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:	developerzy, architekci
Czas trwania:	3 dni
Forma:	20% wykłady / 80% warsztaty

Mikrousługi, chmura, ciągłe wdrażanie oprogramowania na produkcję mocno wpłynęły na to jak współczesne zespoły programistyczne budują systemy informatyczne.

W tym nowym świecie mamy coraz większe zapotrzebowanie na bardzo wydajne serwisy, które w efektywny sposób wykorzystują zasoby serwera. To właśnie w tym kontekście najlepiej sprawdza się Node.js.

Czego si naucz?

- tworzyć testowane API i aplikacje webowe połączone z bazą MongoDB
- ewoluować strukturę aplikacji na podstawie informacji zwrotnej od kodu
- dobierać styl architektoniczny do problemu
- tworzyć aplikacje poprzez kompozycję małych bibliotek i funkcji (the Unix Way)

Zalety szkolenia:

- Pokazuje aplikacje Node.js w szerszym kontekście
- Kładzie nacisk na czysty kod i architekturę, a nie na API frameworków i bibliotek
- Prezentuje rozwiązania na bazie których można budować rzeczywiste aplikacje
- Obala mity i pokazuje jak usuwać sztuczną złożoność

BOTTEGA IT minds

Szczegółowy program:

1. Express.js

- 1.1. routes
- 1.2. middleware
- 1.3. obsługa błędów
- 1.4. silniki do szablonów
- 1.5. połączenie z MongoDB

2. REST

- 2.1. Zasoby i reprezentacje
- 2.2. Content Negotiation
- 2.3. Zasoby kolekcji
- 2.4. HATEOAS - hipermedia
- 2.5. HTML as media type

3. Architektura warstwowa

- 3.1. Wstrzykiwanie zależności (Dependency Injection) bez frameworków
- 3.2. Asynchroniczne zależności
- 3.3. CRUD
- 3.4. kontrolery
- 3.5. serwisy
- 3.6. repozytoria
- 3.7. logika domenowa
- 3.8. walidacja danych

4. Architektura oparta o Event Sourcing i CQRS

- 4.1. agregaty, komendy i zdarzenia

BOTTEGA IT minds

4.2. modelowanie niezmienników

4.3. node-eventstore (MongoDB, in-memory)

4.4. command handlers

4.5. publikowanie zdarzeń po HTTP

5. Architektura Pipes and Filters z użyciem strumieni

5.1. typy strumieni (readable, writable, transform)

5.2. strumienie z core Node.js (http, fs, zlib)

5.3. biblioteki 3rd party (through2, split2)

5.4. architektura Pipes and Filters w kontekście strumieni

6. Testowalność

6.1. projektowanie pod testowalność

6.2. "ciężkie" zależności

6.3. Londyńska szkoła testowania jednostkowego

6.4. Chicagowska szkoła testowania jednostkowego

6.5. Testowanie komponentowe

6.6. narzędzia: mocha, tape, supertest

7. Refaktoring Legacy Node.js

7.1. znajdowanie "szwów" w kodzie

7.2. dodawanie tzw. characterization tests

7.3. używanie narzędzi porycia kodu testami do znajdowania martwego kodu

7.4. brzydkie zapachy w kodzie

7.5. automatyczny refaktoring z użyciem IDE