MIKROSERWISY .NET 5

Paweł Łabno

Zawartość

WPROWADZENIE TEORETYCZNE

PODSTAWY PROJEKTU

BAZY DANYCH

BUDOWANIE KONFIGURACJI

CACHOWANIE

KOMUNIKACJA

Prowadzący

- Starszy specjalista ds. rozwoju oprogramowania w Seville More Helory
- Absolwent AGH i UJ
- pawellabno1992@gmail.com
- plabno@sevillemore.com
- Paweł Łabno | LinkedIn
- Pasjonat gier planszowcyh

Materiały

Przedsesyjne

- Repozytorium

Po sesyjne

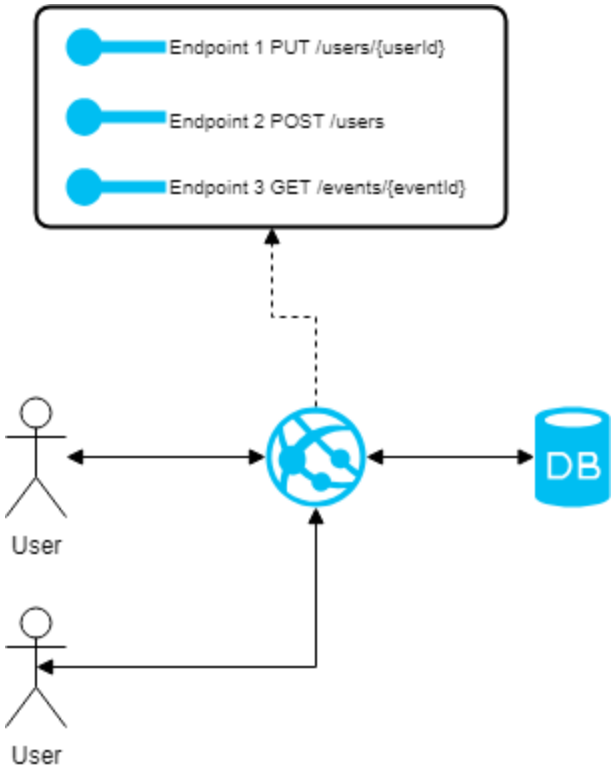
- Gotowy projekt
- Prezentacja
- PDF

Wprowadzenie do warsztatów

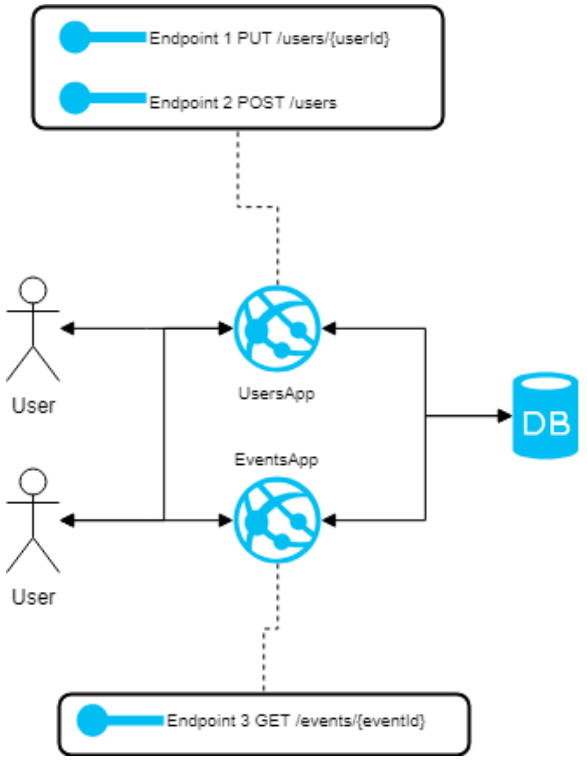
- 3 rodzaje slajdów
  - *Teoria*
  - *Demo*
  - *Zadanie*
- Pytania zadawać jak najszybciej można ;)
- Zewnętrzne zasoby na czas prezentacji
- 1 planowana przerwa
- Korzystajcie z opcji „Live share”

WPROWADZENIE TEORETYCZNE

Aplikacja


Endpoint 1 PUT /users/{userId}
Endpoint 2 POST /users
Endpoint 3 GET /events/{eventId}
DB
User
User

Mikroserwis



Mikroserwis

Budowa aplikacji jako zbiór luźno powiązanych usług. W architekturze mikroserwisów, usługi obsługują względnie atomiczną funkcjonalność a udostępniane protokoły komunikacji są lekkie.

Cechy mikroserwisu

- Modularność
- Skalowalność
- Integrowalność
- Rozproszony rozwój
- Modularne wdrożenie

NANOSERWISY

Czym są kontenery?

- Izolowanym środowiskiem
- Dostarczającym własnych funkcjonalności
- Komunikują się poprzez zdefiniowane kanały
- Korzystają ze wspólnego kernela systemu

CZY POTRZEBNE SĄ NAM KONTENERY?

Kontenery kontra aplikacje

Aplikacje

- Prostsza konfiguracja (ARM)
- Spełniają większość wymagań stawianych aplikacjom

Kontenery

- Niezależność od platformy
- Możliwość agregacji w „clustry usług”
- Obsługa GDI
- Możliwość obsługi skomplikowanych wymagań

Modele wdrożeń mikroserwisów

- Service discovery (Consul)
- API Gateway

CQRS

Segragacja odpowiedzialności

- Query – pobranie danych
- Command – aktualizacja danych
- Event – zmiana w zewnętrznym zasobie

PODSTAWY PROJEKTU

Projekt

- https://github.com/paqaos/SFI2021-Workshops
- Po pobraniu uruchomcie budowę projektu (pobranie paczek z nugeta)

Opis systemu

- Obsługa konferencji
  - *Użytkownicy (Users)*
  - *Warsztaty i sesje (Events)*
  - *Uczestnictwo w sesjach (Participants)*

Struktura projektu

- Common - zestaw klas wykorzystywanych przez wszystkie projekty
  - *BusinessLayer - logika biznesowa*
  - *DatabaseLayer – warstwa danych*
  - *MigrationExecutor – projekt zawierający migracje*
- For each API
  - *Services*
    - Users
    - Events
    - Attendance
  - *Projects*
    - API – obsługa komunikacji z użytkownikiem
    - BusinessLayer
    - DatabaseLayer
    - Dto – zestaw obiektów wykorzystywanych przez danych projekt na wejściu / wyjściu

Uruchomienie projektu

- Debugowanie z poziomu VisualStudio
- Konsola `dotnet build` `dotnet run`
- Poziom Dockera

PODSTAWY PROJEKTU

Wstrzykiwanie zależności

Po co nam wstrzykiwanie zależności

- Wiele implementacji tego samego interfejsu
- Konstrukcja new ClassDeclaration() + kontrola nad Dispose
- New ClassDeclaration(a,b,c,d,e,f,g,h,i,j,k,…, eeeMacarena);

Narzędzia do wstrzykiwania zależności

- Autofac
- Castle Windsor
- Nijnect
- **SimpeInjector**
- Unity
- Kontener wbudowany w platformę .NET (ServiceCollection)

Czas życia obiektu

- Singleton
- Scoped
- Transient

Przygotowanie SimpleInjectora

- Pobranie paczki `SimpleInjector.Integration.AspNetCore.Mvc`
- Utworzenie kontenera SimpleInjectora
- Dodanie AddSimpleInjector w Startup.ConfigureService
- Dodanie UseSimpleInjector w Startup.Configure
- Konfiguracja SimpleInjectora

Możliwości SimpleInjectora

Typ	Opis	Metoda na kontenerze
Implementacja	Rejestrowanie standardowej implementacji	Register<X> lub Register<I,X>
Instancja	Rejestrowanie konkretnego obiektu	RegisterInstance
Długa inicjalizacja	Uruchomienie długotrwałego procesu	RegisterInitializer
Rejestracja szablonów	Rejestrowanie wielu generycznych implementacji	Register(typeof(GenericClass<>))
Klasy z Assembly	Rejestrowanie wielu klas danego typu z Assembly	Register(typeof(IService<>), typeof(UserService).Assembly)
Rejestrowanie kolekcji	Rejestrowanie kolekcji obsługującej jakieś zdarzenie	Collection.Register()
Rejestrowanie dekoratorów	Rejestrowanie klas dekorujących inne klasy	RegisterDecorator<Decorated, Decorator>
Rejestrowanie warunkowe	Rejestrowanie klas w przypadku gdy np. nie ma istniejącej obsługi danego typu	RegisterConditional

Rejestrowanie usług

- Zarejestruj w aplikacji kolekcję domyślnych serwisów i repozytoriów (MemoryBased) CRUDowych oraz Event Handlerów
- Zainicjuj domyślną zawartość obiektów typu Events

Zadania – wstrzykiwanie zależności

- Zaimplementuj oraz zarejestruj własny serwis do outputu (ConsoleOutput) jako rozszerzenie Ioutput – możesz wykorzystać Console.WriteLine
- Wzbogacaj każdą wiadomość o timestamp / datę umieszczenia (Decorator TimestampOutputDecorator) / RegisterDecorator

PODSTAWY PROJEKTU

Dokumentacja

Zalety generowania dokumentacji Swagger

- Jeden standard (Swagger / OpenAPI)
- Możliwość wysyłania testowych requestów
- Możliwość wygenerowania klientów http w kodzie

Przygotowanie Swaggera

- Pobranie paczki `Swashbuckle.AspNetCore`
- Dodanie AddSwagger oraz AddSwaggerUI w Startup.ConfigureService
- Dodanie UseSwagger w Startup.Configure

Generowanie dokumentacji

- Uruchomienie wizualnego endpointa do dokumentowania kodu
- Dodatkowe flagi oraz opcje konfiguracji

PODSTAWY PROJEKTU

Mechanizmy mapujące

Zalety bibliotek mappujących (AutoMapper)

- Uniknięcie przepisywania wartości jedna po drugiej
- Możliwość przygotowania konwerterów dla skomplikowanych scenariuszy
- Możliwość organizacji mapowań w profile
  - *Jeden profil dla projektu*
  - *Wiele profili dla projektu*
- Możliwość łatwego testowania

Wykorzystanie AutoMappera

- Wykorzystanie biblioteki `AutoMapper` oraz `AutoMapper.Extensions.Microsoft.DependencyInjection`
- Dodanie konfiguracji mapowań do usługi
- Wykorzystanie funkcji mapujących

Zadanie

- W projekcie Participants – kontroler zwróć obiekt ParticipantDto (z najważniejszymi danymi). Do prezentacji danych wykorzystaj AutoMappera

PODSTAWY PROJEKTU

Walidacja danych wejściowych

Walidacja danych

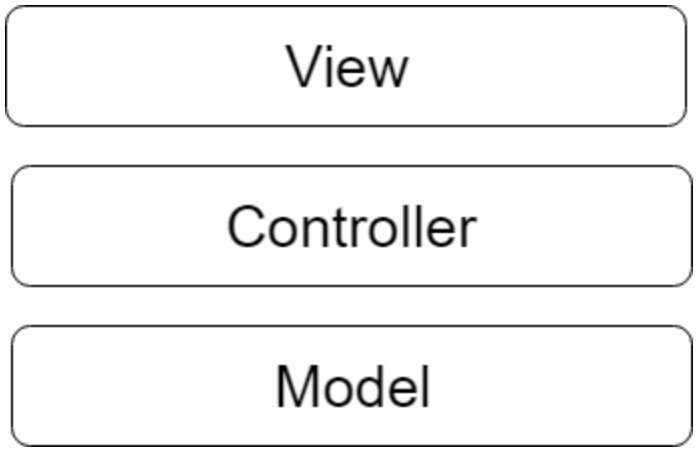
- Własne rozwiązanie
- Fluent Validation

Wykorzystanie fluent validation

- Biblioteka `FluentValidation.AspNetCore`
- AbstractValidator<T>

BAZY DANYCH

Model trójwarstwowy



Trójwarstwowy model – zalety

- Serwis do operacji CRUDowych
- Dowolne źródło danych (baza danych, pamięć)
- Wstrzykiwanie zależności
- Testowanie jednostkowe
- Różny model danych na różnych warstwach

Rodzaje baz danych

Relacyjne (SQL)

- SQL Server
- MySQL
- SQLite

Nierelacyjne (NoSQL)

- Mongo / Cosmos
- Tables
- GraphDB

BAZY DANYCH

Entity framework

EF – standardowy ORM

- Narzędzie ułatwiające obsługę mapowań z / do bazy danych
- Dwa możliwe podejścia
  - *Database-first (Reverse engineering)*
  - *Code-first*

Migracja danych

- Narzędzie ułatwiające obsługę mapowań z / do bazy danych
- Dwa możliwe podejścia
  - *Database-first (Reverse engineering)*
  - *Code-first*

Migracja danych

EF – użytkownicy z bazy danych

Zadania EF

- Uzupełnij operacje CRUDowe dla zdarzeń (edycja, skasowanie)

BAZY DANYCH

Dapper

Dapper

- Narzędzie ułatwiające obsługę mapowań z / do bazy danych
- Dwa możliwe podejścia
  - *Database-first (Reverse engineering)*
  - *Code-first*

Wykorzystanie Dappera

- Narzędzie ułatwiające obsługę mapowań z / do bazy danych
- Dwa możliwe podejścia
  - *Database-first (Reverse engineering)*
  - *Code-first*

Zadania dapper

- Dokończ implementację repozytorium korzystającego z Dappera
- Dodaj do kontrolera nową metodę, która zwróci uczestników jednego wydarzenia na podstawie jego identyfikatora

BAZY DANYCH

CosmosDB jako baza nierelacyjna

Kluczowe aspekty baz nierelacyjnych

- Brak ścisłych powiązań pomiędzy encjami
- Możliwość zmiany struktury danych
- Partycjonowanie

BUDOWANIE KONFIGURACJI

Konfiguracja aplikacji

- Spójna obsługa wielu różnych źródeł danych (w tym asynchroniczne)
- Dane wrażliwe
- Dane współdzielone pomiędzy aplikacje

BUDOWANIE KONFIGURACJI

Źródła „lokalne”

Dane z pamięci, linii komend oraz jsona

- AddInMemoryCollection
- AddCommandLine
- AddJsonFile

BUDOWANIE KONFIGURACJI

Źródła „zewnętrzne”

KeyVault

- Pewne klucze nie powinny być widoczne dla użytkowników
- Jeden punkt dostępu dla różnych aplikacji
- Dostępny w różnych usługach chmurowych

AppConfiguration

- Narzędzie do udostępniania współdzielonych

Zadania konfiguracja

- Opakuj konfiguracje (np. AzureServiceConfig) w klasę, która na podstawie IConfiguration odpowiednio przygotuje obsługę kolejek w AzureServiceBusHandler
- Dodaj konfiguracje z linii komend lub pamięci

CACHOWANIE

Kiedy czas dostępu jest kluczowy

- Trzymanie danych w zserializowanej formie w pamięci
- Optymalizacja czytania
- Konieczność czyszczenia „cache'a" po aktualizacji danych

CACHOWANIE

Cachowanie w pamięci

Cachowanie w pamięci

- Najszybsze
- Możliwe pominięcie serializacji
- Usługi rozproszone
- Ponowne uruchomienie aplikacji kasuje stan

Cachowanie

- Cachowanie w pamięci

CACHOWANIE

Redis cache

Kluczowe aspekty cachowania rozproszonego

- Brak sesyjności
- Możliwość wprowadzenia własnych kluczy
- Możliwość wykorzystania w różnych instancjach aplikacji

KOMUNIKACJA

Rodzaje komunikacji pomiędzy usługami

Synchroniczna

- Natychmiastowy wynik
- Potwierdzenie wykonania
- Obsługa błędu również po stronie „inicjującej”
- Zapytania http

Asynchroniczna

- Wynik otrzymany po pewnym czasie
- „Brak potwierdzenia wykonania”
- Obsługa poprzez kolejki

KOMUNIKACJA

Request over HTTP

Konfiguracja do obsługi zasobów zewnętrznych

- Dodanie klienta
- Skorzystanie z IHttpClientFactory

Podpowiadanie nowych wydarzeń

- W projekcie Events bazując na id użytkownika, podpowiedz zdarzenia w których on nie uczestniczy

KOMUNIKACJA

Request over AzureServiceBus

Integracja z Azure Service Bus

- Biblioteka Azure.Messaging.ServiceBus
- Serializacja oraz deserializacja zdarzeń
- Obsługa dwóch operacji na wiadomości
  - *Wysyłania*
  - *Odbierania*
- Obsługa zdarzeń przez wiele handlerów

Zadania końcowe

- Użytkownik po potwierdzeniu uczestnictwa w zdarzeniu powinien być aktualizowany również jako potwierdzony użytkownik serwisu
- Utworzenie obiektu EventDescription po utworzeniu zdarzenia

PYTANIA

Dziękuję!

- pawellabno1992@gmail.com
- plabno@sevillemore.com
- Paweł Łabno | LinkedIn