Examen Analyste Programmeur – Niveau Senior

UML

1. Quelle est la différence entre un diagramme de classes et un diagramme de séquence ? (approche avancée)
2. À quoi sert une relation d'agrégation en UML ? (approche avancée)
3. Quand utiliser un diagramme d'activités ? (approche avancée)
4. Expliquez la notion d'héritage en UML. (approche avancée)
5. Quelle est l'utilité des interfaces en UML ? (approche avancée)
6. Différence entre association et composition ? (approche avancée)
7. Que représente une multiplicité ? (approche avancée)
8. Diagramme UML le plus adapté pour modéliser un workflow ? (approche avancée)

Networking

9. Expliquez le modèle OSI. (approche avancée)
10. Différence entre TCP et UDP. (approche avancée)
11. À quoi sert le DNS ? (approche avancée)
12. Qu'est-ce qu'une adresse IP privée ? (approche avancée)
13. Différence entre HTTP et HTTPS ? (approche avancée)
14. Qu'est-ce qu'un pare-feu ? (approche avancée)
15. Expliquez le rôle d'un load balancer. (approche avancée)
16. Qu'est-ce que la latence réseau ? (approche avancée)

OOP

17. Expliquez les quatre piliers de la POO. (approche avancée)
18. Différence entre abstraction et encapsulation ? (approche avancée)
19. Qu'est-ce que le polymorphisme ? (approche avancée)
20. Quand utiliser une classe abstraite ? (approche avancée)
21. Différence entre héritage et composition ? (approche avancée)
22. Qu'est-ce qu'un constructeur ? (approche avancée)
23. Avantages de la POO ? (approche avancée)

24. Exemple de surcharge de méthode. (approche avancée)

DSA

25. Différence entre tableau et liste chaînée. (approche avancée)
26. Complexité temporelle de la recherche linéaire. (approche avancée)
27. Qu’est-ce qu’une pile (stack) ? (approche avancée)
28. Qu’est-ce qu’une file (queue) ? (approche avancée)
29. Complexité moyenne du tri rapide. (approche avancée)
30. À quoi sert une table de hachage ? (approche avancée)
31. Différence entre BFS et DFS. (approche avancée)
32. Qu’est-ce qu’un arbre binaire ? (approche avancée)

Design Patterns

33. À quoi sert le pattern Singleton ? (approche avancée)
34. Différence entre Factory et Abstract Factory. (approche avancée)
35. Quand utiliser Observer ? (approche avancée)
36. Avantages du pattern Strategy. (approche avancée)
37. Pattern MVC : rôle du contrôleur ? (approche avancée)
38. Pattern Adapter : cas d’usage. (approche avancée)
39. Différence entre State et Strategy ? (approche avancée)
40. Pourquoi éviter l’abus de Singleton ? (approche avancée)

Backend Patterns

41. Qu’est-ce qu’une architecture REST ? (approche avancée)
42. Différence entre monolithe et microservices. (approche avancée)
43. À quoi sert un API Gateway ? (approche avancée)
44. Qu’est-ce que la pagination côté backend ? (approche avancée)
45. Gestion des erreurs dans une API ? (approche avancée)
46. Qu’est-ce que l’idempotence ? (approche avancée)
47. Rôle des middlewares. (approche avancée)
48. Qu’est-ce qu’un service stateless ? (approche avancée)

SQL & Base de données

49. Différence entre DELETE et TRUNCATE. (approche avancée)
50. À quoi sert une clé primaire ? (approche avancée)
51. Qu’est-ce qu’une clé étrangère ? (approche avancée)
52. Différence entre INNER JOIN et LEFT JOIN. (approche avancée)
53. Qu’est-ce qu’une transaction ? (approche avancée)
54. Niveaux d’isolation des transactions. (approche avancée)
55. Index : avantages et inconvénients. (approche avancée)
56. Qu’est-ce que la normalisation ? (approche avancée)

Frontend Patterns

57. Qu’est-ce que MVC côté frontend ? (approche avancée)
58. Différence entre SPA et MPA. (approche avancée)
59. Rôle d’un state manager. (approche avancée)
60. Qu’est-ce que le data binding ? (approche avancée)
61. Avantages des composants réutilisables. (approche avancée)
62. Qu’est-ce que le lazy loading ? (approche avancée)
63. Gestion des formulaires côté frontend. (approche avancée)
64. Différence entre props et state. (approche avancée)