BOTTEGA IT minds

Program szkolenia:

Node.js - tworzenie zaawansowanych aplikacji przy użyciu NestJS

Informacje:

Nazwa:	**Node.js - tworzenie zaawansowanych aplikacji przy użyciu NestJS**
Kod:	**NestJS-nest**
Kategoria:	NestJS
Odbiorcy:	developerzy, architekci
Czas trwania:	3 dni
Forma:	30% wykłady, 70% warsztaty

Node.js jest technologią o coraz szerszym zastosowaniu, a liczba programistów korzystających z tego narzędzia rośnie bardzo szybko. Z tego względu Node.js jest wykorzystywany w coraz to różniejszych rodzajach systemów.

W dzisiejszych czasach jego zastosowanie wykracza poza inicjalne założenia twórców, dlatego często Node.js jest bazą dla złożonych systemów, które do tej pory kojarzyły się raczej z Javą i Springiem. Właśnie do obsługi tego typu aplikacji najlepiej sprawdza się NestJS.

Czego się nauczę?

- tworzyć testowalne API
- standardowych bloków konstrukcyjnych NestJS
- dobierać odpowiednie styl architektoniczny do problemu

Zalety szkolenia:

- trener, którzy zna NestJS od wewnątrz
- niestandardowe, bardziej zaawansowane podejście do pracy z Nestem
- niebanalne przykłady, które są najczęstszą przyczyną problemów i osłabienia utrzymywalnoście aplikacji

BOTTEGA
IT minds

Szczegółowy program:

1. Dlaczego NestJS?

- 1.1. Alternatywy dla NestJS i jego geneza
- 1.2. Narzędzia do obsługi zapytań HTTP
- 1.3. System wstrzykiwania zależności
- 1.4. Moduły

2. Architektura warstwowa

- 2.1. CRUD
- 2.2. Repozytoria
- 2.3. Walidacja danych
- 2.4. Dostęp do zasobów

3. Architektura heksagonalna

- 3.1. Główne porty
- 3.2. Logika biznesowa
- 3.3. Porty drugorzędne
- 3.4. Podłączanie adapterów do portów
- 3.5. Podmienianie adapterów przy pomocy dynamicznych modułów

4. Modelowanie logiki biznesowej

- 4.1. Różnice pomiędzy modelem, encją i DTO
- 4.2. Wyznaczanie niezmienników i tworzenie agregatów

5. CQRS

- 5.1. Biblioteka @nestjs/cqrs
- 5.2. Komendy i zapytania
- 5.3. Zdarzenia i ich obsługa

BOTTEGA IT minds

5.4. Model odczytowy

6. Stabilność kodu

6.1. Projektowanie pod testowalność

6.2. Testy jednostkowe

6.3. Testy integracyjne

6.4. Testy architektury

7. Na co trzeba uważać?

7.1. Zależności cykliczne

7.2. Funkcje pomocnicze wspierające złe projektowanie

8. Mikroserwisy

8.1. Uruchamianie NestJS w trybie mikroserwisowym

8.2. Komunikacja pomiędzy mikroserwisami