BO·TT·EGA
IT minds

Program szkolenia:

# Nowoczesna architektura aplikacji Web oparta o Node.js

## Informacje:

| | |
|---|---|
| **Nazwa:** | **Nowoczesna architektura aplikacji Web oparta o Node.js** |
| **Kod:** | **modern-Node** |
| **Kategoria:** | Node.js |
| **Odbiorcy:** | architekci, developerzy |
| **Czas trwania:** | 2-3 dni |
| **Forma:** | 20% wykłady / 80% warsztaty |

Node.js zapewnia dużo większą swobodę w porównaniu do tradycyjnych ekosystemów programistycznych klasy enterprise. Chociaż niektórzy programiści próbują przekształcić Node.js w platformę bazującą na ciężkich frameworkach, istnieje alternatywne podejście.

Na tym szkoleniu nauczysz się budować wysoce komponowalne, bezpiecznie typowane i testowalne systemy bez tradycyjnej magii frameworkowej. Jeżeli chcesz oduczyć się używania:

- kontenerów Dependency Injection
- programowania z klasami
- ORMów
- architektur zorientowanych na technologie
- magicznych konwencji
- łupieżu adnotacyjnego
- new Date() rozsianego po kodzie
- ręcznie pisanych DTO
- frameworków do mockowania importów
- refleksji i metaprogramowania

# To niniejsze szkolenie jest dla Ciebie.

Alternatywą do ciężkich frameworków nie jest Dziki Zachód programowania ani budowanie własnego frameworka. Najlepszą alternatywą jest odkrycie prawdziwej natury JS/TS, wykorzystanie tylko dobrych części tych języków, tak aby nigdy więcej nie dać się uwięzić w kolejnym ciężkim frameworku.

# Czego si naucz?

- tworzyć testowalne i bezpiecznie typowane API połączone z bazą SQL
- ewoluować strukturę aplikacji na podstawie informacji zwrotnej od kodu
- dobierać komponenty architektoniczne do problemu
- tworzyć aplikacje poprzez kompozycję małych bibliotek i funkcji (the Unix Way)

## Zalety szkolenia:

- Pokazuje ponadczasowe techniki, które przeżyją kolejny framework

- Uczy programować w języku, a nie we frameworku
- Obala mity i pokazuje jak usuwać sztuczną złożoność

BOTTEGA IT minds

# Szczegółowy program:

## 1. Zasady i techniki inżynierskie

- 1.1. Minimalne zależności
- 1.2. Funkcje zamiast klas
- 1.3. Parsuj zamiast walidować
- 1.4. Prawo Postela
- 1.5. Domyślny tryb in-memory
- 1.6. Architektura zorientowana na możliwości biznesowe
- 1.7. Kompozycja zamiast konwencji
- 1.8. bezpieczeństwo typów E2E
- 1.9. Programowanie w języku, zamiast we frameworku
- 1.10. Development sterowany feature flagami

## 2. Komponenty architektoniczne

- 2.1. Pipes and filters (express.js middleware, handlers)
- 2.2. Porty i adaptery
- 2.3. Routery i kontrolery
- 2.4. Serwisy aplikacyjne
- 2.5. Repozytoria
- 2.6. Error handlers
- 2.7. Typy domenowe i tiny types
- 2.8. Parsery wejściowe i wyjściowe
- 2.9. DTO za darmo
- 2.10. Composition root
- 2.11. Model do zapisu a model do odczytu

BOTTEGA IT minds

2.12. Zewnętrzny model do odczytu (model widoku) a wewnętrzny model do odczytu

2.13. Zasięg widoczności aplikacyjny i per zapytanie

## 3. Strategie testowania

3.1. Testy jednostkowe zachowania (mocha)

3.2. Testy integracyjne (mocha)

3.3. Testy komponentowe (supertest)

3.4. Testowanie bez mocków

3.5. Testowanie trudnych zależności (czas, generowanie id)

3.6. Strategie czyszczenia danych testowych

## 4. Integracja z bazą SQL

4.1. Migracje bazodanowe

4.2. Generowanie typów ze schemy bazodanowej

4.3. Weryfikacja kontraktu repozytoriów

4.4. SQL query builder

4.5. Obsługa transakcji bazodanowych

## 5. Bezpieczeństwo typów

5.1. Bezpiecznie typowana konfiguracja (zod)

5.2. Bezpiecznie typowane zapytania bazodanowe (kysely)

5.3. Bezpiecznie typowane wejście i wyjście (zod)

5.4. Bezpiecznie typowane Dependency Injection

5.5. Bezpiecznie typowane ścieżki do API (static-path)