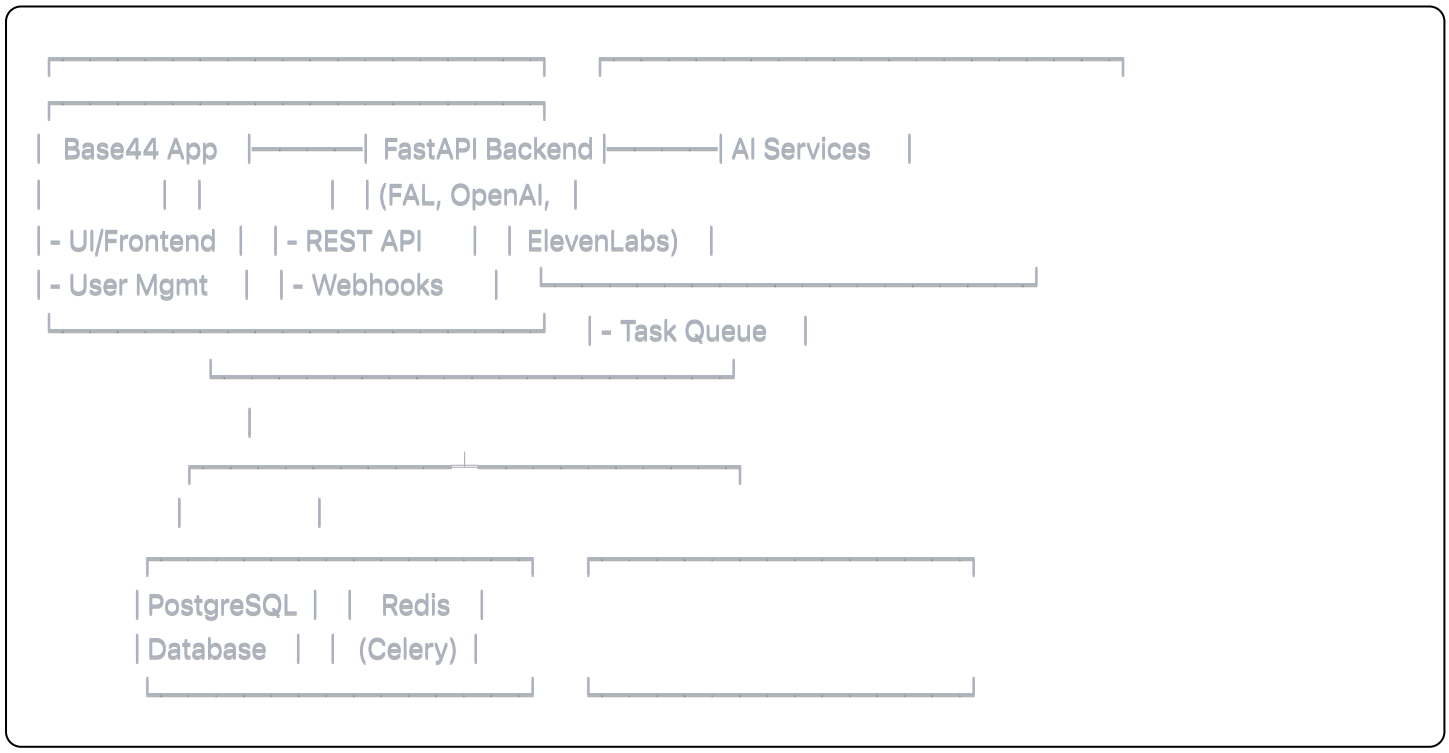
Video AI Backend

A complete Python + FastAPI backend that replaces the n8n video generation workflows. This system converts your n8n "Main Video Ad" and "Revision" workflows into a scalable, maintainable Python backend.

🚀 Features

- **Complete n8n Workflow Conversion**: Converted both Main Video Ad and Revision workflows to Python
- **Async Processing**: Uses Celery for background video generation tasks
- **AI Service Integration**: FAL AI, ElevenLabs, OpenAI, and Lyria2 clients
- **Database Management**: PostgreSQL with SQLAlchemy ORM
- **Real-time Status**: Track video generation progress
- **Base44 Integration**: Seamless callbacks to your existing Base44 app
- **Docker Support**: Complete containerized deployment
- **API Documentation**: Auto-generated FastAPI docs
- **Monitoring**: Celery Flower for task monitoring
- **Error Handling**: Comprehensive error handling and logging

🏗️ Architecture



📋 Workflow Conversion

Main Video Generation Pipeline

Converts n8n "Main Video Ad.json" workflow:

1. **Script Generation** (AI Agent1) → OpenAI GPT-4 client
2. **Image Processing** (Luma Photon) → FAL client reframe_image()
3. **Scene Enhancement** (Gemini) → FAL client enhance_image_with_gemini()
4. **Video Generation** (Minimax) → FAL client generate_video_from_image()
5. **Voice Generation** (ElevenLabs) → ElevenLabs client batch_text_to_speech()
6. **Music Generation** (Lyria2) → Lyria client generate_music()
7. **Video Composition** → FAL client compose_videos()
8. **Caption Addition** → FAL client add_captions()

Revision Pipeline

Converts n8n "Revision.json" workflow:

1. **Revision Analysis** → OpenAI client analyze_revision_request()
2. **Selective Regeneration** → Based on changed fields
3. **Video Recomposition** → Updated elements only
4. **Final Processing** → Captions and delivery

🚦 Getting Started

Prerequisites

- Docker and Docker Compose
- Python 3.11+ (if running locally)
- API Keys for: FAL AI, OpenAI, ElevenLabs

Quick Start

1. **Clone and Setup**

```bash
git clone <repository>
cd video-ai-backend
chmod +x run.sh
```

2. **Configure Environment**

```bash
```

```
cp .env.example .env
# Edit .env with your API keys
```

3. **Start Services**

```
bash

./run.sh
```

This will:

- Build Docker containers
- Start PostgreSQL, Redis, FastAPI, and Celery
- Run database migrations
- Start monitoring services

Manual Setup

1. **Install Dependencies**

```
bash

pip install -r requirements.txt
```

2. **Setup Database**

```
bash

# Start PostgreSQL and Redis
docker-compose up db redis -d

# Run migrations
alembic upgrade head
```

3. **Start Services**

```
bash
```

```
# Terminal 1: FastAPI
uvicorn app.main:app --reload

# Terminal 2: Celery Worker
celery -A app.tasks.celery_app worker --loglevel=info

# Terminal 3: Celery Beat (optional)
celery -A app.tasks.celery_app beat --loglevel=info
```

🔗 API Endpoints

Video Generation

```http
POST /api/v1/webhooks/video-generation
Content-Type: application/json

{
  "prompt": "Your video prompt here",
  "image_url": "https://example.com/image.jpg",
  "user_id": "user123",
  "chat_id": "chat456"
}
```

Check Status

```http
GET /api/v1/videos/{video_id}/status
```

Request Revision

```http
POST /api/v1/webhooks/revision
Content-Type: application/json

{
  "revision_request": "Change the voiceover in scene 2",
  "original_video_id": "original-video-id",
  "video_id": "new-video-id",
  "user_id": "user123"
}
```

🔧 Configuration

Environment Variables

```env
# Database
DATABASE_URL=postgresql+asyncpg://postgres:password@localhost:5432/video_ai

# Redis/Celery
REDIS_URL=redis://localhost:6379
CELERY_BROKER_URL=redis://localhost:6379/0

# API Keys (Required)
FAL_API_KEY=Key your-fal-api-key
OPENAI_API_KEY=sk-your-openai-key
ELEVENLABS_API_KEY=your-elevenlabs-key

# Base44 Integration
BASE44_CALLBACK_URL=https://base44.app/api/apps/your-app-id/functions/n8nVideoCallback

# App Settings
DEBUG=true
LOG_LEVEL=INFO
```

Webhook URLs

Update your Base44 webhooks to point to:

- **Main Video:** `https://your-domain.com/api/v1/webhooks/video-generation`
- **Revision:** `https://your-domain.com/api/v1/webhooks/revision`

📊 Monitoring

- **API Docs:** http://localhost:8000/docs
- **Health Check:** http://localhost:8000/health
- **Celery Monitor:** http://localhost:5555
- **Logs:** `docker-compose logs -f api`

🔄 Migration from n8n

1. **Export Data** from your current Supabase database
2. **Run Migration Script:**

```bash
```

```
# Update SUPABASE_URL in migrate_from_n8n.py
python migrate_from_n8n.py
```

3. **Update Base44 webhooks** to new endpoints
4. **Test** with sample requests
5. **Gradually switch** from n8n to new backend

🏃 Development

Project Structure

```
app/
├── api/v1/endpoints/      # API route handlers
├── core/                  # Configuration, database
├── models/                # SQLAlchemy models
├── schemas/               # Pydantic schemas
├── services/              # Business logic
│   ├── ai_clients/        # AI service clients
│   └── video_generation/  # Pipeline logic
├── tasks/                 # Celery tasks
└── utils/                 # Utilities
```

Adding New Features

1. **New AI Service:** Add client in `app/services/ai_clients/`
2. **Pipeline Step:** Extend `MainPipeline` or `RevisionPipeline`
3. **API Endpoint:** Add route in `app/api/v1/endpoints/`
4. **Background Task:** Add task in `app/tasks/`

Testing

```bash
# Install test dependencies
pip install pytest pytest-asyncio

# Run tests
pytest

# Run with coverage
pytest --cov=app
```

🚀 Deployment

Production Deployment

1. Environment Setup

```bash
# Production environment variables
DEBUG=false
DATABASE_URL=postgresql+asyncpg://user:pass@prod-db:5432/video_ai
SENTRY_DSN=your-sentry-dsn
```

2. Docker Production

```bash
docker-compose -f docker-compose.prod.yml up -d
```

3. Scaling Workers

```bash
# Scale Celery workers
docker-compose up --scale celery-worker=4
```

Cloud Deployment Options

- **AWS**: ECS + RDS + ElastiCache
- **GCP**: Cloud Run + Cloud SQL + Memorystore
- **Azure**: Container Instances + PostgreSQL + Redis Cache
- **DigitalOcean**: App Platform + Managed Database

📝 Performance Optimizations

- **Concurrent Processing**: Parallel scene generation
- **Caching**: Redis caching for AI responses
- **Connection Pooling**: Database connection pooling
- **Task Queues**: Separate queues for different task types
- **Resource Management**: Memory-efficient video processing

🐛 Troubleshooting

Common Issues

1. **Video Generation Fails**
   - Check API keys in `.env`
   - Verify FAL AI service status
   - Check Celery worker logs
2. **Database Connection Errors**
   - Ensure PostgreSQL is running
   - Check DATABASE_URL format
   - Verify network connectivity
3. **Celery Tasks Not Processing**
   - Check Redis connection
   - Verify Celery worker is running
   - Check task queue status

Debug Mode

```bash
# Enable debug logging
export LOG_LEVEL=DEBUG

# Run with detailed logs
docker-compose logs -f celery-worker
```

🤝 Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/new-feature`)
3. Commit changes (`git commit -am 'Add new feature'`)
4. Push to branch (`git push origin feature/new-feature`)
5. Create Pull Request

📄 License

🙏 Acknowledgments

- **n8n**: Original workflow inspiration
- **FastAPI**: Amazing async web framework
- **Celery**: Reliable task queue
- **Base44**: Platform integration
- **AI Services**: FAL AI, OpenAI, ElevenLabs, Lyria

📞 Support

- **Issues**: GitHub Issues
- **Documentation**: `/docs` endpoint
- **Monitoring**: Sentry integration
- **Logs**: Structured logging with Loguru

🎉 **Congratulations!** You've successfully converted your n8n video generation workflows to a robust Python backend. This system is now more maintainable, scalable, and easier to debug than the visual n8n workflows.