BOTTEGA IT minds

Program szkolenia:

Nowoczesna architektura aplikacji web oparta o Node.js - Microservices, REST, noSQL

Informacje:

Nazwa:	**Nowoczesna architektura aplikacji web oparta o Node.js - Microservices, REST, noSQL**
Kod:	**arch-ms-modern-Node**
Kategoria:	Microservices
Grupa docelowa:	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 jest coraz większe zapotrzebowanie na bardzo szybkie i wydajne serwisy które w efektywny sposób wykorzystują zasoby serwera. To właśnie w tym kontekście najlepiej sprawdza się node.js.

Podczas szkolenia zdobędziesz praktyczne doświadczenie jak budować mikrousługi z wykorzystaniem node.js. Pod koniec zajęć będziesz w stanie lepiej odpowidzieć na pytanie: jak efektywnie tworzyć mikrousługi w node.js w moim kolejnym projekcie.

Czego się nauczysz podczas praktycznych ćwiczeń i dyskusji grupowej:

- Jak zrozumienie działania pętli zdarzeń wpływa na to jak myślimy o naszych programach
- Jak programowanie z this i bez this wpływa na czytelność Twojego kodu
- Jak tworzyć testowalne API i aplikacje webowe w express.js (routing, middleware, obsługa błędów, komunikacja z bazą danych)
- Jak testować i debugować aplikacje node.js (mocha, supertest, siege)
- Jak nadać strukturę nietrywialnym aplikacjom node.js z użyciem wzorca Dependency Injection (bez frameworków)
- Jak refaktorować kod asynchroniczny z callbacków na Promisy
- Jak budować aplikacje i zarządzać zależnościami w npm
- Jak budować potoki wdrożeń dla aplikacji node.js i wdrażać je na produkcję (na przykładzie Heroku)
- Jak tworzyć interfejs użytkownika dla mikrousług z użyciem wzorca Backend For Frontend (różne opcje integracji serwisów: client side, edge side i server side include)
- Jak radzić sobie z problemami sieciowymi i pisać stabilne mikrousługi (timeout, retry, cachowanie, circuit breaker, prawo Postela w praktyce)
- Jak zabezpieczyć kto korzysta z naszej mikrousługi w użyciem basic auth i JWT (JSON Web Token)
- Jak orkiestracja i choreografia wpływają na mikro i makro architekturę

- Jak odchudzić aplikacje node.js i przerzucić część odpowiedzialności na infrastrukturę (lekkie podejście do logowania, metryk, cross-app tracing, rejestracji i odkrywania)

Co robimy podczas warsztatów?

Budujemy księgarnię internetową składającą się z kilku serwisów. Tworzymy serwis backendowy (w express.js), który ma swoją bazę danych (mongo) oraz serwis który agreguje inne serwisy po stronie klienta lub serwera (express.js + hogan.js).
W czasie ćwiczeń korzystamy z wybranych elementów Domain Driven Design (bounded context, repository pattern) oraz uczymy się pisać czysty kod asynchroniczny w node.js. Kładziemy duży nacisk na testowalność kodu.
Automatyzujemy powtarzalne czynności tj. proces budowania i wdrożenia na środowiska testowe i produkcyjne.
Żeby lepiej zrozumieć omawiane zagadnienia robimy też kilka mniejszych pobocznych ćwiczeń.

Zalety szkolenia:

- Zastosowanie Bounded Context z DDD
- Integracja technologii i całościowe podejście
- Dobór klasy rozwiązania do klasy problemu

BOTTEGA IT minds

Szczegółowy program:

1. Myślenie w Javascript

- 1.1. programowanie synchroniczne i asynchroniczne
- 1.2. pętla zdarzeń i wzorzec reaktor
- 1.3. JS w przeglądarce i poza przeglądarką
- 1.4. programowanie z this'em i bez this'a
- 1.5. pragmatyczne elementy programowania funkcyjnego
- 1.6. async control flow: callbacks
- 1.7. async control flow: promises

2. node.js podstawy

- 2.1. filozofia i model koncepcyjny
- 2.2. modularny kod z CommonJS
- 2.3. EventEmitter: wzorzec obserwator
- 2.4. zarządzanie zależnościami w npm
- 2.5. budowanie aplikacji w npm

3. Aplikacje webowe w node.js

- 3.1. express.js: routing
- 3.2. express.js: obsługa błędów
- 3.3. wzorzec middleware: cross cutting concerns
- 3.4. silnik szablonów z rodziny mustache: hogan.js
- 3.5. interakcja z bazą danych na przykładzie MongoDB i wzorca Repository
- 3.6. zasięg widoczności obiektów: request scope i application scope

4. Testowanie aplikacji w node.js

- 4.1. wzorzec Dependency Injection

BOTTEGA IT minds

4.2. testowanie całej aplikacji z użyciem supertest

4.3. testy jednostkowe z mocha i assert

4.4. tworzenie stubów na potrzeby testów

4.5. pisanie czystego i testowalnego kodu w JS

5. Wdrażanie aplikacji node.js na produkcję

5.1. 12factor: przygotowanie aplikacji do wdrożenia

5.2. Continuous Integration/Delivery/Deployment

5.3. budowanie potoku wdrożeń od A do Z dla aplikacji node.js

5.4. pragmatyczny wybór narzędzi do CI/CD

5.5. hostowanie aplikacji node.js na Heroku

5.6. wzorce wdrażania aplikacji na produkcję

5.7. infrastruktura jako kod

5.8. odchudzanie aplikacji i przenoszenie odpowiedzialności na infrastrukturę: logowanie, metryki, cross-app tracing, rejestracja usług, odkrywanie usług

6. node.js w architekturze mikrousług

6.1. określanie granicy usług wg. Bounded Context

6.2. budowanie REST API

6.3. ewolucja pojedynczych mikrousług: wzorzec Tolerant Reader

6.4. budowanie UI dla mikrousług: wzorzec Backends For Frontends

6.5. Ewolucyjna architektura z Self-Contained Systems

6.6. opcje integracji mikroserwisów: Client Side, Edge Side i Server Side

6.7. stabilny ekosystem mikrousług: retry, caching, circuit breaker

6.8. kontrolowanie dostępu do mikrousług z JSON Web Token (JWT)

6.9. orkiestracja i choreografia - różne podejścia do modelowania procesów biznesowych

6.10. wzorzec Publish/Subscribe z użyciem IronMQ