Guía Completa de Configuración NoeAPI en CachyOS

1. 🗄️ Instalación de SQL Server en CachyOS

Opción A: SQL Server en Docker (Recomendado)

bash

```bash
# Instalar Docker si no lo tienes
sudo pacman -S docker docker-compose
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker $USER
# Reinicia la sesión o ejecuta: newgrp docker

# 1. Eliminar el contenedor problemático
docker rm -f sqlserver

# 2. Eliminar el directorio con permisos incorrectos
sudo rm -rf ~/sqlserver-data

# 3. Crear el directorio con permisos correctos
mkdir -p ~/sqlserver-data

# 4. Dar permisos completos al directorio (SQL Server necesita UID 10001)
sudo chown -R 10001:0 ~/sqlserver-data
sudo chmod -R 755 ~/sqlserver-data

# 5. Crear el contenedor nuevamente
docker run -e "ACCEPT_EULA=Y" \
           -e "MSSQL_SA_PASSWORD=YourStrong@Passw0rd" \
           -p 1433:1433 \
           --name sqlserver \
           --hostname sqlserver \
           -v ~/sqlserver-data:/var/opt/mssql \
           -d mcr.microsoft.com/mssql/server:2022-latest

# 6. Verificar que inicie correctamente
docker start sqlserver
docker ps
docker logs -f sqlserver

# Verificar que está ejecutándose
docker ps
```

Opción B: PostgreSQL como alternativa (Más fácil en Linux)

bash

```bash
# Instalar PostgreSQL
sudo pacman -S postgresql

# Inicializar base de datos
sudo -u postgres initdb -D /var/lib/postgres/data

# Habilitar y iniciar servicio
sudo systemctl enable postgresql
sudo systemctl start postgresql

# Crear usuario y base de datos
sudo -u postgres createuser --interactive
# Nombre: tu_usuario, Superuser: y

sudo -u postgres createdb noeapidb -O tu_usuario

# Establecer contraseña para el usuario
sudo -u postgres psql
ALTER USER tu_usuario PASSWORD 'tu_contraseña';
\q
```

2. 🔧 Configuración del Connection String

Para SQL Server (Docker):

json

```json
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost,1433;Database=NoeApiDB;User Id=sa;Password='
  }
}
```

Para PostgreSQL (si elegiste esta opción):

Instalar paquete NuGet en Noe.DAL:

bash

```bash
dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL
```

Modificar Program.cs:

```csharp
// Cambiar esta línea:
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"))

// Por esta:
options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection"))
```

Connection String para PostgreSQL:

```json
{
  "ConnectionStrings": {
    "DefaultConnection": "Host=localhost;Database=noeapidb;Username=tu_usuario;Passwor
  }
}
```

3. 🔑 Configurar Google Client ID

Crear proyecto en Google Cloud Console:

1. Ve a Google Cloud Console
2. Crea un nuevo proyecto o selecciona uno existente
3. Habilita la **Google+ API** y **Google Identity API**
4. Ve a **Credenciales** → **Crear credenciales** → **ID de cliente OAuth 2.0**
5. Configura:
   - **Tipo de aplicación**: Aplicación web
   - **Orígenes autorizados**: `http://localhost:4200`, `https://localhost:4200`
   - **URI de redirección**: `http://localhost:4200/auth/callback`

Actualizar appsettings.json:

```json
{
  "Google": {
    "ClientId": "tu-client-id-aqui.apps.googleusercontent.com"
  }
}
```

4. 🔐 Generar Clave JWT Segura

Generar clave aleatoria segura:

bash

```bash
# Opción 1: Con OpenSSL
openssl rand -base64 32

# Opción 2: Con Node.js (si lo tienes)
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"

# Opción 3: Manual en C#
dotnet run --project temp_project -c "Console.WriteLine(Convert.ToBase64String(System.
```

Actualizar appsettings.json:

json

```json
{
  "Jwt": {
    "Key": "tu-clave-generada-de-32-caracteres-o-mas-aqui",
    "Issuer": "NoeAPI",
    "Audience": "NoeAPI"
  }
}
```

5. 🌐 Configurar CORS para Angular

En Program.cs (ya incluido en el código):

csharp

```csharp
builder.Services.AddCors(options =>
{
    options.AddDefaultPolicy(builder =>
    {
        builder.WithOrigins("http://localhost:4200", "https://localhost:4200")
               .AllowAnyHeader()
               .AllowAnyMethod()
               .AllowCredentials();
    });
});
```

6. 🏗️ Configurar y Ejecutar Proyecto

Crear estructura de proyectos:

bash

```bash
mkdir NoeAPI
cd NoeAPI

# Crear solución
dotnet new sln -n NoeAPI

# Crear proyectos
dotnet new classlib -n Noe.Models
dotnet new classlib -n Noe.DAL
dotnet new classlib -n Noe.BLL
dotnet new webapi -n Noe.API

# Agregar proyectos a la solución
dotnet sln add Noe.Models/Noe.Models.csproj
dotnet sln add Noe.DAL/Noe.DAL.csproj
dotnet sln add Noe.BLL/Noe.BLL.csproj
dotnet sln add Noe.API/Noe.API.csproj

# Agregar referencias entre proyectos
cd Noe.DAL
dotnet add reference ../Noe.Models/Noe.Models.csproj

cd ../Noe.BLL
dotnet add reference ../Noe.DAL/Noe.DAL.csproj
dotnet add reference ../Noe.Models/Noe.Models.csproj

cd ../Noe.API
dotnet add reference ../Noe.BLL/Noe.BLL.csproj
dotnet add reference ../Noe.DAL/Noe.DAL.csproj
dotnet add reference ../Noe.Models/Noe.Models.csproj
```

Instalar paquetes NuGet:

bash

```bash
# En Noe.DAL
cd Noe.DAL
dotnet add package Microsoft.EntityFrameworkCore
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
dotnet add package Microsoft.EntityFrameworkCore.Tools

# En Noe.BLL
cd ../Noe.BLL
dotnet add package AutoMapper
dotnet add package AutoMapper.Extensions.Microsoft.DependencyInjection
dotnet add package Google.Apis.Auth
dotnet add package Microsoft.IdentityModel.Tokens
dotnet add package System.IdentityModel.Tokens.Jwt
dotnet add package Microsoft.Extensions.Configuration.Abstractions

# En Noe.API
cd ../Noe.API
dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer
dotnet add package Microsoft.EntityFrameworkCore.Design
dotnet add package Swashbuckle.AspNetCore
```

7. 🗃️ Ejecutar Migrations

bash

```bash
# Desde la carpeta raíz del proyecto
cd NoeAPI

# Crear migración inicial
dotnet ef migrations add InitialCreate --project Noe.DAL --startup-project Noe.API

# Aplicar migración a la base de datos
dotnet ef database update --project Noe.DAL --startup-project Noe.API

# Si hay errores, verificar connection string y que SQL Server esté ejecutándose
```

8. ▶️ Ejecutar la API

bash

```bash
# Desde Noe.API
cd Noe.API
dotnet run

# O para desarrollo con hot reload
dotnet watch run
```

La API estará disponible en:

- HTTP: `http://localhost:5000`
- HTTPS: `https://localhost:5001`
- Swagger: `https://localhost:5001/swagger`

9. 🧪 Verificar Configuración

Probar SQL Server:

bash

```bash
# Si usas Docker
docker exec -it sqlserver /opt/mssql-tools/bin/sqlcmd -S localhost -U sa -P 'YourStron
SELECT @@VERSION;
GO
```

Probar PostgreSQL:

bash

```bash
psql -h localhost -U tu_usuario -d noeapidb
\dt  -- Listar tablas
\q   -- Salir
```

Probar endpoints:

bash

```bash
# Probar endpoint público
curl https://localhost:5001/api/auth/me

# Debería devolver 401 Unauthorized (correcto, necesita autenticación)
```

10. 🔧 Solución de Problemas Comunes

SQL Server no conecta:

bash

```bash
# Verificar que el contenedor esté ejecutándose
docker ps

# Ver logs del contenedor
docker logs sqlserver

# Reiniciar contenedor
docker restart sqlserver
```

Error de certificados SSL:

bash

```bash
# Generar certificados de desarrollo
dotnet dev-certs https --trust
```

Error en migrations:

bash

```bash
# Limpiar migrations
dotnet ef migrations remove --project Noe.DAL --startup-project Noe.API

# Recrear
dotnet ef migrations add InitialCreate --project Noe.DAL --startup-project Noe.API
```

📝 Archivo appsettings.json Completo

json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost,1433;Database=NoeApiDB;User Id=sa;Password=`
  },
  "Jwt": {
    "Key": "tu-clave-jwt-muy-segura-de-al-menos-32-caracteres-aqui",
    "Issuer": "NoeAPI",
    "Audience": "NoeAPI"
  },
  "Google": {
    "ClientId": "tu-google-client-id.apps.googleusercontent.com"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

🚀 Siguiente Paso: Angular

Una vez que tengas la API ejecutándose, puedes integrarla con Angular usando el Google Sign-In button y llamando a tus endpoints.

¡Con esto tendrás tu NoeAPI completamente funcional en CachyOS!