🚀 Quick-Start Guide: .NET 8, React & PostgreSQL

Architecture: Docker (DB) + ASP.NET Core 8 (API) + React (UI) + Azure Data Studio

---

Phase 1: The Database (PostgreSQL via Docker)

Goal: Get the database engine running and connected to a management tool.

1. Start the Database

Open your terminal (PowerShell or CMD) and run:

Bash

```
docker run --name my-postgres -e POSTGRES_PASSWORD=mysecretpassword -p 5432:5432 -d postgres
```

2. Connect with Azure Data Studio (ADS)

1. Open **Azure Data Studio**.
2. Click **New Connection**.
3. Enter these details:
    - **Connection Type:** PostgreSQL (Ensure extension is installed)
    - **Server Name:** `localhost`
    - **User Name:** `postgres`
    - **Password:** `mysecretpassword`
    - **Database Name:** `postgres`
4. Click **Connect**.

Phase 2: The Backend (ASP.NET Core 8 API)

Goal: Create the API, connect it to the DB, and set up the background worker.

1. Create Project & Install Packages

Bash

```
dotnet new webapi -n MyWebApp.Api
```

```
cd MyWebApp.Api
dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL
dotnet add package Microsoft.EntityFrameworkCore.Design
```

2. Configure `appsettings.json`

Add the connection string:

JSON

```
{
  "ConnectionStrings": {
    "DefaultConnection":
"Host=localhost;Port=5432;Database=MyFullStackDb;Username=postgres;Password=mysecre
tpassword"
  },
  "AllowedHosts": "*"
}
```

3. Setup `Program.cs`

Update your `Program.cs` to include the DB connection, CORS (for React), and the Background Worker.

C#

```
using Microsoft.EntityFrameworkCore;
using MyWebApp.Api; // Namespace for your Worker class

var builder = WebApplication.CreateBuilder(args);

// 1. Add DB Context
var connectionString =
builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<AppDbContext>(options =>
    options.UseNpgsql(connectionString));

// 2. Add Background Worker (Free alternative to Console App)
builder.Services.AddHostedService<MyBackgroundWorker>();

// 3. Add CORS (Allow React)
builder.Services.AddCors(options => {
    options.AddPolicy("AllowReact", policy =>
        policy.WithOrigins("http://localhost:5173")
              .AllowAnyMethod()
              .AllowAnyHeader());
});

builder.Services.AddControllers();
var app = builder.Build();

// 4. Use CORS
app.UseCors("AllowReact");

app.MapControllers();
app.Run();
```

4. Create the Background Worker (`Worker.cs`)

Create a new class file named `Worker.cs`:

C#

```
public class MyBackgroundWorker : BackgroundService
{
    protected override async Task ExecuteAsync(CancellationToken stoppingToken)
    {
        while (!stoppingToken.IsCancellationRequested)
        {
            // Your background logic goes here
            Console.WriteLine($"Worker running at: {DateTime.UtcNow}");
            await Task.Delay(TimeSpan.FromHours(1), stoppingToken);
        }
    }
}
```

5. Create Database Context (`Data/AppDbContext.cs`)

C#

```
public class AppDbContext : DbContext
{
    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }
    // Add your tables here, e.g., public DbSet<User> Users { get; set; }
}
```

6. Run Migrations

Bash

```
dotnet ef migrations add InitialCreate
dotnet ef database update
```

Phase 3: The Frontend (React + Vite)

Goal: Create the UI and fetch data from the API.

1. Create React App

Open a terminal **outside** your API folder:

Bash

```
npm create vite@latest my-react-app -- --template react
cd my-react-app
npm install
```

2. Update `src/App.jsx`

JavaScript

```
import { useEffect, useState } from 'react';

function App() {
  const [data, setData] = useState([]);

  useEffect(() => {
    // Ensure this port matches your running API port
    fetch('http://localhost:5000/weatherforecast')
      .then(res => res.json())
      .then(result => setData(result))
      .catch(err => console.error(err));
  }, []);

  return (
    <div>
      <h1>My Full Stack App</h1>
      <pre>{JSON.stringify(data, null, 2)}</pre>
    </div>
  );
}

export default App;
```

3. Run React

Bash

```
npm run dev
```

Open your browser to the URL shown (usually `http://localhost:5173`).

✅ Checklist for Success

1. [ ] **Docker container** is running (`docker ps` shows it).
2. [ ] **ADS** can connect to the database.
3. [ ] **.NET API** is running (check the port, usually 5000 or similar).
4. [ ] **React App** is running and shows data (no CORS errors in browser console).