M.Hidayatullah

Jawaban untuk soal teori:

1. Dalam membangun sistem backend yang berskala besar. Bagaimana Anda mendesain

arsitekturnya, dan apa tantangan yang Anda hadapi?

Answer: Saya biasanya menggunakan microservices atau modularmonolith tergantung kebutuhan.

* Microservices**:** Jika sistem perlu scalable secara independen. Gunakan API Gateway(Kong/Nginx) untuk routing, gRPC untuk komunikasi cepat, Kafka untuk event-driven architecture.
* Modular Monolith: Jika sistem masih dalam tahap awal, tetap terstruktur tapi mudah dikembangkan.  
  Tantangan**:** Latency antar layanan, data consistency, monitoring & observability (gunakan Prometheus + Grafana).
  1. Bagaimana Anda memilih antara SQL dan NoSQL? Sebutkan contoh kasus di mana Anda

menggunakan masing-masingnya.

Answer: SQL (MySQL/PostgreSQL): Untuk data yang memiliki relasi kuat, transaksi penting (ACID). Contoh: sistem keuangan, e-commerce.

NoSQL (MongoDB/Redis): Untuk data fleksibel, high read/write throughput. Contoh: real-time analytics, caching

3. Bagaimana Anda memastikan bahwa API yang Anda buat aman, mudah di-maintain, dan

scalable? Apa tools yang Anda gunakan untuk dokumentasi API?

Answer:

Keamanan API (Scalability & Maintenance)

* JWT/OAuth2 untuk autentikasi
* Rate limiting (Laravel Throttle, Nginx) untuk mencegah DDoS
* API Gateway (Kong, Nginx) untuk routing dan monitoring
* Documentation: Swagger atau Postman untuk dokumentasi API

4. Bagaimana Anda menangani autentikasi dan otorisasi di aplikasi backend? Sebutkan protokol

seperti OAuth atau JWT yang pernah Anda implementasikan.

Answer:

Autentikasi & Otorisasi

* OAuth2 jika aplikasi multi-platform (SSO).
* JWT jika butuh stateless authentication.
* Role-Based Access Control (RBAC) untuk izin granular.

5. Dalam mengoptimalkan performa aplikasi backend. Apa saja strategi atau alat yang kandidat

gunakan?

Answer:

Optimasi Performa Backend

* Caching (Redis, Memcached) untuk mengurangi query ke database.
* Queue (Laravel Queue dengan Redis/RabbitMQ) untuk background job.
* Load Balancing (Nginx, HAProxy) untuk distribusi trafik.
* Profiling (Blackfire, New Relic) untuk melihat bottleneck performa.

6. Apakah kandidat pernah bekerja dengan arsitektur microservices? Bagaimana kandidat

menangani komunikasi antar microservices dan data consistency?

Answer:

**Optimasi Performa Backend**

* Caching (Redis, Memcached) untuk mengurangi query ke database.
* Queue (Laravel Queue dengan Redis/RabbitMQ) untuk background job.
* Load Balancing (Nginx, HAProxy) untuk distribusi trafik.
* Profiling (Blackfire, New Relic) untuk melihat bottleneck performa.

7. Seberapa jauh pengalaman kandidat dalam DevOps? Bagaimana kandidat mengatur pipeline

CI/CD untuk deploy backend ke production?

Answer:

CI/CD & Deployment

* GitHub Actions / GitLab CI/CD untuk otomatisasi testing & deployment.
* Docker + Kubernetes (k8s) untuk scaling otomatis.
* Monitoring (Prometheus, ELK Stack) untuk logging dan observability.

8. Dalam menangani masalah kompleks di production. Apa pendekatan kandidat dalam debugging

dan root cause analysis?

Answer:

Debugging & Root Cause Analysis

* Log aggregation (ELK Stack, Loki, Datadog)
* Distributed tracing (Jaeger, OpenTelemetry)
* Profiling code dengan Xdebug atau Blackfire.

9. Framework atau bahasa backend apa yang Anda paling dikuasai, dan mengapa kandidat

memilih teknologi tersebut untuk proyek tertentu?

Answer:

 Laravel untuk rapid development dan ecosystem lengkap.

 Node.js jika butuh high-performance API.

10. Bagaimana cara Anda membimbing developer junior atau mid-level dalam tim? Bagaimana kandidat dapat memastikan tim backend berkembang?  
Bagaimana kandidat menangani data cache untuk meningkatkan performa aplikasi? Sebutkan pengalaman kandidat dengan Redis atau teknologi serupa.  
Bagaimana cara calon kandidat menangani migrasi database yang kompleks tanpa mengganggu aplikasi yang sedang berjalan?

Answer:

Mentoring Developer Junior/Mid-Level

* Pair programming, code review, knowledge sharing.
* Buat best practice documentation.
* Gunakan automated testing & linting untuk menjaga kualitas kode.