FitnessAnalyticsHub Dokumentation

**Bereits enthalten:**

Entity Framework

DB mit SqlLite

CleanArchitecture (mit Jason Projekt vergleichen)

Solid Regeln

RepostioryPattern (erklären lassen) ist das doppelt gemoppel mit DBContext?

**Todo:**

UserSecrets verwenden (Strava, OpenAPI)

Oberfläche mit Angular…

Im Entities Ordner ValueObjects einbauen

DataAnnotations in die Klassen (required) etc

In alle Methoden mit async await CancellationToken rein

SonarQube Coderegeln überwachen?

Arc42?

Architekturüberwachung mit Tool?

Doker, Kubernetes

Azure

Irgendwas mit ChatGpt 😊

Hangfire

HealthChecks

RabbitMq

GitActions

DomainDrivenDesign

Doku zu bestehenden Funktionen erstellen lassen

UserSecrets:

## Für die lokale Entwicklung: User Secrets

In .NET gibt es einen integrierten Mechanismus namens "User Secrets", der ideal für die Entwicklungsphase ist:

1. **\*\*Aktiviere User Secrets in deinem Projekt:\*\***

```bash

# Im Projektverzeichnis ausführen

dotnet user-secrets init

```

Dies fügt eine `UserSecretsId` zu deiner `.csproj` Datei hinzu.

2. **\*\*Speichere deinen API-Schlüssel:\*\***

```bash

dotnet user-secrets set "OpenAI:ApiKey" "dein-echter-api-key"

```

3. **\*\*Zugriff im Code:\*\***

Die Secrets werden automatisch in die Konfiguration geladen und über den normalen `IConfiguration`-Mechanismus verfügbar.

## Für die Produktion: Umgebungsvariablen oder Azure Key Vault

### Option 1: Umgebungsvariablen

In `Program.cs` ergänzen:

```csharp

*// Füge diese Zeile vor dem builder.Build() hinzu*

builder.Configuration.AddEnvironmentVariables();

```

Auf dem Server/in der Produktionsumgebung setzt du dann die Umgebungsvariable:

```

OPENAI\_\_APIKEY=dein-api-key

```