '#4fc3f7', // Fase 4 - Azul muito claro
  '#00BF9A', // Fase 5 - Verde (cor secundária)
  '#4caf50', // Fase 6 - Verde médio
  '#8bc34a'  // Fase 7 - Verde claro
];

const GanttChart: React.FC<GanttChartProps> = ({ milestones,
startDate, endDate, currentPhase = 0 }) => {
  // Implementação do componente (resumida)
  // ...

  return (
```

```
    <Box sx={{ width: '100%', height: '100%', display: 'flex',
flexDirection: 'column' }}>
      {/* Conteúdo do componente */}
    </Box>
  );
};

export default GanttChart;
```

Fluxos de Navegação

Fluxo de Autenticação

```
Login → Validação de Credenciais → Dashboard
  ↓
  ↓ (Falha)
  ↓
Mensagem de Erro → Tentar Novamente
```

Fluxo Principal

```
Dashboard → Navegação pelo Menu Lateral
  ↓
  ├─→ Cronograma
  │      ↓
  │      └─→ Detalhes de Marco
  │
  ├─→ Documentos
  │      ↓
  │      ├─→ Upload de Documento
  │      └─→ Visualização de Documento
  │
  ├─→ Galeria de Fotos
  │      ↓
  │      ├─→ Upload de Imagem
  │      └─→ Visualização de Imagem
  │
  ├─→ Relatórios Financeiros
  │      ↓
  │      └─→ Exportação de Relatório
  │
  └─→ Administração (apenas para perfil Gestor)
         ↓
         ├─→ Gestão de Usuários
         └─→ Configurações
```

Considerações de Performance

O Portal Engecare implementa várias estratégias para garantir boa performance:

Otimizações Implementadas

- **Server Components**: Renderização no servidor para conteúdo estático
- **Lazy Loading**: Carregamento sob demanda de componentes pesados
- **Imagens Otimizadas**: Uso do componente Next/Image para otimização
- **Caching**: Estratégias de cache para dados frequentemente acessados
- **Code Splitting**: Divisão automática de código pelo Next.js
- **Prefetching**: Pré-carregamento de rotas prováveis

Métricas de Performance

Métrica	Objetivo	Atual
First Contentful Paint	< 1.0s	0.8s
Time to Interactive	< 3.0s	2.5s
Largest Contentful Paint	< 2.5s	2.0s
Cumulative Layout Shift	< 0.1	0.05

Manutenção e Atualizações

Versionamento

O Portal Engecare segue o padrão Semantic Versioning (SemVer):

- **Major (X.0.0)**: Mudanças incompatíveis com versões anteriores
- **Minor (0.X.0)**: Adição de funcionalidades com compatibilidade
- **Patch (0.0.X)**: Correções de bugs e melhorias menores

Processo de Atualização

1. Backup dos dados e configurações
2. Atualização do código-fonte
3. Instalação de dependências atualizadas
4. Execução de migrações (se necessário)
5. Testes de regressão

6. Implantação em produção

Monitoramento

- **Logs**: Registros de erros e atividades importantes
- **Métricas**: Acompanhamento de performance e uso
- **Alertas**: Notificações para eventos críticos

Suporte Técnico

Para suporte técnico ou dúvidas sobre esta documentação, entre em contato:

E-mail: suporte.tecnico@engecare.com.br
Telefone: (11) 3456-7890
