📋 Sistema Elegante de Gerenciamento de Chamados

🎯 Visão Geral

Um sistema moderno e elegante para gerenciamento de chamados (tickets) com interface intuitiva, dashboard visual, autenticação OAuth, sistema de assinatura com Stripe e suporte a múltiplos usuários.

Tecnologias Principais:

- **Frontend:** React 19, Tailwind CSS 4, TypeScript, Framer Motion
- **Backend:** Express.js, tRPC, Node.js
- **Banco de Dados:** MySQL/TiDB com Drizzle ORM
- **Autenticação:** Manus OAuth
- **Pagamentos:** Stripe
- **Testes:** Vitest

📁 Estrutura de Diretórios

Plain Text

```
sistema-chamados-moderno/
├── client/                          # Frontend React
│   ├── public/                      # Arquivos estáticos
│   ├── src/
│   │   ├── pages/                   # Páginas da aplicação
│   │   │   ├── Home.tsx             # Página inicial
│   │   │   ├── Dashboard.tsx        # Dashboard com gráficos
│   │   │   ├── TicketsList.tsx      # Listagem de chamados
│   │   │   ├── TicketDetail.tsx     # Detalhes do chamado
│   │   │   ├── CreateTicket.tsx     # Criar novo chamado
│   │   │   ├── Pricing.tsx          # Página de planos
│   │   │   ├── Billing.tsx          # Gerenciamento de faturamento
│   │   │   └── NotFound.tsx         # Página 404
│   │   ├── components/              # Componentes reutilizáveis
│   │   ├── contexts/                # React Contexts
│   │   ├── hooks/                   # Custom hooks
│   │   ├── lib/
│   │   │   └── trpc.ts              # Cliente tRPC
│   │   ├── App.tsx                  # Componente raiz
```

```
│   │   ├── main.tsx                # Entrada da aplicação
│   │   └── index.css               # Estilos globais
│   └── index.html                  # HTML template
│
├── server/                         # Backend Node.js
│   ├── _core/                      # Framework core (não editar)
│   ├── routers.ts                  # Rotas tRPC
│   ├── routers/
│   │   └── subscription.ts         # Rotas de assinatura
│   ├── db.ts                       # Funções de banco de dados
│   ├── subscription-db.ts          # Funções de assinatura
│   ├── products.ts                 # Configuração de planos
│   └── *.test.ts                   # Testes unitários
│
├── drizzle/                        # ORM e Migrations
│   ├── schema.ts                   # Schema do banco de dados
│   └── migrations/                 # Histórico de migrações
│
├── shared/                         # Código compartilhado
│   └── const.ts                    # Constantes
│
├── storage/                        # Integração S3
│   └── index.ts                    # Helpers de storage
│
├── package.json                    # Dependências e scripts
├── tsconfig.json                   # Configuração TypeScript
├── vite.config.ts                  # Configuração Vite
├── tailwind.config.ts              # Configuração Tailwind
├── drizzle.config.ts               # Configuração Drizzle
└── README.md                       # Este arquivo
```

🚀 Instalação e Setup

Pré-requisitos

- Node.js 18+
- pnpm (gerenciador de pacotes)
- Banco de dados MySQL/TiDB

Passos de Instalação

1. **Clonar o repositório**

```
Bash
```

```
git clone <seu-repositorio>
cd sistema-chamados-moderno
```

1. **Instalar dependências**

```
Bash

pnpm install
```

1. **Configurar variáveis de ambiente**

   Crie um arquivo `.env` na raiz do projeto:

```
Plain Text

DATABASE_URL=mysql://usuario:senha@localhost:3306/chamados
JWT_SECRET=sua-chave-secreta-aqui
VITE_APP_ID=seu-app-id
OAUTH_SERVER_URL=https://api.manus.im
VITE_OAUTH_PORTAL_URL=https://portal.manus.im
STRIPE_SECRET_KEY=sk_test_...
VITE_STRIPE_PUBLISHABLE_KEY=pk_test_...
STRIPE_WEBHOOK_SECRET=whsec_...
```

1. **Executar migrações do banco de dados**

```
Bash

pnpm db:push
```

1. **Iniciar servidor de desenvolvimento**

```
Bash

pnpm dev
```

O servidor estará disponível em `http://localhost:3000`

📊 Funcionalidades Principais

1. Dashboard Visual

- Gráficos de status de chamados (novo, em andamento, resolvido, fechado)

- Indicadores de prioridade (baixa, média, alta, urgente)
- Estatísticas em tempo real
- Interface elegante com animações suaves

2. Gerenciamento de Chamados

- Criar novo chamado com título, descrição, prioridade e categoria
- Listar chamados com filtros por status e prioridade
- Visualizar detalhes completo de cada chamado
- Atualizar status do chamado
- Histórico de alterações

3. Sistema de Assinatura

- 3 planos profissionais (Básico, Profissional, Enterprise)
- Preços mensais e anuais
- Integração com Stripe para pagamentos
- Painel de faturamento
- Histórico de pagamentos

4. Autenticação e Segurança

- Login via Manus OAuth
- Autenticação segura com JWT
- Controle de acesso por função (user/admin)
- Proteção de rotas

5. Interface Responsiva

- Design mobile-first
- Compatível com todos os navegadores modernos
- Tema claro/escuro
- Animações suaves com Framer Motion

🔧 Configuração de Banco de Dados

Schema Principal

Tabela: `users`

SQL

```
- id: INT (PK)
- openId: VARCHAR (Unique)
- name: TEXT
- email: VARCHAR
- loginMethod: VARCHAR
- role: ENUM (user, admin)
- createdAt: TIMESTAMP
- updatedAt: TIMESTAMP
- lastSignedIn: TIMESTAMP
```

Tabela: `tickets`

SQL

```
- id: INT (PK)
- title: VARCHAR
- description: TEXT
- priority: ENUM (baixa, média, alta, urgente)
- status: ENUM (novo, em_andamento, resolvido, fechado)
- categoryId: INT (FK)
- createdById: INT (FK)
- assignedToId: INT (FK, nullable)
- createdAt: TIMESTAMP
- updatedAt: TIMESTAMP
```

Tabela: `ticket_categories`

SQL

```
- id: INT (PK)
- name: VARCHAR
- color: VARCHAR
- createdAt: TIMESTAMP
```

Tabela: `ticket_history`

SQL

```
- id: INT (PK)
- ticketId: INT (FK)
- changedById: INT (FK)
- fieldChanged: VARCHAR
- oldValue: TEXT
- newValue: TEXT
- createdAt: TIMESTAMP
```

Tabela: `subscriptions`

SQL

```
- id: INT (PK)
- userId: INT (FK)
- planId: INT (FK)
- stripeSubscriptionId: VARCHAR
- status: ENUM (active, canceled, expired)
- currentPeriodStart: TIMESTAMP
- currentPeriodEnd: TIMESTAMP
- createdAt: TIMESTAMP
- updatedAt: TIMESTAMP
```

🔌 Rotas tRPC

Autenticação

TypeScript

```
trpc.auth.me.useQuery()              // Obter usuário atual
trpc.auth.logout.useMutation()       // Fazer logout
```

Tickets

TypeScript

```
trpc.tickets.list.useQuery()         // Listar todos os chamados
trpc.tickets.getById.useQuery({id})  // Obter chamado por ID
trpc.tickets.create.useMutation()    // Criar novo chamado
trpc.tickets.updateStatus.useMutation() // Atualizar status
trpc.tickets.getHistory.useQuery()   // Obter histórico
```

Categorias

```typescript
trpc.categories.list.useQuery()      // Listar categorias
```

Assinatura

```typescript
trpc.subscription.getPlans.useQuery()        // Obter planos
trpc.subscription.checkout.useMutation()     // Criar sessão checkout
trpc.subscription.getCurrentPlan.useQuery()  // Plano atual do usuário
```

🎨 Paleta de Cores

A aplicação usa Tailwind CSS 4 com tema elegante:

Elemento	Cor	Código
Primary	Roxo Vibrante	`#a855f7`
Secondary	Rosa	`#ec4899`
Success	Verde	`#10b981`
Warning	Amarelo	`#f59e0b`
Error	Vermelho	`#ef4444`
Background	Cinza Claro	`#f8fafc`
Text	Cinza Escuro	`#1e293b`

📝 Scripts Disponíveis

```bash
# Desenvolvimento
pnpm dev                # Iniciar servidor de desenvolvimento
```

```
# Build
pnpm build              # Build para produção

# Testes
pnpm test               # Executar testes com Vitest

# Banco de Dados
pnpm db:push            # Executar migrações

# Formatação
pnpm format             # Formatar código com Prettier

# Type Check
pnpm check              # Verificar tipos TypeScript
```

🧪 Testes

Os testes são escritos com Vitest e estão localizados em `server/*.test.ts`:

Bash

```
# Executar todos os testes
pnpm test

# Executar testes em modo watch
pnpm test --watch

# Executar teste específico
pnpm test auth.logout
```

Exemplo de teste:

TypeScript

```
import { describe, expect, it } from "vitest";
import { appRouter } from "./routers";

describe("auth.logout", () => {
  it("clears the session cookie", async () => {
    const caller = appRouter.createCaller(ctx);
    const result = await caller.auth.logout();
    expect(result).toEqual({ success: true });
  });
});
```

💳 Integração Stripe

Configuração

1. Crie uma conta em stripe.com
2. Obtenha suas chaves de teste/produção
3. Configure as variáveis de ambiente:
   - `STRIPE_SECRET_KEY`
   - `VITE_STRIPE_PUBLISHABLE_KEY`
   - `STRIPE_WEBHOOK_SECRET`

Planos Configurados

Plano	Preço Mensal	Preço Anual	Chamados	Usuários
Básico	R$ 49,00	R$ 499,00	100	3
Profissional	R$ 149,00	R$ 1.499,00	1.000	10
Enterprise	R$ 499,00	R$ 4.999,00	Ilimitado	Ilimitado

Teste de Pagamento

Use o cartão de teste: `4242 4242 4242 4242`

🔐 Segurança

- **Autenticação:** Manus OAuth com JWT
- **HTTPS:** Todas as conexões são criptografadas
- **CORS:** Configurado para aceitar requisições do frontend
- **Rate Limiting:** Implementado em rotas críticas
- **Validação:** Todas as entradas são validadas com Zod

📱 Responsividade

A aplicação é totalmente responsiva:

- **Mobile:** < 640px
- **Tablet:** 640px - 1024px
- **Desktop:** > 1024px

🚀 Deploy

Opções de Deploy

1. **Manus (Recomendado)**
   - Hosting gerenciado
   - Domínio customizável
   - Suporte a banco de dados
   - Deploy automático
2. **Vercel**

   ```
   Bash

   vercel deploy
   ```

3. **Railway**
   - Conecte seu repositório GitHub
   - Configure variáveis de ambiente
   - Deploy automático
4. **Docker**

   ```
   Bash

   docker build -t sistema-chamados .
   docker run -p 3000:3000 sistema-chamados
   ```

🐛 Troubleshooting

Erro: "Cannot find module"

```
Bash
```

```
pnpm install
pnpm db:push
```

Erro: "Database connection failed"

Verifique se `DATABASE_URL` está correto e o banco está acessível.

Erro: "OAuth not configured"

Configure `VITE_APP_ID` e `OAUTH_SERVER_URL` nas variáveis de ambiente.

Erro: "Stripe key invalid"

Verifique se as chaves Stripe estão corretas no arquivo `.env`.

📚 Recursos Adicionais

- Documentação React
- Documentação Tailwind CSS
- Documentação tRPC
- Documentação Drizzle ORM
- Documentação Stripe

📞 Suporte

Para dúvidas ou problemas, entre em contato com o time de desenvolvimento.

📄 Licença

Este projeto está sob licença MIT.

Versão: 1.0.0

Última atualização: Janeiro 2026