Kod szkolenia: **J/ARCH**

Tytuł szkolenia: **Architektura systemów (Java i integracja)**

Dni: 5

Opis:

Adresaci szkolenia:

Szkolenie adresowane jest do osób, które chciałyby zapoznać się z praktycznymi aspektami tworzenia architektury. Dla osób, które chcą otworzyć przed sobą nowe możliwości w zakresie realizacji zadań związanych z wyższymi kompetencjami architekta lub pragną osiągnąć wyższą świadomość konsekwencji płynących z dobieranych rozwiązań, w celu podejmowania lepszych decyzji.

Szkolenie jest odpowiednie zarówno dla programistów jak i projektantów, analityków, ale także dla architektów chcących usystematyzować wiedzę i wymienić doświadczenia.

Cel szkolenia:

Celem szkolenia jest zdobycie wiedzy niezbędnej do tworzenia i weryfikacji architektury oraz umiejętności rozpatrywania potencjalnych rozwiązań z punktu widzenia parametrów systemowych. W ramach szkolenia uczestnicy poznają język UML, w zakresie modelowania architektury i umiejętności tworzenia modeli architektonicznych.

Szkolenie kładzie duży nacisk na osiągnięcie wysokiej świadomości konsekwencji związanych z doborem rozwiązań, technologii, wzorców i innych decyzji architektonicznych. W oparciu o tę świadomość ćwiczymy i budujemy umiejętność podejmowania i weryfikacji decyzji architektonicznych poruszając się w realiach nieklarownych wizji systemu i dużej ilości założeń architektonicznych. Omawiamy decyzje z punktu widzenia korzyści i wad, oraz omawiając sposoby weryfikacji zarówno decyzji jak i założeń.

Szerokim tematem szkolenia są również wzorce oraz modelowanie w UML, gdzie poznanie języka UML jest tylko środkiem, a jako cel wyznaczone jest nabycie umiejętności tworzenia modeli architektonicznych bazując na wymaganiach klienta.

Wymagania:

Szkolenie wprowadza do zagadnień architektury od podstaw. Zarówno w tematyce samej architektury, wzorców, technologii, jak i UML. W związku z tym szkolenie nie posiada żadnych wymagań wstępnych stawianych uczestnikom.

Parametry szkolenia:

4*8 lub 5*8 godzin (4*7 lub 5*7 godzin netto) wykładów i warsztatów (z wyraźną przewagą warsztatów). Proporcje warsztatów różnią się w zależności od aktualnej tematyki (wyższe przy UML, niższe przy technologiach poszczególnych warstw aplikacji).

Wielkość grupy: maks. 8-10 osób.

Program szkolenia:

1. Podstawy Architektury
    I. Czym jest architektura
        i. Architektura a projekt
        ii. Cele tworzenia architektury
    II. Kim jest architekt i jaką pełni rolę
        i. Kim jest architekt - różne poziomy
            A. Technolog
            B. Strateg
            C. Polityk
        ii. Co robi architekt
        iii. Potrzeba istnienia architekta a skala projektu
    III. Proces architektoniczny
    IV. Dokumentacja architektoniczna
    V. Zarządzanie ryzykiem
2. Parametry systemowe
    I. Czym są parametry systemowe
    II. Jak poprawnie definiować wymagania niefunkcjonalne
    III. Parametry systemowe
        i. Wygoda użytkownika (Usability)
        ii. Bezpieczeństwo (Security)
        iii. Wydajność (Performance)
            A. Przepustowość (Throughput)
            B. Czas odpowiedzi (Response Time)
            C. Czas reakcji (Responsivness)
        iv. Dostępność (Availability)
        v. Niezawodność (Reliability)
        vi. Skalowalność (Scalability)
        vii. Różne wymiary elastyczności systemu
            A. Rozszerzalność (Extensibility)
            B. Reużywalność (Reusability)
            C. Przenaszalność (Portablity)
            D. Elastyczność (Flexibility)
        viii. Realizowalność (Realizability)
        ix. Planowalność (Planability)
        x. Testowalność (Testability)
        xi. Utrzymanie (Maintainability)

xii. Serwisowalność (Serviceability)
xiii. Zarządzalność (Managebility)
IV. Wymiary systemu
i. Wymiary związane z infrastrukturą
A. Pojemność (Capacity)
B. Redundantność/Replikacja (Redundancy)
C. Modułowość (Modularity)
ii. Wpływ wymiarów na parametry systemu
iii. Inne wymiary systemu
A. Tolerancja (Tolerance)
B. Obciążenie (Workload)
C. Niejednorodność/Jednorodność (Homo/Heterogenity)
V. Priorytety parametrów systemu
i. Skąd wynikają priorytety?
ii. Problemy priorytetowania
3. Wzorce architektoniczne
I. Wprowadzenie do wzorców
i. Definicja wzorca
ii. Cechy i zalety wzorców
iii. Rodzaje wzorców
II. Wzorce architektoniczne
i. Problemy architektury komponentowej
ii. Wzorce podziału odpowiedzialności
A. MVC (Model View Controll)
B. Web-centric
C. Application-centric
D. Enterprise
a. Wymagania systemów Enterprise
b. Architektura Enterprise w JEE
E. Architektura wielowarstwowa (Layers Pattern)
a. Architektura wielowarstwowa w JEE
iii. Wzorce EAI (Enterprise Application Integration)
A. SOA (Service Oriented Architecture)
B. ESB (Enterprise Service Bus; Szyna Danych; Broker Integracyjny)
C. MOM (Message Oriented Middleware)
D. Microsercvices
a. Założenia
b. Monolit a Microservices
c. DB w monolicie a w Microservices
d. Dlaczego Microservices?
e. Problemy z microservicves
iv. Wzorce infrastruktury
A. Redundancja Ścieżek
B. Skalowanie pionowe
C. Skalowanie poziome (Replikacja)

D. Równoważenie obciążenia (Load Balancing)
E. Klastry (Clustering)
    a. HA (High Availability)
    b. Failover
F. Forward Proxy Cache

v. Wzorce blokowania zasobów
A. Blokowanie optymistyczne (Optimistic Lock)
B. Blokowanie pesymistyczne (Pessimistic Lock)
    a. Blokada domyślna (Implicit Lock)
    b. Blokowanie gruboziarniste (Coarse-Grained Lock)

vi. Inne wzorce architektoniczne
A. Dependency Injection
B. Dependency Inversion
C. Stable Dependency Principle
D. Szablon wzorców POSA
E. Szablon wzorców PEAA
F. Szablon wzorców Core J2ee
G. Szablon wzorców EAI (wzorce JMS)

4. Prototypowanie
I. Po co prototypować
II. Prototyp Proof of Concept
III. Prototyp ewolucyjny
IV. Antywzorzec Lava Flow

5. Metodyki wytwarzania oprogramowania a architektura
I. Metodyka kaskadowa
II. USDP (UP) – Unified Software Development Process
    i. Założenia
    ii. Wymiary
    iii. Fazy
        A. Rozpoczęcie (Inception)
        B. Opracowanie (Elaboration)
        C. Budowa (Construction)
        D. Wdrożenie (Transition)
III. RUP – Rational Unified Process
IV. SynTone Architecture Metodology
V. Metodyki Agile
    i. Extreme Programming (XP)
    ii. Scrum
VI. Podejście hybrydowe

6. Architektura warstwy klienta i prezentacji
I. Podział klientów
    i. Klient gruby
    ii. Klient Cienki
        A. RIA
II. Przechowywanie sesji
III. Technologie klienta grubego

i. Swing
ii. SWT
iii. RCP

IV. Klient gruby zanurzony w kliencie cienkim
i. Applet
ii. Java Web Start
iii. Java FX

V. Technologie klienta cienkiego
i. HTML Statyczny
ii. HTML Dynamiczny
A. Servlety + JSP + JSTL
B. Portlety
C. JSF (Java Server Faces)
D. Ajax
a. Java Script
b. Prototype
c. Ajax4JSF
d. PrimeFaces
e. GWT
E. Wsparcie JavaScript
a. JSON
b. jQuery
F. SPA/SPI
a. Przykładowe frmeworki: Angular, Ember JS
b. Przykładowe biblioteki JavaScript: React, Vue
G. WebSocket
a. Protokół full-duplex
b. Użycie w przeglądarce
c. Użycie w JEE (@ServerEndpoint)
d. Kiedy stosować WebSocket

7. Architektura warstwy biznesowej
I. Przetwarzanie rozproszone
II. Komunikacja zdalna a lokalna
III. Optymalizacja komunikacji sieciowej
IV. Protokoły komunikacyjne
i. CORA i IIOP
ii. Web Services SOAP
A. SOAP
B. WSDL
C. Repozytoria usług: UDDI, ebXML
D. Specyfikacje JEE
a. JAX-P (Java API for XML Processing)
b. SAAJ (SOAP with Attachments API for Java)
c. JAX-B (Java API for XML Binding)
d. JAX-R (Java API for XML Registries)
e. JAX-WS (Java API for XML Web Services)

E. Web Service Orchestration (wsparcie procesów biznesowych)
iii. Web Services REST
A. REST i wsparcie JAX-RS (od JEE6)
B. WADL i alternatywy (API Blueprint, Swagger)
C. HATEOAS
iv. GraphQL
A. Czym jest GraphQL
B. GraphQL vs REST
v. Sockets (własny protokół)
vi. RMI (Remote Method Invocation)
vii. EJB i RMI-IIOP
A. Rodzaje komponentów EJB
a. Sesyjne
1. Statefull
2. Stateless
3. Singleton (od EJB3.1 – JEE6)
B. MDB
C. Encyjne (do EJB2.x)
viii. Implementaca i komunikaca w EJB
ix. JNDI jako repozytorium EJB
x. Serwery aplikacji
A. Usługi serwera aplikacji
B. Kryteria wyboru serwera

8. Architektura warstwy integracji i zasobów
I. Technologie utrwalania danych
i. Bazy relacyjne
A. JDBC
B. Entity EJB
C. JDO (także bazy obiektowe i inne)
D. JPA
ii. LDAP (bazy hierarchiczne)
iii. NoQSL
iv. JCR – Java Content Repository (systemy CMS)
v. JCA – Java Connector Architecture (systemy EIS)
II. Komunikacja asynchroniczna
i. JMS (Java Message Service)
A. Topic
B. Queue
ii. MDB EJB (Message Driven Bean)
III. Systemy „Legacy”
IV. Screen Scrapping

9. Wzorce projektowe a architektura (tylko wersja 5cio dniowa)
I. Jak wzorce projektowe mogą wpływać na architekturę
II. Wybrane wzorce Core J2EE
i. Warstwa Prezentacji
A. Intercepting Filter (flexibility)

B. Context Object (maintenance, flexibility)
C. Composite View (flexibility)
D. Service To Worker (flexibility)

ii. Warstwa Biznesowa
A. Business Delegate (maintenance)
B. Service Locator (maintenance)
C. Session Façade (performance, flexibility)
D. Transfer Object (performance)
E. Value List Handler (performance, scalability)
F. Application Service (flexibility, maintenance)
G. Business Object (flexibility, maintenance)

iii. Warstwa Integracji
A. DAO (flexibility)
B. Domain Store (flexibility)

III. Wybrane wzorce GOF
i. Factory Method (flexibility)
ii. Abstract Factory (reliability, flexibility)
iii. Builder (reliability, flexibility)
iv. Prototype (performance)
v. Singleton (performance)
vi. Façade (performance, flexibility)
vii. Command (flexibility)
viii. Strategy (flexibility)
ix. Adapter (flexibility)
x. Mediator (flexibility)
xi. Observer (performance, flexibility)
xii. Template Method (reliability, flexibility)
xiii. Bridge (reliability, flexibility)
xiv. Memento (reliability)
xv. Visitor (flexibility)
xvi. Flyweight (memory performance)
xvii. Chain of responsibility (flexibility)
xviii. Proxy (flexibility, performance)
xix. Decorator (flexibility)

10. Wprowadzenie do UML
I. Czym jest modelowanie
II. Czym jest a czym nie jest UML
III. Rozwój UML
IV. Podstawowe elementy UML
i. Podstawowe kwalifikatory
A. Klasa (Class)
B. Interfejs (Interface)
C. Obiekt (Object)
D. Aktor (Actor)
E. Przypadek Użycia (Use Case)
F. Komponent (Component)

G. Węzeł (Node)
ii. Relacje (Relationships)
A. Asocjacja (Association)
B. Asocjacja (Association)
C. Zależność (Dependency)
D. Realizacja (Realization)
iii. Diagramy (Diagrams)
iv. Komentarze (Note)
v. Mechanizmy rozszerzenia
A. Stereotypy (Stereotype)
B. Etykiety (Tagged Values)
C. Ograniczenia (Constraints)
V. Diagram a model UML
VI. Zastosowania UML
11. Modelowanie architektury w UML
I. Diagram komponentów (component diagram)
i. Komponent (component)
ii. Komponenty zagnieżdżone
iii. Interfejs (interface)
A. Interfejs wymagany (required interface)
B. Interfejs dostarczany (provided interface)
iv. Złączenie (assembly)
II. Diagram wdrożenia (deployment diagrram)
i. Węzeł (node)
ii. Łącze (communication path)
A. Łącze kierunkowe
B. Liczność łącza
12. Zaawansowane aspekty modelowania architektury w UML
I. Zaawansowane elementy diagramu komponentów (component diagram)
i. Porty
ii. Konektory
iii. Realizacja komponentu
II. Zaawansowane elementy diagramu wdrożenia (deployment diagram)
i. Instancyjne diagramy wdrożenia
ii. Niskopoziomowe diagramy wdrożenia
iii. Szablony architektoniczne
iv. Model wdrożenia na diagramach wdrożenia
A. Po do model wdrożenia
B. Artefakt
C. Stereotypy artefaktów
a. file
b. document
c. library
d. executable
e. script
f. source

D. Specyfikacja konfiguracji (deployment specification)
E. Relacje między artefaktami
    a. Kompozycji (composition)
    b. Zależności (dependency)
F. Instalacja artefaktów (deployment) deploy
G. Manifestacja (manifestation) manifest

v. Diagram pakietów (package diagram)
A. Pakiet
B. Zagnieżdżanie (nest, nesting)
C. Przestrzeń nazw
D. Importowanie (package import)
    a. import
    b. access
E. Łączenie (merge)
F. Diagramy pakietów i modelowanie warstw architektury
    a. Przecięcie warstw i poziomów

13. Przejście z architektury do projektu
I. Warstwy i komponenty a realizacja projektu
II. Warstwy i komponenty a model projektowy
III. Uwzględnienie ograniczeń architektury w projekcie
    i. Na modelu statycznym
    ii. Na modelu dynamicznym
14. Bezpieczeństwo
I. Mechanizmy bezpieczeństwa
    i. Uwierzytelnianie (Authentication)
    ii. Autoryzacja (Authorization)
    iii. Kontrola dostępu (Access Control)
    iv. Logowanie
    v. Audyt
    vi. Szyfrowanie danych
    vii. Szyfrowanie transmisji
    viii. Integralność i przywracanie danych (backup)
    ix. Certyfikaty
II. Bezpieczeństwo w Javie
    i. JAAS
    ii. Servlety
    iii. Spring ACEGI
    iv. EJB
III. Serwery SSO (Single Sign On)
IV. Zarządzanie bezpieczeństwem
V. Podstawowe rodzaje ataków
    i. DOS i DDOS
    ii. SQL Injection
    iii. Cross Site Scripting (XSS)
    iv. Cross Site Request Forgery
15. Transakcje

I. ACID i BASE
II. CAP Theorem
III. Poziomy izolacji
    i. SERIALIZABLE
    ii. REPETABLE_READ
    iii. READ_COMMITED
    iv. READ_UNCOMMITED
IV. Efekty uboczne obniżania poziomu izolacji
    i. Fantomy (Phantoms)
    ii. Niepowtarzalny odczyt (Unrepetable read)
    iii. Brudny odczyt (Dirty read)
V. Wpływ transakcji na system
VI. Transakcje rozproszone (JTA, 2PC)
VII. Transakcje kompensacyjne
    i. Na czym polega kompensacja
    ii. 3PC
VIII. Kontrola obciążenia systemu transakcjami
    i. Zasięg i rozmiar transakcji
    ii. Transakcje biznesowe a systemowe
    iii. Unikanie transakcji
        A. Kompensacja
        B. Memento
        C. Rezygnacja z transakcji biznesowych
            a. Konflikty zapisu danych
                1. Blokowanie optymistyczne
                2. Blokowanie pesymistyczne
        D. Buforowanie danych
IX. Transakcje a EJB
    i. Container Managed Transaction
        A. Required
        B. Requires new
        C. Mandatory
        D. Supports
        E. Not supported
        F. Never
    ii. Bean Managed Transaction
    iii. Client Managed Transaction
X. Transakcje długoterminowe w WebService

16. Weryfikacja i ocena architektury
I. Po co weryfikować?
II. Zespół weryfikujący
III. Techniki weryfikacji i oceny
IV. Proces weryfikacji
V. Raport z weryfikcji