🎬 StreamAI Studio v2.0

Plataforma Profissional de Broadcasting Multi-Canal com IA

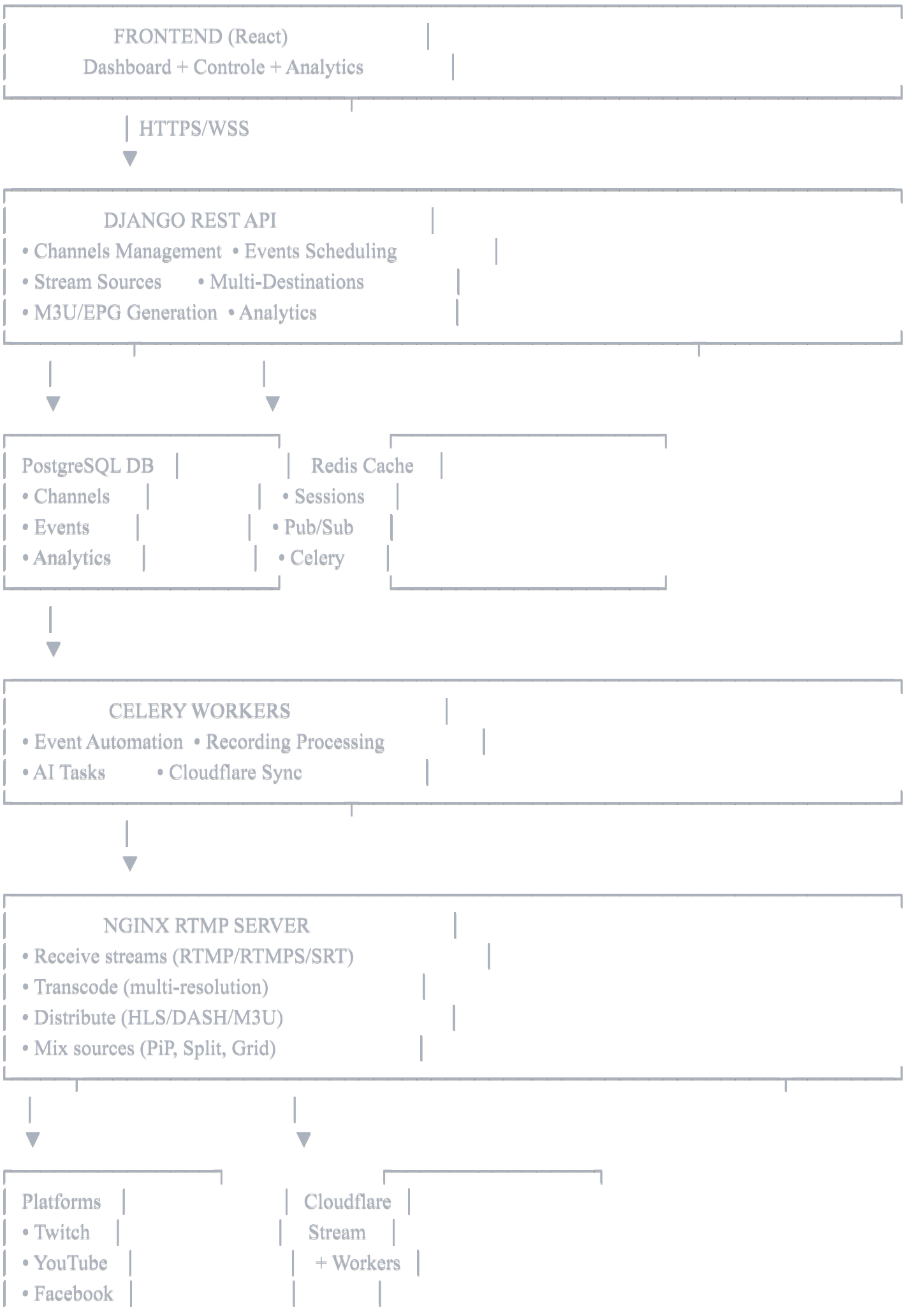
Mostrar Imagem Mostrar Imagem Mostrar Imagem Mostrar Imagem

🌟 Visão Geral

StreamAI Studio é uma **plataforma completa de broadcasting** que permite:

✨ **Gerenciamento de Canais** - Crie e gerencie múltiplos canais de transmissão
🔄 **Multi-Streaming** - Transmita simultaneamente para Twitch, YouTube, Facebook, Kick
🤖 **IA Integrada** - Análise de código, chat inteligente, automação completa
📺 **EPG & M3U** - Sistema completo de programação (IPTV-ready)
🎚️ **Mixer de Streams** - Combine múltiplas fontes em tempo real
☁️ **Cloudflare** - Distribuição global via CDN
📊 **Analytics** - Métricas detalhadas em tempo real
🔐 **Multi-Tenancy** - Suporte para múltiplas organizações

🧬 Arquitetura


FRONTEND (React)
Dashboard + Controle + Analytics
HTTPS/WSS
DJANGO REST API
• Channels Management • Events Scheduling
• Stream Sources • Multi-Destinations
• M3U/EPG Generation • Analytics
PostgreSQL DB
• Channels
• Events
• Analytics
Redis Cache
• Sessions
• Pub/Sub
• Celery
CELERY WORKERS
• Event Automation • Recording Processing
• AI Tasks • Cloudflare Sync
NGINX RTMP SERVER
• Receive streams (RTMP/RTMPS/SRT)
• Transcode (multi-resolution)
• Distribute (HLS/DASH/M3U)
• Mix sources (PiP, Split, Grid)
Platforms
• Twitch
• YouTube
• Facebook
Cloudflare
Stream
+ Workers

• Kick		Global CDN	

🎯 Casos de Uso

1. Streaming Educativo de Tecnologia

- Transmita suas lives de código
- IA analisa código em tempo real
- Overlays com métricas e explicações
- Documentação automática

2. Broadcasting Multi-Plataforma

- Configure uma vez, transmita para todos
- Gerenciar múltiplas plataformas de um lugar
- Adaptive bitrate automático
- Redundância e fallback

3. IPTV Profissional

- EPG (Electronic Program Guide)
- Playlists M3U8
- Programação de eventos
- VOD integrado

4. Produção Multi-Câmera

- Mixer de fontes ao vivo
- Picture-in-Picture
- Split screen
- Grid layouts

🚀 Quick Start

Pré-requisitos

bash

```
# Docker & Docker Compose
curl -fsSL https://get.docker.com | sh
```

Instalação (5 minutos)

bash

```
# 1. Clone o repositório
git clone https://github.com/seu-usuario/streamai-studio.git
cd streamai-studio

# 2. Configure variáveis de ambiente
cp .env.example .env
nano .env  # Edite com suas credenciais

# 3. Inicie tudo
docker-compose up -d

# 4. Aguarde inicialização (1-2 minutos)
docker-compose logs -f django-api

# 5. Crie superuser
docker-compose exec django-api python manage.py createsuperuser

# 6. Acesse!
# Frontend: http://localhost:3000
# API: http://localhost:8000/admin
# RTMP Stats: http://localhost:8080/stat
```

📦 Componentes Principais

1. Django Backend

- ✅ REST API completa
- ✅ Sistema de autenticação (JWT)
- ✅ Multi-tenancy
- ✅ Gerenciamento de canais
- ✅ Eventos programados
- ✅ Analytics em tempo real

2. NGINX RTMP

- ✅ Servidor RTMP/RTMPS/SRT
- ✅ Transcodificação (7 qualidades)
- ✅ HLS/DASH adaptativo
- ✅ Mixer de fontes
- ✅ Gravação automática

3. Celery Workers

- ✅ Automação de eventos

- ✅ Processamento de gravações
- ✅ Upload para Cloudflare
- ✅ Análise com IA (Claude)
- ✅ Sync de analytics

4. React Dashboard

- ✅ Controle de canais
- ✅ Programação de eventos
- ✅ Analytics em tempo real
- ✅ Mixer de streams
- ✅ WebSocket updates

5. Cloudflare Integration

- ✅ Live Inputs
- ✅ VOD com CDN global
- ✅ Workers para lógica edge
- ✅ Analytics integrado

📖 Documentação Completa

APIs REST

Canais:

bash

```
GET    /api/channels/                  # Listar canais
POST   /api/channels/                  # Criar canal
GET    /api/channels/{id}/             # Detalhes
POST   /api/channels/{id}/start_stream/  # Iniciar
POST   /api/channels/{id}/stop_stream/   # Parar
GET    /api/channels/{id}/stats/       # Estatísticas
```

Eventos:

bash

```
GET    /api/events/                    # Listar eventos
POST   /api/events/                    # Criar evento
GET    /api/events/schedule/           # EPG
POST   /api/events/{id}/start_now/     # Iniciar agora
```

Playlists:

bash

```
GET  /api/playlists/channel/{slug}.m3u8    # M3U de canal
GET  /api/playlists/all_channels.m3u8      # Todos canais
GET  /api/playlists/my_channels.m3u8       # Meus canais
```

EPG:

bash

```
GET  /api/epg/xmltv.xml            # EPG em XMLTV
GET  /api/epg/guide.json           # EPG em JSON
```

Mixer:

bash

```
POST  /api/mixer/create            # Criar stream mixado
POST  /api/mixer/stop              # Parar mixer
```

Configurar OBS

Configurações de Stream:

```
Server: rtmp://seu-servidor:1935/live
Key: {seu_canal_slug}
```

Para RTMPS (seguro):

```
Server: rtmps://seu-servidor:443/live_secure
Key: {seu_canal_slug}
```

Gerar M3U para IPTV

bash

```
# Via navegador
https://seu-servidor/api/playlists/all_channels.m3u8

# Via curl
curl -O https://seu-servidor/api/playlists/all_channels.m3u8

# Usar em VLC, Kodi, Perfect Player, etc
```

Integrações

Cloudflare Stream

python

```
# Criar Live Input
POST /api/channels/{id}/create_cloudflare_input/

# Resposta:
{
  "rtmps_url": "rtmps://live.cloudflare.com:443/live/...",
  "stream_key": "..."
}
```

Plataformas de Streaming

python

```
# Adicionar destino
POST /api/destinations/
{
  "channel": "uuid",
  "platform": "twitch",
  "rtmp_url": "rtmp://live.twitch.tv/app/",
  "stream_key": "your_key"
}
```

IA (Claude)


python

```
# Análise automática de código
# Configurada via Celery tasks
# Roda a cada X minutos em canais ativos
```

📊 Analytics & Métricas

Em Tempo Real


javascript

```
// WebSocket
ws://seu-servidor/ws/streaming

// Recebe:
{
  "type": "viewer_update",
  "channel_id": "...",
  "current_viewers": 42,
  "peak_viewers": 105
}
```

Histórico


bash

```
GET /api/analytics/overview/
{
  "total_channels": 5,
  "live_channels": 2,
  "current_viewers": 342,
  "last_7_days": {
    "total_views": 15420,
    "total_bandwidth_gb": 245.3
  }
}
```

🛠️ Desenvolvimento

Estrutura do Projeto

```
streamai-studio/
├── backend/                # Django API
│   ├── streaming/          # App principal
│   │   ├── models.py       # Modelos do banco
│   │   ├── api_views.py    # ViewSets REST
│   │   ├── advanced_views.py # M3U, EPG, Mixer
│   │   ├── tasks.py        # Celery tasks
│   │   └── urls.py         # Rotas
│   ├── streamai/           # Configuração
│   └── requirements.txt    # Dependências Python
├── frontend/               # React Dashboard
│   ├── src/
│   └── package.json
├── nginx/                  # NGINX configs
│   ├── nginx.conf          # Config RTMP
│   └── proxy.conf          # Reverse proxy
├── monitoring/             # Prometheus & Grafana
├── docker-compose.yml      # Orquestração
└── .env                    # Variáveis de ambiente
```

Rodar Localmente

bash

```
# Backend
cd backend
python manage.py runserver

# Frontend
cd frontend
npm start

# Celery
celery -A streamai worker -l info
celery -A streamai beat -l info
```

🔐 Segurança

Checklist

- ☑ JWT para autenticação
- ☑ HTTPS/TLS em produção
- ☑ RTMPS para streams seguros
- ☑ Secrets em variáveis de ambiente
- ☑ Rate limiting
- ☑ CORS configurado
- ☑ Sanitização de inputs
- ☑ SQL injection protection (ORM)

Recomendações

1. **Mudar senhas padrão**
2. **Usar HTTPS** (Let's Encrypt)
3. **Firewall** (UFW/iptables)
4. **Backups regulares**
5. **Monitorar logs**

📈 Escalabilidade

Horizontal Scaling

yaml

```
# docker-compose.yml
django-api:
  deploy:
    replicas: 3

celery-worker:
  deploy:
    replicas: 5
```

Load Balancer


nginx

```
upstream django {
    server django-api-1:8000;
    server django-api-2:8000;
    server django-api-3:8000;
}
```

CDN

- ✅ Cloudflare para conteúdo estático
- ✅ CloudFront como alternativa
- ✅ HLS/DASH para adaptive bitrate

🤝 Contribuindo

Contribuições são muito bem-vindas!

1. Fork o projeto
2. Crie sua feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit suas mudanças (`git commit -m 'Add AmazingFeature'`)
4. Push para a branch (`git push origin feature/AmazingFeature`)
5. Abra um Pull Request

Áreas que Precisam de Ajuda

- ☐ Testes automatizados (pytest, jest)
- ☐ Documentação API (Swagger/OpenAPI)
- ☐ Suporte para mais plataformas
- ☐ Novos layouts de mixer
- ☐ Internacionalização (i18n)
- ☐ Mobile app (React Native)

📄 Licença

Este projeto está sob a licença MIT. Veja LICENSE para mais detalhes.

🙏 Agradecimentos

- **Anthropic** - Claude API para IA
- **OBS Project** - Software de streaming
- **NGINX** - Módulo RTMP
- **Cloudflare** - Stream & Workers
- **Django** - Framework web
- **React** - UI library
- Comunidade open source ❤️

📞 Contato & Suporte

- 🌐 Website: https://streamai.studio
- 📧 Email: suporte@streamai.studio
- 💬 Discord: https://discord.gg/streamai
- 🐛 Issues: https://github.com/seu-usuario/streamai-studio/issues
- 📖 Docs: https://docs.streamai.studio
- 🐦 Twitter: @streamai_studio

🎯 Roadmap

v2.1 (Q2 2025)

- [ ] Mobile app (iOS/Android)
- [ ] WebRTC para latência ultra-baixa
- [ ] AI voice cloning para narração
- [ ] Geração automática de highlights

v2.2 (Q3 2025)

- [ ] Live commerce integration
- [ ] NFT/Web3 features
- [ ] Advanced analytics ML
- [ ] Multi-language support

v3.0 (Q4 2025)

- [ ] Kubernetes deployment
- [ ] Microservices architecture
- [ ] GraphQL API
- [ ] Real-time collaboration

📊 Stats

Mostrar Imagem | Mostrar Imagem | Mostrar Imagem | Mostrar Imagem

---

Feito com ❤️ por desenvolvedores, para desenvolvedores

StreamAI Studio - O futuro do broadcasting é aqui 🚀