Multi-Tenant Payload CMS - Complete Project Documentation

📋 Project Overview

This project is a **multi-tenant SaaS CMS platform** built on Payload CMS v3.74.0 and Next.js 15.4. It enables a single codebase to power multiple independent client websites, each with their own isolated database, custom admin panel, and distinct collections.

Current Status & Achievements

✅ **Completed Implementation:**

- Multi-tenant architecture with tenant-based configuration
- PostgreSQL database isolation (separate databases per tenant)
- Two live tenants: **Mistrut** (pages-only) and **Synergy** (blog platform)
- Dynamic Payload configuration factory system
- Docker-based PostgreSQL with automatic database initialization
- Live preview functionality across all tenant instances
- Multi-instance startup automation

Active Tenants

Tenant	Port	Collections	Database	Admin URL
Mistrut	3001	Users, Media, Pages	`mistrut_db`	http://localhost:3001/admin
Synergy	3002	Users, Media, Posts	`synergy_db`	http://localhost:3002/admin

---

🏗️ Architecture & How It Works

Single Codebase, Multiple Instances

The system uses a **tenant configuration registry** (`src/tenants.config.ts`) that defines each tenant with:

```
{
  id: 'mistrut',                    // Unique identifier
  name: 'Mistrut',                  // Display name
  databaseUrl: 'postgresql://...',  // Isolated database
  port: 3001,                       // Dedicated port
  payloadSecret: 'secret-key',      // Unique encryption key
  customConfig: {
    collections: ['users', 'media', 'pages']  // Tenant-specific collections
  }
}
```

Payload Configuration Factory

The `src/payload.config.ts` implements a **factory pattern** that generates tenant-specific configurations:

1. **Runtime Tenant Detection**: Reads `TENANT_ID` from environment variables
2. **Dynamic Collection Loading**: Only loads collections specified in tenant config
3. **Database Isolation**: Each tenant connects to their own PostgreSQL database
4. **Branded Admin Panel**: Customizes admin UI with tenant-specific branding

Key Code Flow:

```
// At runtime
const tenant = getCurrentTenant(); // Gets tenant from TENANT_ID env
export default createPayloadConfig(tenant); // Builds custom Payload config
```

Database Architecture

Setup: Single PostgreSQL server (v16-alpine) running multiple isolated databases

```
PostgreSQL Server (localhost:5433)
├── mistrut_db       (Mistrut tenant database)
├── synergy_db       (Synergy tenant database)
└── postgres         (PostgreSQL system database)
```

Isolation Strategy:

- Each tenant has a completely separate database
- Zero data sharing between tenants
- Independent schema migrations per tenant
- Separate admin users and authentication

Database Tables (per tenant):

- `users` - Admin users and authentication
- `pages` / `posts` - Content (varies by tenant)
- `media` - File uploads with sharp image processing
- `payload_migrations` - Migration history
- `payload_preferences` - User preferences

🐳 Docker Configuration

PostgreSQL Container

File: `docker-compose.postgres.yml`

```
services:
  postgres:
    image: postgres:16-alpine
    ports: ["5433:5432"] # Exposed on host port 5433
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./scripts/init-databases.sql:/docker-entrypoint-initdb.d/01-init-
databases.sql
```

Automated Database Initialization: `scripts/init-databases.sql`

- Runs automatically on first container startup
- Creates `mistrut_db` and `synergy_db` databases
- Grants permissions to `payload` user

Optional: pgAdmin Web UI

```
docker-compose --profile tools -f docker-compose.postgres.yml up -d
# Access: http://localhost:5050
# Login: admin@payload.com / admin
```

🚀 Running the Application

Prerequisites

```
node: ^18.20.2 || >=20.9.0
pnpm: ^9 || ^10
Docker: For PostgreSQL
```

1️⃣ Start PostgreSQL

```
docker-compose -f docker-compose.postgres.yml up -d
```

Verify it's running:

```
docker ps | grep payload-postgres
docker logs payload-postgres
```

2️⃣ Load Environment Variables

```
# Load multi-tenant configuration
cp .env.multi .env
source .env.multi
```

Key Environment Variables (`.env.multi`):

- `MISTRUT_DATABASE_URL` - PostgreSQL connection for Mistrut
- `MISTRUT_PAYLOAD_SECRET` - Encryption key for Mistrut
- `MISTRUT_PORT` - Port number (3001)
- Same pattern for Synergy (3002)

3️⃣ Run Database Migrations

Each tenant requires separate migrations:

```
# Mistrut
TENANT_ID=mistrut \
  DATABASE_URL="$MISTRUT_DATABASE_URL" \
  PAYLOAD_SECRET="$MISTRUT_PAYLOAD_SECRET" \
  pnpm payload migrate

# Synergy
TENANT_ID=synergy \
  DATABASE_URL="$SYNERGY_DATABASE_URL" \
  PAYLOAD_SECRET="$SYNERGY_PAYLOAD_SECRET" \
  pnpm payload migrate
```

4️⃣ Start All Instances

Option A: Automated Script (Recommended)

```
./scripts/start-multi.sh
```

Option B: Manual (Separate Terminals)

```
# Terminal 1 - Mistrut
TENANT_ID=mistrut \
  DATABASE_URL="$MISTRUT_DATABASE_URL" \
  PAYLOAD_SECRET="$MISTRUT_PAYLOAD_SECRET" \
  PORT=3001 \
  npm run dev

# Terminal 2 - Synergy
TENANT_ID=synergy \
  DATABASE_URL="$SYNERGY_DATABASE_URL" \
  PAYLOAD_SECRET="$SYNERGY_PAYLOAD_SECRET" \
  PORT=3002 \
  npm run dev
```

5️⃣ Access the Applications

Service	URL	Purpose
Mistrut Admin	http://localhost:3001/admin	CMS admin panel
Mistrut Frontend	http://localhost:3001	Public website
Synergy Admin	http://localhost:3002/admin	CMS admin panel
Synergy Frontend	http://localhost:3002	Public website
PostgreSQL	localhost:5433	Database (user: payload, pass: payload)

📊 Database Access & Management

Method 1: psql (Command Line)

```
# Connect to Mistrut database
psql -h localhost -p 5433 -U payload -d mistrut_db
# Password: payload

# Useful commands
\dt                             # List tables
SELECT * FROM pages;            # View pages
SELECT email FROM users;        # View users
\q                              # Exit
```

Method 2: Docker exec

```
# Connect to PostgreSQL container
docker exec -it payload-postgres psql -U payload

# View all databases
\l

# Connect to specific tenant
\c mistrut_db
\dt  # List tables
```

Method 3: GUI Tools

DBeaver / pgAdmin / TablePlus:

- Host: `localhost`
- Port: `5433`
- Username: `payload`
- Password: `payload`
- Database: `mistrut_db` or `synergy_db`

📁 Project Structure

```
cms-poc-payload/
├── src/
│   ├── tenants.config.ts          # ⭐ Tenant registry and configuration
│   ├── payload.config.ts          # ⭐ Factory function for Payload config
│   ├── collections/
│   │   ├── Users.ts               # Authentication collection (all tenants)
│   │   ├── Media.ts               # File uploads (all tenants)
│   │   ├── Pages.ts               # Page builder (Mistrut only)
│   │   └── Posts.ts               # Blog posts (Synergy only)
│   ├── globals/
│   │   ├── Header.ts              # Global navigation configuration
│   │   └── DesignSystem.ts        # Theme settings per tenant
│   └── app/                       # Next.js 15 app directory
│       ├── (payload)/              # Payload admin routes
│       ├── [slug]/page.tsx         # Dynamic page rendering
│       └── posts/[slug]/page.tsx   # Blog post rendering
├── scripts/
│   ├── init-databases.sql          # PostgreSQL initialization script
│   └── start-multi.sh              # Multi-instance startup automation
├── docker-compose.postgres.yml     # PostgreSQL Docker configuration
├── .env.multi                      # Multi-tenant environment variables
└── package.json                    # Dependencies (Payload 3.74, Next 15.4)
```

🔧 Common Commands

Development

```
# Single instance (traditional)
npm run dev                          # Start on port 3000

# Multi-instance (current setup)
./scripts/start-multi.sh             # Start all tenants

# Clean build
npm run devsafe                      # Remove .next and restart
```

Database Operations

```
# Run migrations
TENANT_ID=<tenant> DATABASE_URL="<url>" PAYLOAD_SECRET="<secret>" pnpm payload
migrate

# Reset migrations (⚠️ destructive)
TENANT_ID=<tenant> DATABASE_URL="<url>" PAYLOAD_SECRET="<secret>" pnpm payload
migrate:reset

# Generate TypeScript types
npm run generate:types
```

Docker Management

```
# Start PostgreSQL
docker-compose -f docker-compose.postgres.yml up -d

# Stop PostgreSQL
docker-compose -f docker-compose.postgres.yml down

# View logs
docker logs payload-postgres
```

```
# Restart container
docker-compose -f docker-compose.postgres.yml restart
```

➕ Adding New Tenants

1. Update Tenant Configuration

Edit `src/tenants.config.ts`:

```
{
  id: 'newclient',
  name: 'New Client',
  domain: 'newclient.local',
  adminDomain: 'admin.newclient.local',
  databaseUrl: process.env.NEWCLIENT_DATABASE_URL || 'postgresql://...',
  port: 3003,
  payloadSecret: process.env.NEWCLIENT_PAYLOAD_SECRET,
  customConfig: {
    collections: ['users', 'media', 'pages', 'posts']  // Choose collections
  }
}
```

2. Add Environment Variables

Edit `.env.multi`:

```
NEWCLIENT_DATABASE_URL=postgresql://payload:payload@localhost:5433/newclient_db
NEWCLIENT_PAYLOAD_SECRET=$(openssl rand -base64 32)
NEWCLIENT_PORT=3003
NEWCLIENT_NEXT_PUBLIC_SERVER_URL=http://localhost:3003
```

3. Create Database

```
docker exec -it payload-postgres psql -U payload -c "CREATE DATABASE newclient_db;"
```

4. Run Migrations

```
TENANT_ID=newclient \
  DATABASE_URL="$NEWCLIENT_DATABASE_URL" \
  PAYLOAD_SECRET="$NEWCLIENT_PAYLOAD_SECRET" \
  pnpm payload migrate
```

5. Start Instance

```
TENANT_ID=newclient \
  DATABASE_URL="$NEWCLIENT_DATABASE_URL" \
  PAYLOAD_SECRET="$NEWCLIENT_PAYLOAD_SECRET" \
  PORT=3003 \
  npm run dev
```

🛑 Troubleshooting

Port Already in Use

```
lsof -i :3001          # Find process using port
kill -9 <PID>          # Kill process
```

PostgreSQL Connection Failed

```
docker ps | grep payload-postgres      # Check if running
docker-compose -f docker-compose.postgres.yml restart
```

Migration Errors

```
# Reset and re-run migrations
TENANT_ID=<tenant> DATABASE_URL="<url>" PAYLOAD_SECRET="<secret>" pnpm payload
migrate:reset
TENANT_ID=<tenant> DATABASE_URL="<url>" PAYLOAD_SECRET="<secret>" pnpm payload
migrate
```

Database Not Found

```
# Manually create database
docker exec -it payload-postgres psql -U payload -c "CREATE DATABASE <dbname>;"
```

📚 Key Technologies

Technology	Version	Purpose
Payload CMS	3.74.0	Headless CMS engine
Next.js	15.4.11	React framework for frontend/backend
PostgreSQL	16-alpine	Relational database
React	19.2.1	UI library
TypeScript	5.7.3	Type safety
Sharp	0.34.2	Image processing
Lexical	@payloadcms/richtext-lexical	Rich text editor

🎯 Current Capabilities

✅ Complete data isolation between tenants
✅ Per-tenant collection customization
✅ Live preview with field focus sync
✅ Image uploads with automatic optimization
✅ Flexible page builder with blocks architecture
✅ Blog platform (Posts collection)
✅ User authentication per tenant
✅ Automated database initialization
✅ Multi-instance development environment

📖 **Full Setup Guide**: MULTI_INSTANCE_SETUP.md
📊 **Database Access Guide**: DATABASE_ACCESS.md