**Dodatek**

# Zalecenia związane z jakością i wydajnością, dotyczące budowy każdej z warstw wielowarstwowego oprogramowania

Cel budowy wielowarstwowej – przydział następujących funkcji aplikacji do poszczególnych warstw oraz zalecenia dotyczące *poprawy wydajności, skalowalności i jakości (czyli poprawy abstrakcji kodu, wielożywalności, pielęgnowalności, zrozumiałości, niezawodności)*

1) **warstwa klienta**
   - 1.1) **struktura**: klienci aplikacji, aplety, aplikacje i inne elementy z graficznym interfejsem użytkownika
   - 1.2) **zadania**: interakcja z użytkownikiem, obsługa urządzeń i prezentacja interfejsu użytkownika
   - 1.3) **zalecenia**: przeprowadzanie walidacji, co poprawia wydajność aplikacji dzięki obniżeniu ruchu w sieci.
2) **warstwa prezentacji**
   - 2.1) **struktura:** strony JSP, ASP, PHP itd servlety i inne elementy interfejsu użytkownika
   - 2.2) **zadania**: logowanie, zarządzanie sesją, walidacja i formatowanie danych i dostarczanie danych za pomocą obsługi zdarzeń
   - 2.3) **zalecenia**: oddzielenie kodu prezentacyjnego od przetwarzania biznesowego (usług biznesowych),
     **szczegóły zaleceń**: brak dostępu do struktur danych warstwy biznesowej; brak dostępu do struktur danych warstwy prezentacji w innych warstwach; przekazywanie danych między warstwą prezentacji a warstwą biznesową i integracji za pomocą obiektu klasy pomocniczej (tzw. obiekt transferowy); uniemożliwienie wysyłania ponownie tego samego formularza uruchamiającego ponownie transakcję dotyczącą tych samych danych; ograniczenie dostępu do pewnych formularzy lub ich części w wyniku uwierzytelniania (np. przez logowanie) i autoryzacji; stosowanie szablonów formularzy; szyfrowanie danych wrażliwych przesyłanych między klientem a serwerem; w kodzie strony HTML, generowanej dynamicznie nie powinno być czystego kodu napisanego w języku np. Java (tzw. skryplet), ponieważ jest on udostępniony klientowi aplikacji;
3) **warstwa biznesowa**
   - 3.1) **struktura**: komponenty usług biznesowych np. EJB i inne obiekty biznesowe
   - 3.2) **zadania**: logika biznesowa, transakcje, obsługa danych i realizacja usług
   - 3.3) **zalecenia**: wydzielenie przetwarzania biznesowego w celu zwiększenia jego wielożywalności, wprowadzenie komponentów sesyjnych wspierających wielożywalność usług biznesowych, zarządzanie transakcjami realizują komponenty sesyjne lub infrastruktura wewnętrzna oprogramowania (np. kontenery),
     **szczegóły zaleceń**: stosowanie jednego komponentu sesyjnego (fasada zdalna warstwy biznesowej) do obsługi wielu przypadków użycia, udostępnianych grupie klientów; obiekty zdalne wolno stosować tylko jako fasady usług; wyszukiwanie usług (obsługujących poszczególne przypadki użycia) jest realizowane przez wyspecjalizowane obiekty warstwy biznesowej; wyjątki klas warstwy biznesowej są obsługiwane wewnątrz warstwy biznesowej i są zamieniane na wyjątki warstwy prezentacji w razie konieczności monitorowania błędów przez warstwę prezentacji; do obsługi wzorca architektury Model-View-Controler model danych udostępniany warstwie prezentacji jest niezależny od modelu obiektowego lub modelu danych używanego do mapowania do modelu relacyjnego; metody obiektów zdalnych nie blokują się w trakcie przetwarzania; stosuje się fasady usług przechowujące

stan sesji tylko wtedy, gdy ważny jest stan sesji, a bezstanowe tylko wtedy, gdy stan sesji jest nieistotny; przetwarzanie danych w warstwie biznesowej za pomocą obiektowego modelu danych np. niezależnie od silnika bazy danych; zastosowanie technologii mapowania obiektowego modelu danych warstwy biznesowej na relacyjny (ORM) lub obiektowy w celu utrwalania danych aplikacji odpowiednio w relacyjnych lub obiektowych bazach danych; przy mapowaniu modelu relacyjnego na obiekty warstwy biznesowej nie należy używać obiektów zdalnych lub komponentów o rozbudowanej funkcjonalności (zarządzanie sesją i transakcjami);

4) **warstwa integracji**
   4.1) **struktura**: JMS, JDBC, konektory i połączenia z systemami zewnętrznymi
   4.2) **zadania:** adaptery zasobów, systemy zewnętrzne, mechanizmy zasobów, przepływ sterowania
   4.3) **zalecenia**: wydzielenie kodu dostępu do danych poprawia wieloużywalność, abstrakcję i zmniejszenie zależności z innymi funkcjami systemu; stosowanie puli połączeń czyli współdzielenie połączeń z bazą danych przez wielu klientów aplikacji; w złożonych projektach umieszczanie kodu dostępu do danych logicznie i fizycznie bliżej źródła danych;
5) **warstwa zasobów**
   **5.1)** **struktura:** bazy danych, systemy zewnętrzne i pozostałe zasoby
   **5.2)** **zadania:** obsługa zasobów i danych, usługi zewnętrzne.
   **5.3)** **Zalecenia:** używanie schematu bazy danych dopasowanego do przypadków użycia realizowanych przez aplikację – dobra praktyka to korzystanie z technologii np ORM