

SZKOLENIE TECHNICZNE ZAAWANSOWANE

Program szkolenia

1. Zaawansowane metodyki i praktyki inżynierskie

- TDD, BDD i DDD
- Trunk-Based Development i CI/CD – metodyki pracy DevOps
- Chaos Engineering i Observability – metodyki pracy SRE
- Event Storming i Continuous Discovery
- Co metodyki mówią nam o seniority kandydata i umiejętnościach miękkich.

2. Zaawansowane technologie i języki programowania

- Dane i analityka – R, Julia, Scala.
- Systemy korporacyjne – COBOL, ABAP, PL/SQL.
- Wysokowydajny backend i mikroserwisy – Elixir, Rust, Go.
- Systemy niskopoziomowe i edge computing – Nim, Mojo, Zig, WebAssembly.
- Programowanie funkcyjne i systemy rozproszone – Erlang, Haskell, Clojure.
- Cross-platform mobile i frontend nowej generacji – Kotlin Multiplatform, Dart/Flutter, Svelte, Solid.js, Astro.

3. Architektura systemów i role architektoniczne

- Czym jest a czym nie jest architektura.
- Opis poszczególnych typów – architektura SOA, wielowarstwowa, P2P
- Wzorce projektowe – MVC, MVVM, widok pasywny.
- Paradygmaty programowania – opis, osadzenie w nich języków programowania. i wyjaśnienie dlaczego ich znajomość wpływa na seniority kandydata.
- Solution Architect vs Software Architect vs Enterprise Architect vs Data Architect – zakresy odpowiedzialności, różnice w perspektywie i celach.
- Jak rola architekta różni się w małych zespołach vs korporacjach

4. Technologie w biznesie

- Systemy klasy ERP i ich rola w zarządzaniu procesami (np. Microsoft Dynamics 365 – Sales, Finance & Operations, Marketing).
- Systemy WMS i TMS w logistyce i łańcuchu dostaw.
- Salesforce i narzędzia do integracji oraz zarządzania API (MuleSoft, Apigee).
- Jak wygląda proces integracji systemów zewnętrznych i gdzie w tym wszystkim jest rekruter.

5. AI, Data i Machine Learning w praktyce

- Od modelu do produkcji – jak wygląda proces ML Ops.
- Najpopularniejsze narzędzia: AWS SageMaker, Kubeflow, MLflow, LangChain, LlamaIndex, Hugging Face, TensorFlow, PyTorch
- AWS vs Google Clouds vs Azure – czym się różnią i jakie są charakterystyczne cechy
- Role wokół AI/ML: Data Scientist, ML Engineer, Data Engineer, AI Product Manager.

6. Technologia, która działa w tle

- Cybersecurity – kompetencje i certyfikaty, jak rozumieć poziomy specjalizacji, czym różnią się poszczególne role
- Embedded Systems – firmware, mikrokontrolery, technologie, typowe projekty
- DevOps i SRE różnice i technologie Kubernetes, Terraform, Prometheus, Grafana.
- Jak czytać CV osób, które nie tworzą produktu, ale sprawiają, że on działa.

7. QA i testowanie

- Piramida testów.
- Role testerskie: QA Engineer, Automation Engineer, Test Lead, SDET
- Narzędzia: frameworki (Selenium, Cypress, Playwright, RestAssured, Appium) i języki (Java, Python, JS)
- Shift-left & shift-right testing
- Coverage, defect density, MTTR, flakiness; jak po rozmowie z kandydatem poznać, że rozumie jakość systemowo.

7. QA i testowanie

- Piramida testów.
- Role testerskie: QA Engineer, Automation Engineer, Test Lead, SDET
- Narzędzia: frameworki (Selenium, Cypress, Playwright, RestAssured, Appium) i języki (Java, Python, JS)
- Shift-left & shift-right testing
- Coverage, defect density, MTTR, flakiness; jak po rozmowie z kandydatem poznać, że rozumie jakość systemowo.

8. Paradygmaty programowania

- Dlaczego paradygmaty w rekrutacji są kluczowe
- Główne sposoby myślenia o kodzie
- Jak rozpoznać paradygmat po technologii
- Jak pytać o paradygmat w rozmowie
- Co paradygmat mówi o kandydacie

9. Zaawansowane interview techniczne

- Jak rozpoznać realne doświadczenie projektowe i decyzje architektoniczne.
- Ocena „seniority myślenia” – kontekst, skala, wpływ decyzji.
- Analiza techniczna CV i wychwytywanie niespójności.
- Pytania, które warto zadać, żeby wniosły wartość do raportu i rekomendacji.
- Współprowadzenie rozmowy z ekspertem technicznym i interpretacja wyników.
- Jak przekładać złożone kompetencje inżynierskie na język biznesu.