Examen: Applications Réparties - Microservices

Partie 1: Questions à Choix Multiples (20 points)

1. Quelle est la principale caractéristique d'une application monolithique?

   a) Plusieurs services indépendants

   b) Tous les composants regroupés dans une seule base de code

   c) Utilisation exclusive de microservices

   d) Architecture orientée événements

2. Quel est l'avantage principal du déploiement d'une application monolithique?

   a) Scalabilité indépendante des composants

   b) Simplicité de copier l'application empaquetée sur un serveur

   c) Possibilité d'utiliser différentes technologies pour chaque composant

   d) Tolérance aux pannes améliorée

3. Dans une architecture microservices, comment communiquent généralement les services entre eux?

   a) Par partage de mémoire

   b) Via des API (REST, gRPC) ou des systèmes de messagerie

   c) Par accès direct aux bases de données des autres services

   d) Via des appels de méthode directs

4. Quel est le rôle d'un API Gateway dans une architecture microservices?

   a) Stocker les données de tous les services

   b) Gérer toutes les requêtes clients et les router aux microservices concernés

   c) Exécuter la logique métier centrale

   d) Synchroniser les bases de données des différents services

5. Quel pattern permet de gérer les pannes en cascade entre microservices?

   a) Singleton

b) Circuit Breaker

c) Factory

d) Observer

6. Quelle technologie est souvent utilisée pour la communication asynchrone entre microservices?

a) FTP

b) SOAP

c) RabbitMQ

d) SQL

7. Dans une architecture microservices, comment sont généralement gérées les bases de données?

a) Une base de données unique partagée par tous les services

b) Chaque service possède sa propre base de données

c) Pas de persistance des données

d) Seulement des bases de données relationnelles

8. Quel est l'inconvénient principal des microservices mentionné dans le cours?

a) Difficulté d'évolution

b) Complexité accrue de gestion des communications

c) Performance réduite due au partage de mémoire

d) Impossibilité d'utiliser différentes technologies

9. Quel protocole est utilisé par gRPC pour la communication entre services?

a) HTTP/1.1

b) HTTP/2

c) FTP

d) SMTP

10. Quel est le principe "Stateless" dans une API REST?

a) Le serveur conserve l'état de la session

b) Chaque requête contient toutes les informations nécessaires

c) Le client ne peut pas changer d'état

d) Aucun état n'est jamais stocké

11. Quelle méthode HTTP est utilisée pour créer une ressource dans une API REST?

a) GET

b) POST

c) PUT

d) DELETE

12. Quel format de données est le plus couramment utilisé dans les API REST modernes?

a) XML

b) CSV

c) JSON

d) HTML

13. Quel outil est mentionné pour l'orchestration de microservices conteneurisés?

a) Jenkins

b) Kubernetes

c) Git

d) Nginx

14. Quel est l'avantage du "couplage faible" dans les microservices?

a) Meilleure performance des requêtes

b) Possibilité d'évoluer et déployer les services indépendamment

c) Simplification du code source

d) Réduction des coûts de développement

15. Quel mécanisme permet de découvrir automatiquement les instances de service disponibles?

a) API Gateway

b) Service Discovery

c) Circuit Breaker

d) Message Broker

16. Quelle méthode HTTP est utilisée pour mettre à jour complètement une ressource dans une API REST?

a) PATCH

b) POST

c) PUT

d) UPDATE

17. Quel est un exemple de Message Broker mentionné dans le cours?

a) MySQL

b) Apache Kafka

c) MongoDB

d) PostgreSQL

18. Quel code HTTP indique qu'une ressource a été créée avec succès?

a) 200

b) 201

c) 204

d) 400

19. Quel mécanisme de sécurité est recommandé pour les API REST?

a) HTTP Basic Auth

b) OAuth 2.0 ou JWT

c) Mot de passe en clair

d) IP Whitelisting uniquement

20. Quel outil est mentionné pour documenter une API REST?

a) JIRA

b) Swagger/OpenAPI

c) Trello

d) Slack

Partie 2: Étude de Cas (30 points)

Cas: Une entreprise de réservation de voyages en ligne souhaite migrer son application monolithique vers une architecture microservices. L'application actuelle gère les utilisateurs, les hôtels, les réservations et les paiements.

1. Proposez une décomposition en microservices pour cette application. (10 points)

2. Décrivez les mécanismes de communication que vous mettrez en place entre ces microservices, en justifiant vos choix. (10 points)

3. Identifiez trois défis potentiels de cette migration et proposez des solutions pour chacun. (10 points)

Corrigé

Partie 1: QCM

1. b) Tous les composants regroupés dans une seule base de code
2. b) Simplicité de copier l'application empaquetée sur un serveur
3. b) Via des API (REST, gRPC) ou des systèmes de messagerie
4. b) Gérer toutes les requêtes clients et les router aux microservices concernés
5. b) Circuit Breaker
6. c) RabbitMQ
7. b) Chaque service possède sa propre base de données
8. b) Complexité accrue de gestion des communications
9. b) HTTP/2
10. b) Chaque requête contient toutes les informations nécessaires
11. b) POST
12. c) JSON
13. b) Kubernetes
14. b) Possibilité d'évoluer et déployer les services indépendamment
15. b) Service Discovery
16. c) PUT
17. b) Apache Kafka
18. b) 201
19. b) OAuth 2.0 ou JWT
20. b) Swagger/OpenAPI

Partie 2: Étude de Cas

1. Décomposition proposée:
   - Service Utilisateurs: Gestion des comptes et authentification
   - Service Hôtels: Gestion des établissements et disponibilités
   - Service Réservations: Gestion des réservations et séjours

- Service Paiements: Traitement des transactions
- Service Notifications: Envoi de confirmations et alertes
- API Gateway: Point d'entrée unique

2. Mécanismes de communication:
- Synchrones: REST/HTTP pour les requêtes directes (ex: vérification de disponibilité)
- Asynchrones: Kafka pour les événements (ex: réservation confirmée → préparation facture)
- Service discovery pour localiser dynamiquement les services
- Circuit breaker pour gérer les pannes temporaires

3. Défis et solutions:
- Défi 1: Cohérence des données → Pattern Saga pour les transactions distribuées
- Défi 2: Surveillance des services → Implémentation de logging centralisé et monitoring
- Défi 3: Sécurité → API Gateway avec authentification centralisée et rate limiting

Note: Les réponses à l'étude de cas peuvent varier tant qu'elles sont pertinentes et bien argumentées.