BO·TT·EGA
IT minds

Program szkolenia:

Architektura Oprogramowania dla Developerów Frontend

Informacje:

Nazwa:	**Architektura Oprogramowania dla Developerów Frontend**
Kod:	**architecture-fe**
Kategoria:	JS i Front-end
Odbiorcy:	architekci, developerzy, liderzy zespołów
Czas trwania:	3-4 dni
Forma:	33% warsztaty, 33% praca w grupie, 33% ćwiczenia

Szkolenie przeznaczone jest dla doświadczonych programistów frontend, którzy chcą i/lub potrzebują "wyjść poza ramy" swojego frameworka i języka programowania i zgłębić modele, style, wzorce i pryncypia architektury oprogramowania. Poruszane zagadnienia albo dotyczą stricte frontentu, albo są uniwersalne, albo których znajomość znacząco usprawni komunikację między developerami frontendowymi a backendowcami i architektami.

Program szkolenia jest ogólną ramą - konkretne szkolenie poprzedzamy analizą przed-szkoleniową.

Zalety szkolenia:

- Myślenie architektoniczne (wysoko-poziomowo) zamiast myślenia kodem i frameworkami (nisko-poziomowo)
- Nacisk na rozumienie istoty wzorców i stylów architektonicznych
- Poszukiwanie zalet i wad każdego z rozwiązań, rozumienie trade-offów
- Dopasowywanie klasy rozwiązania do klasy problemu
- Nacisk na zrozumienie kontekstu biznesowego i jego wpływu na architekturę

BOTTEGA IT minds

Szczegółowy program:

1. Wstęp do Architektury

- 1.1. Drivery architektoniczne
- 1.2. Cele architektury
- 1.3. Architect jako rola

2. Dokumentowanie Architektury oprogramowania

- 2.1. ADR (Architecture Decision Records)
- 2.2. RFC (Request For Comments)
- 2.3. C4 Model
- 2.4. Event Storming / Context Mapping

3. Style i Wzorce Architektoniczne

- 3.1. Modularny Monolit / Modulith
  - 3.1.1. Nie-Modularny Monolit
- 3.2. Mikroserwisy
  - 3.2.1. Wzorce mikroserwisowe
  - 3.2.2. Rozproszony Monolit
  - 3.2.3. MikroFrontends
- 3.3. Micro Kernel
- 3.4. Hexagonal
- 3.5. CRUD
- 3.6. CQRS
- 3.7. Event Sourcing

4. Modularyzacja

- 4.1. Rodzaje Couplingu

BOTTEGA IT minds

- 4.2. Kohezja
- 4.3. Enkapsulacja
- 4.4. Anti-Corruption Layer
- 4.5. Kanoniczny Model Danych
- 4.6. Boskie klasy
- 4.7. Anemiczny model domeny
- 4.8. Odwracanie zależności
- 4.9. Prawo Demeter
- 4.10. Odwracanie kontroli
- 4.11. Zasada Hollywood

5. Zdolności

- 5.1. Reużywalność
- 5.2. Skalowalność
- 5.3. Odporność
- 5.4. Dostępność
- 5.5. Fault Tolerance
- 5.6. Testowalność
- 5.7. Wydajność
- 5.8. Spójność

6. Integracja (moduł opcjonalny)

- 6.1. Komendy, Zdarzenia, Kwerendy
- 6.2. REST
- 6.3. CQRS vs API Composition
- 6.4. Messaging
- 6.5. Kontrakty

BOTTEGA IT minds

- 6.5.1. Consumer-Driven Contracts
- 6.5.2. Contract Testing

7. Transakcje (moduł opcjonalny)

- 7.1. ACID
- 7.2. Wzorzec Saga
- 7.3. Orkiestracja vs Choreografia
- 7.4. Semantyka dostarczeń wiadomości
  - 7.4.1. At-most-once delivery
  - 7.4.2. At-least-once delivery
  - 7.4.3. Exactly-once delivery

8. Strategiczne DDD (wstęp)

- 8.1. Subdomeny
- 8.2. Konteksty ograniczone
- 8.3. Mapowanie kontekstów
- 8.4. Wszędobylski język
- 8.5. Heurystyki

9. DevOps

- 9.1. CI: git flow vs trunk-based
- 9.2. Infrastruktura
  - 9.2.1. Provisioning
- 9.3. Observability, Monitoring, Logging
- 9.4. (Distributed) Tracing
- 9.5. Metryki DORA

10. Architektura Mikro-Frontends

- 10.1. Micro-Frontend Architecture - w ujęciu strategicznym

BOTTEGA IT minds

- 10.2. Korzyści, Koszty, Ograniczenia
- 10.3. Różne implementacje
  - 10.3.1. The Strangler Pattern
  - 10.3.2. IFrames
  - 10.3.3. Webpack Module Federation
  - 10.3.4. Ng-elements / WebComponents
  - 10.3.5. Frameworks, Single-SPA

11. Wzorce Aplikacji Frontendowych

- 11.1. Monitoring
- 11.2. Realtime Operations
- 11.3. Optimistic Updates
- 11.4. Caching
- 11.5. Queries
- 11.6. Lazy Loading
- 11.7. Hot-Module Replacement
- 11.8. Time-travelling