Fractal Generator

A powerful, full-stack interactive fractal explorer that allows users to define custom formulas, manipulate colors through gradients, and navigate complex planes in real-time.

Features

- **Custom Formula Engine**: Define your own iteration formulas and divergence criteria using a custom-built stack-based execution engine.
- **Interactive Exploration**: Drag to pan and scroll to zoom into the infinite complexity of fractals.
- **Dynamic Color Gradients**: Fully customizable outside-color gradients and inside-color selection.
- **User Accounts**: Register and log in to save and manage your favorite fractal configurations.
- **High-Performance Rendering**: Server-side rendering using .NET 10 and SkiaSharp for precise and efficient image generation.
- **Advanced Mathematical Support**: Support for complex number operations including trigonometric, exponential, and logarithmic functions.

Tech Stack

Frontend

- **Framework**: React 19
- **Build Tool**: Vite
- **State Management**: TanStack Query v5
- **Routing**: React Router v7
- **Language**: TypeScript
- **API Generation**: NSwag

Backend

- **Framework**: .NET 10 Web API
- **Graphics**: SkiaSharp for high-performance PNG rendering.
- **Database**: PostgreSQL
- **ORM**: Entity Framework Core
- **Authentication**: JWT (JSON Web Tokens) with Refresh Token support.

- **Documentation**: OpenAPI / Swagger (via NSwag).

Infrastructure

- **Reverse Proxy**: Traefik
- **Containerization**: Docker & Docker Compose
- **Database Management**: pgAdmin 4

Getting Started

Prerequisites

- Docker Desktop
- mkcert (optional, for local HTTPS)
- Node.js (for certificate setup and frontend development)

Quick Start (Docker)

1. **Environment Setup**: Copy the *.env.example* file and rename it to *.env*. Edit the *.env* with your desired credentials
2. **(Optional) Local HTTPS Setup**: To avoid browser security warnings, install local certificates:

```
# Install mkcert (Windows example via Choco)
choco install mkcert

# Generate certificates for the project
node ./FractalGenerator.Frontend/certs/setup-certs.ts
```

3. **Launch the Application**:

```
# Basic start
docker-compose -f docker-compose.yml -f docker-compose.traefik.yml up -d

# With trusted certificates (if step 2 was performed)
docker-compose -f docker-compose.yml -f docker-compose.traefik.yml -f docker-
compose.traefik.trusted.yml up -d
```

4. **Access the Services**:
    - **Web App**: https://localhost:8083
    - **Traefik Dashboard**: http://localhost:8084
    - **pgAdmin**: http://localhost:5050

Development

The setup for development is described in the *README.md* in the *FractalGenerator.Frontend* and *FractalGenerator.WebApi* folders.

Architecture

The system follows a clean, modular architecture:

- **Frontend**: A React SPA that handles user interaction, state synchronization, and canvas manipulation. It communicates with the backend via a generated TypeScript client.
- **Web API**: An ASP.NET Core layer providing RESTful endpoints for authentication, fractal configuration management, and fractal generation.
- **Service Layer**: Contains the core business logic, including the **Formula Parser** (Lexer/Parser) and the **Execution Engine** (Virtual Machine) that iterates over the complex plane using stack-based instructions.
- **Data Layer**: EF Core with a PostgreSQL provider for persisting users, their saved configurations, and refresh tokens.