Architektura oprogramowania (kod: ARCHITEKTURA)

Opis i cel szkolenia

Architektura - to te decyzje, które później najtrudniej zmienić. Z jakich części składa się system? jak te części się komunikują, gdzie system działa i w jaki sposób spełnia wymagania dotyczące dostępności, wydajności czy bezpieczeństwa? Na szkoleniu uczestnicy przechodzą tę drogę na przykładowym systemie — od wymagań jakościowych, przez wybór stylu architektonicznego (monolit modularny, mikroserwisy, architektura zdarzeniowa) i sposobu integracji, po wdrożenie, dokumentację w UML i modelu C4 oraz przegląd gotowego projektu.

Po szkoleniu:

— zrozumiesz, skąd biorą się decyzje architektoniczne i jak zamienić ogólnikowe „system ma być szybki i niezawodny" na konkretne, sprawdzalne wymagania,
— poznasz najważniejsze style i wzorce architektoniczne — od architektury warstwowej po mikroserwisy i przetwarzanie zdarzeń — razem z ich kosztami i ryzykami,
— dobierzesz sposób komunikacji, przechowywania danych i wdrożenia do potrzeb konkretnego systemu,
— opiszesz architekturę tak, żeby zrozumieli ją programiści, analitycy i zamawiający: diagramy UML i C4 oraz zapis kluczowych decyzji,
— przeprowadzisz przegląd architektury i wskażesz jej słabe punkty — także w systemie, który odbierasz od wykonawcy.

Zajęcia mają charakter warsztatowy: pracujemy na wymaganiach, diagramach i decyzjach, nie na kodzie. Osobny blok poświęcamy AI — zarówno narzędziom, które wspierają pracę architekta i analityka, jak i temu, co zmienia się w architekturze systemu korzystającego z modelu językowego.

To szkolenie dotyczy architektury całego systemu. Jeśli szukasz projektowania na poziomie kodu — zasad SOLID, klasycznych wzorców projektowych i refaktoryzacji — wybierz szkolenie Wzorce projektowe w procesie tworzenia oprogramowania.

Szkolenie jest dla osób, które projektują, rozwijają, opisują lub odbierają systemy IT: programistów i liderów technicznych wchodzących w rolę architekta, początkujących architektów i projektantów, analityków systemowych i biznesowych, testerów i inżynierów DevOps, a także osób, które zamawiają systemy u zewnętrznych wykonawców i odpowiadają za ich odbiór. Dla grup na zamówienie dostosowujemy akcenty programu — więcej wymagań i modelowania dla zespołów analityków, więcej decyzji technicznych dla programistów i architektów — a program można rozszerzyć o dodatkowe dni warsztatu projektowego.

Czas trwania

3 dni, 9:00 - 17:00

Program

1. Architektura oprogramowania — czym jest i kto za nią odpowiada
   — Architektura systemu a projekt szczegółowy i wzorce projektowe na poziomie kodu
   — Decyzje architektoniczne: te, które najtrudniej i najdrożej zmienić później

Zapytaj o szczegóły

tel. 22 63 64 164
akademia@alx.pl

Najbliższe terminy

2026-12-21 (Zdalnie)
2026-12-21 (Warszawa)
2027-04-14 (Zdalnie)
2027-04-14 (Warszawa)
2027-06-09 (Zdalnie)
2027-06-09 (Warszawa)

- — Rola architekta w projekcie i w organizacji; współpraca z analitykiem, zespołem programistów i zamawiającym
- — Ograniczenia biznesowe, prawne, technologiczne i organizacyjne (prawo Conwaya)
- — Architektura jako zarządzanie ryzykiem

2. Wymagania jakościowe jako źródło decyzji architektonicznych
   - — Najważniejsze cechy jakościowe: dostępność, niezawodność, wydajność, skalowalność, bezpieczeństwo, utrzymywalność, testowalność, obserwowalność
   - — Scenariusze jakościowe: jak zamienić „system ma być szybki” w mierzalne wymaganie
   - — Konflikty między cechami jakościowymi i świadome kompromisy
   - — Zbieranie i opisywanie wymagań pozafunkcjonalnych — wspólne zadanie analityka i architekta
3. Style i wzorce architektoniczne
   - — Monolit, monolit modularny, architektura warstwowa (N-tier)
   - — Wzorce warstwy prezentacji: MVC i pokrewne
   - — Architektura heksagonalna (porty i adaptery): oddzielenie logiki biznesowej od infrastruktury
   - — Architektura usługowa (SOA), szyna integracyjna (ESB) i integracja aplikacji w przedsiębiorstwie
   - — Mikroserwisy
   - — Architektura sterowana zdarzeniami, potoki i filtry
   - — Równoważenie obciążenia, skalowanie pionowe i poziome
   - — Jak wybrać styl architektoniczny: kryteria i typowe błędy
4. Mikroserwisy w praktyce
   - — Wyznaczanie granic usług: kontekst ograniczony (bounded context) i elementy strategicznego DDD
   - — Dane w systemie rozproszonym: baza danych per usługa, różne rodzaje baz w jednym systemie (polyglot persistence)
   - — ACID a BASE, spójność ostateczna, transakcje rozproszone i sagi
   - — CQRS i Event Sourcing — kiedy mają sens, a kiedy tylko komplikują system
   - — DevOps i CI/CD jako warunek działania mikroserwisów
   - — Kiedy mikroserwisy nie są dobrym wyborem
5. Warstwy systemu: klient, usługi, dane i integracja
   - — Warstwa klienta: aplikacje grube i cienkie, aplikacje jednostronicowe (SPA), aplikacje mobilne, komunikacja w czasie rzeczywistym (WebSocket)
   - — Warstwa usług: REST, gRPC, GraphQL, SOAP — porównanie i kryteria wyboru
   - — Projektowanie i wersjonowanie API
   - — Warstwa danych: bazy relacyjne, NoSQL, hurtownie danych i przetwarzanie dużych wolumenów — kryteria wyboru
   - — Komunikacja synchroniczna i asynchroniczna: kolejki i brokery komunikatów (np. RabbitMQ, Kafka), pamięć podręczna (np. Redis)
6. Wdrożenie, infrastruktura, bezpieczeństwo i koszty
   - — Własna infrastruktura, chmura (IaaS, PaaS, SaaS, serverless) i rozwiązania hybrydowe
   - — Konteneryzacja i orkiestracja (Docker, Kubernetes) z perspektywy architekta
   - — Wysoka dostępność, odtwarzanie po awarii, kopie zapasowe
   - — Monitorowanie i obserwowalność systemu
   - — Bezpieczeństwo w architekturze: uwierzytelnianie i autoryzacja, komunikacja między usługami, ochrona danych
   - — Koszt jako cecha architektury: całkowity koszt utrzymania, koszty chmury

Zapytaj o szczegóły

tel. 22 63 64 164
akademia@alx.pl

Najbliższe terminy

2026-12-21 (Zdalnie)
2026-12-21 (Warszawa)
2027-04-14 (Zdalnie)
2027-04-14 (Warszawa)
2027-06-09 (Zdalnie)
2027-06-09 (Warszawa)

7. Dokumentowanie architektury
   — Widoki architektury: logiczny, procesów, wdrożeniowy — i dla kogo jest każdy z nich
   — UML w dokumentacji architektury: diagramy komponentów, wdrożenia i sekwencji
   — Model C4: kontekst, kontenery, komponenty
   — Zapisywanie decyzji: Architecture Decision Records (ADR)
   — Szablony dokumentacji architektury (np. arc42)
   — Dokumentacja „tyle, ile trzeba”: jak utrzymać ją aktualną, diagramy jako kod
8. Ocena i rozwój architektury
   — Przegląd architektury: analiza scenariuszy, ryzyk i kompromisów
   — Dług techniczny i architektoniczny
   — Modernizacja systemów zastanych, m.in. stopniowe wydzielanie usług z monolitu
   — Architektura w projekcie zwinnym: kiedy decydować, a kiedy świadomie odłożyć decyzję
   — Odbiór systemu od wykonawcy: na co patrzeć w dokumentacji i w architekturze
9. AI a architektura oprogramowania
   — Narzędzia AI w pracy architekta i analityka: analiza wymagań, warianty rozwiązań, szkice diagramów i decyzji, przegląd dokumentacji
   — Weryfikacja wyników generowanych przez AI i granice zaufania
   — System z komponentem AI: integracja z modelem językowym przez API, wykorzystanie danych firmowych (RAG), koszty, czas odpowiedzi, poufność danych, model w chmurze czy lokalnie
10. Warsztat podsumowujący
   — Od wymagań do opisanej architektury przykładowego systemu: cechy jakościowe, wybór stylu, diagramy C4 i UML, kluczowe decyzje (ADR)
   — Prezentacja rozwiązań i wspólny przegląd architektury

Przeznaczenie i wymagania

Szkolenie jest przeznaczone przede wszystkim dla:
- programistów i liderów technicznych, którzy wchodzą w rolę architekta lub projektanta systemu,
- początkujących architektów oprogramowania i rozwiązań, którzy chcą uporządkować wiedzę,
- analityków systemowych i biznesowych, którzy współpracują z architektem, opisują wymagania pozafunkcjonalne i chcą rozumieć konsekwencje decyzji architektonicznych,
- testerów, inżynierów DevOps i administratorów, którzy chcą rozumieć budowę systemów, z którymi pracują,
- kierowników projektów i osób po stronie zamawiającego, które odbierają systemy od wykonawców.

Doświadczonym architektom szkolenie posłuży raczej jako przegląd znanych zagadnień.

Od uczestników oczekujemy:
- posiadania doświadczenia w pracy przy projektach IT — jako programista, analityk, tester, administrator lub innej roli,
- ogólnej orientacji w tym, jak działają aplikacje webowe i bazy danych (klient, serwer, API, baza),
- podstawowej znajomości diagramów UML albo gotowości, by szybko je przyswoić — diagramy są na szkoleniu jednym z "języków pracy".

Zapytaj o szczegóły
tel. 22 63 64 164
akademia@alx.pl

Najbliższe terminy
2026-12-21 (Zdalnie)
2026-12-21 (Warszawa)
2027-04-14 (Zdalnie)
2027-04-14 (Warszawa)
2027-06-09 (Zdalnie)
2027-06-09 (Warszawa)

Umiejętność programowania nie jest wymagana. Osobom, które nie miały wcześniej kontaktu z UML, polecamy szkolenie Modelowanie w języku UML.

Certyfikaty

Uczestnicy szkolenia otrzymują imienne certyfikaty sygnowane przez ALX.

Lokalizacje

— Warszawa – ul. Jasna 14/16A
— Zdalnie – zajęcia realizowane poprzez platformę Zoom
— Kraków – ul. św. Filipa 23
— Warsaw (English) – Jasna 14/16A
— Online (English) – your home, office or wherever you want
— na życzenie dowolne miejsce w Polsce, lub UE (zajęcia prowadzone w języku angielskim)

Cena szkolenia

2990 PLN netto (VAT 23%)

W cenę szkoleń organizowanych w naszej siedzibie wliczone są:

— autorskie materiały szkoleniowe,
— indywidualne stanowisko komputerowe do pracy podczas zajęć,
— certyfikaty ukończenia szkolenia,
— drobny poczęstunek oraz ciepłe i zimne napoje,
— możliwość jednorazowego kontaktu z instruktorem (instruktorami) po szkoleniu i zadawania pytań dotyczących materiału szkolenia.

Cena szkolenia nie zawiera obiadów. Można je dokupić w cenie 35 zł netto za obiad.

Zapytaj o szczegóły
tel. 22 63 64 164
akademia@alx.pl

Najbliższe terminy
2026-12-21 (Zdalnie)
2026-12-21 (Warszawa)
2027-04-14 (Zdalnie)
2027-04-14 (Warszawa)
2027-06-09 (Zdalnie)
2027-06-09 (Warszawa)