

# Guia Completo de Arquivos do Projeto CS2 Skin Monitor

## Estrutura Geral

```
cs2_skin_monitor_github/
├── app/                # Next.js App Router
├── components/         # Componentes React
├── daemon/             # Scripts de monitoramento
├── hooks/              # Custom React Hooks
├── lib/                # Bibliotecas e utilidades
├── prisma/             # Schema do banco de dados
├── public/             # Arquivos estáticos
└── scripts/           # Scripts auxiliares
```

## Arquivos de Configuração Raiz

Arquivo	Descrição
package.json	Dependências e scripts do projeto
tsconfig.json	Configuração do TypeScript
next.config.js	Configuração do Next.js
tailwind.config.ts	Configuração do Tailwind CSS
postcss.config.js	Configuração do PostCSS
components.json	Configuração do Shadcn/ui
.gitignore	Arquivos ignorados pelo Git
.env.example	Exemplo de variáveis de ambiente
README.md	Documentação do projeto

## Frontend (app/)

### app/layout.tsx

- Layout principal da aplicação
- Configuração de fontes e metadados
- Wrapper de provedores (tema, toast)

## app/page.tsx

- Página inicial
- Importa o componente principal

## app/globals.css

- Estilos globais
  - Variáveis CSS para temas
  - Configuração do Tailwind
- 



## Backend - API Routes (app/api/)

---

### app/api/skins/route.ts

- GET - Lista todas as skins
- POST - Adiciona nova skin
- Scraping automático de dados da Steam

### app/api/skins/[id]/route.ts

- DELETE - Remove uma skin
- PATCH - Atualiza uma skin (ativar/desativar)

### app/api/config/route.ts

- GET - Busca configurações do usuário
- POST - Atualiza configurações (email)

### app/api/monitor/route.ts

- POST - Verifica preços de todas as skins ativas
- Chamado pelo cron job a cada 15 minutos
- Envia alertas por email quando necessário

### app/api/alerts/route.ts

- GET - Lista histórico de alertas disparados
- 



## Componentes React (components/)

---

### components/skins-monitor-app.tsx

- ★ **COMPONENTE PRINCIPAL**
- Interface completa do monitor
- Gerenciamento de estado
- Tabs: Skins Monitoradas, Configurações, Histórico

### components/theme-provider.tsx

- Provedor de tema claro/escuro
- Usa next-themes

## components/ui/ (52 arquivos)

- Componentes Shadcn/ui
- Botões, inputs, cards, dialogs, etc.
- Totalmente personalizáveis

### Principais componentes UI:

- `button.tsx` - Botões
- `input.tsx` - Campos de entrada
- `card.tsx` - Cards
- `dialog.tsx` - Modais
- `tabs.tsx` - Abas
- `switch.tsx` - Toggle
- `badge.tsx` - Tags
- `toast.tsx` / `toaster.tsx` - Notificações
- `table.tsx` - Tabelas



## Bibliotecas e Utilidades (lib/)

### lib/db.ts

```
// Cliente Prisma singleton
export const prisma = new PrismaClient()
```

### lib/email.ts

- `enviarEmailAlerta()` - Envia email quando preço atinge alvo
- Configuração do Nodemailer
- Templates de email em HTML

### lib/price-scraper.ts

- `extrairDadosDaSkin()` - Scraping da página da Steam
- Extrai: nome, preço, imagem
- Conversão de moeda USD para BRL

### lib/types.ts

- Tipos TypeScript do projeto
- Interfaces: Skin, Alerta, Configuracao

### lib/utils.ts

- Função `cn()` - Merge de classes CSS
  - Utilitários gerais
-

## Banco de Dados (prisma/)

### prisma/schema.prisma

Define 3 modelos:

#### Model Skin

```
model Skin {
  id          String
  nome        String
  link        String (unique)
  precoAlvo   Float
  precoAtual  Float?
  status      String (ativo/inativo)
  imageUrl    String?
  createdAt   DateTime
  updatedAt   DateTime
  alertas     Alerta[]
}
```

#### Model Alerta

```
model Alerta {
  id          String
  skinId       String
  precoAtingido Float
  dataAlerta   DateTime
  skin         Skin (relação)
}
```

#### Model Configuracao

```
model Configuracao {
  id          String
  email       String?
  ultimaVerificacao DateTime?
  createdAt   DateTime
  updatedAt   DateTime
}
```

## Daemon (daemon/)

### daemon/monitor-daemon.ts

- Script de monitoramento (referência)
- Pode ser usado como processo separado
- Na produção, usa-se o endpoint `/api/monitor`

## Hooks (hooks/)

---

### hooks/use-toast.ts

- Hook customizado para notificações toast
  - Gerenciamento de estado de toasts
- 

## Scripts (scripts/)

---

### scripts/seed.ts

- Popula banco de dados com dados de exemplo
  - Útil para desenvolvimento/testes
  - Executar: `yarn prisma db seed`
- 

## Arquivos Estáticos (public/)

---

### public/favicon.svg

- Ícone do site

### public/og-image.png

- Imagem para compartilhamento social (Open Graph)
- 

## Variáveis de Ambiente (.env)

---

```
# Banco de Dados
DATABASE_URL="postgresql://..."

# Email SMTP
SMTP_HOST="smtp.gmail.com"
SMTP_PORT="587"
SMTP_USER="seu-email@gmail.com"
SMTP_PASS="senha-de-app"
SMTP_FROM="seu-email@gmail.com"

# Aplicação
NEXT_PUBLIC_APP_URL="http://localhost:3000"
```

---

## Fluxo de Funcionamento

---





### 1 Adicionar Skin

Usuário cola link →  
API POST /api/skins →  
price-scraper.ts extrai dados →  
Salva no banco (Prisma) →  
Retorna para frontend

### 2 Monitoramento Automático

Cron job chama /api/monitor →  
Busca todas skins ativas →  
Para cada skin:  
- Scraping do preço atual  
- Compara com preço alvo  
- Se atingiu: envia email + salva alerta  
→ Atualiza banco de dados

### 3 Visualização

Frontend carrega dados   
GET /api/skins (lista skins)   
GET /api/alerts (histórico)   
GET /api/config (configurações)   
Renderiza na interface

---

## Dependências Principais

---

### Produção

- next - Framework React
- react / react-dom - Biblioteca React
- @prisma/client - ORM banco de dados
- cheerio - Web scraping
- nodemailer - Envio de emails
- @radix-ui/\* - Componentes UI base
- lucide-react - Ícones
- tailwindcss - CSS utility-first

### Desenvolvimento

- typescript - Linguagem
  - prisma - CLI do Prisma
  - eslint - Linter
  - @types/\* - Tipos TypeScript
-

## Arquivos Mais Importantes

---

### Para entender a aplicação:

1. `components/skins-monitor-app.tsx` - Interface principal
2. `lib/price-scraper.ts` - Lógica de scraping
3. `app/api/monitor/route.ts` - Monitoramento automático
4. `prisma/schema.prisma` - Estrutura do banco

### Para configurar:

1. `.env` - Variáveis de ambiente
2. `package.json` - Dependências
3. `README.md` - Instruções

### Para personalizar:

1. `app/globals.css` - Estilos globais
  2. `tailwind.config.ts` - Tema
  3. `components/ui/*` - Componentes visuais
- 



## Interface do Usuário

---

### Tab 1: Skins Monitoradas

- Lista de skins com cards
- Botão “Adicionar Nova Skin”
- Status e preços em tempo real
- Botão “Verificar Agora”

### Tab 2: Configurações

- Campo de email
- Toggle liga/desliga monitoramento
- Informações sobre funcionamento

### Tab 3: Histórico de Alertas

- Tabela com todos os alertas disparados
  - Data, skin, preço atingido
-

## Comandos Úteis

```
# Desenvolvimento
yarn dev                # Servidor de desenvolvimento
yarn prisma studio      # Interface visual do banco

# Banco de Dados
yarn prisma migrate dev  # Criar migração
yarn prisma generate     # Gerar cliente Prisma
yarn prisma db seed      # Popular com dados de teste
yarn prisma db push      # Sincronizar schema

# Produção
yarn build               # Build otimizado
yarn start               # Servidor de produção

# Manutenção
yarn lint                # Verificar código
```

## Tecnologias e Conceitos

- **Next.js 14 App Router** - Roteamento e SSR
- **Server Actions** - Ações do servidor
- **API Routes** - Endpoints REST
- **Prisma ORM** - Abstração de banco de dados
- **Web Scraping** - Extração de dados HTML
- **SMTP** - Protocolo de email
- **Cron Jobs** - Tarefas agendadas
- **TypeScript** - Tipagem estática
- **Tailwind CSS** - Estilização utility-first
- **React Hooks** - Gerenciamento de estado
- **Radix UI** - Componentes acessíveis

✓ **Todos os arquivos estão prontos para GitHub!**

 Baixe o arquivo compactado: `cs2_skin_monitor_github.tar.gz`