■ QUESTIONNAIRE

Architecture Microservices & API Gateway

Bachelier Informatique - BAC3
50 Questions - 100 Points

Nom: _____

Prénom: _____

Date: _____

PARTIE 1 : Questions à Choix Multiples (QCM)

40 points (2 points par question)

Section A : Concepts de base

1. Qu'est-ce qu'une architecture microservices ?

A) Une application monolithique divisée en modules

B) Un style architectural où l'application est découpée en petits services indépendants

C) Un framework Java pour développer des applications

D) Un type de base de données distribuée

2. Quel est le principe fondamental concernant les bases de données dans une architecture microservices ?

A) Tous les services partagent la même base de données

B) Chaque microservice a sa propre base de données

C) Les microservices n'utilisent pas de bases de données

D) Une base de données centralisée gère tous les services

3. Quel est le rôle principal d'un API Gateway ?

A) Stocker les données des microservices

B) Point d'entrée unique qui route les requêtes vers les bons services

C) Compiler le code Java

D) Gérer uniquement la sécurité

4. Quelle méthode permet d'identifier les frontières des microservices ?

A) Agile

B) Scrum

C) Domain-Driven Design (DDD)

D) Waterfall

5. Dans une architecture microservices, comment les services communiquent-ils principalement ?

A) Via des fichiers partagés

B) Via des API REST, gRPC, ou message queues

C) Via une base de données commune

D) Via des appels directs de méthodes

Section B : API Gateway et Communication

6. Quel port utilise typiquement l'API Gateway dans les exemples du cours ?

A) 8081

B) 8082

C) 8080

D) 3000

7. Quels sont les avantages de l'API Gateway ? (plusieurs réponses possibles)

A) Les clients n'ont qu'une seule URL à connaître

B) Sécurité centralisée

C) Les microservices restent cachés

D) Toutes les réponses ci-dessus

8. Quelle est la différence principale entre REST et gRPC ?

A) REST utilise JSON, gRPC utilise du binaire (Protocol Buffers)

B) REST est plus rapide que gRPC

C) gRPC ne fonctionne que sur Windows

D) REST est asynchrone, gRPC est synchrone

9. Quand devrait-on utiliser REST plutôt que gRPC ?

A) Pour la communication interne entre microservices

B) Pour la communication externe (API publique, frontend → backend)

C) Jamais, gRPC est toujours meilleur

D) Uniquement pour les bases de données

10. Quel est le flux correct d'une requête passant par l'API Gateway ?

A) Client → Microservice → API Gateway

B) API Gateway → Client → Microservice

C) Client → API Gateway → Microservice → API Gateway → Client

D) Client → Base de données → API Gateway

Section C : Structure et Configuration

11. Quelles sont les 4 couches principales d'un microservice Spring Boot ?

A) Model, View, Controller, Database

B) Model, Repository, Service, Controller

C) Entity, DAO, Business, API

D) Frontend, Backend, Database, Cache

12. Dans quelle couche se trouve la logique métier (business logic) ?

A) Model

B) Repository

C) Service

D) Controller

13. Où se trouve le fichier application.properties ?

A) À la racine du projet

B) Dans src/main/java/

C) Dans src/main/resources/

D) Dans src/test/

14. Que contient typiquement le fichier application.properties ?

A) Le code Java de l'application

B) Les tests unitaires

C) La configuration (port, DB, URLs des services)

D) Les dépendances Maven

15. Quelle configuration définit le port d'écoute d'un service ?

A) spring.port=8080

B) server.port=8080

C) application.port=8080

D) service.port=8080

Section D : Annotations Spring

16. Quelle annotation démarre une application Spring Boot ?

A) @Application
B) @SpringBoot
C) @SpringBootApplication
D) @EnableSpring

17. Quelle annotation indique qu'une classe gère des API REST ?

A) @Controller
B) @RestController
C) @APIController
D) @WebController

18. Quelle annotation répond aux requêtes HTTP GET ?

A) @Get
B) @HTTPGet
C) @GetMapping
D) @RequestGet

19. Quelle annotation récupère une variable depuis l'URL (ex: /customers/{id}) ?

A) @URLVariable
B) @PathVariable
C) @RouteVariable
D) @RequestParam

20. Quelle annotation indique qu'une classe contient la logique métier ?

A) @Business
B) @Logic
C) @Service
D) @Component

PARTIE 2 : Questions Ouvertes

20 points (2 points par question)

21. Expliquez la différence entre une architecture monolithique et une architecture microservices.

22. Pourquoi est-il important que chaque microservice ait sa propre base de données ?

23. Donnez 3 rôles/responsabilités d'un API Gateway.

24. Dans un système e-commerce, citez 4 microservices différents que vous pourriez créer et leur responsabilité.

25. Expliquez le principe de "Separation of Concerns" dans le contexte des microservices.

26. Quelle est la différence entre communication synchrone et asynchrone ? Donnez un exemple de chaque.

27. Décrivez le flux complet d'une requête depuis le client jusqu'au microservice en passant par l'API Gateway.

28. Quels sont les avantages et inconvénients d'un découpage trop fin des microservices ?

29. Expliquez le rôle de chacune des 4 couches d'un microservice Spring Boot (Model, Repository, Service, Controller).

30. Pourquoi utilise-t-on l'annotation @Autowired dans Spring Boot ?

PARTIE 3 : Exercices Pratiques

25 points (5 points par exercice)

Exercice 1 : Configuration

31. Complétez le fichier application.properties suivant pour un service nommé "Product Service" qui écoute sur le port 8081 et utilise une base de données H2 en mémoire :

```
# Nom du service
spring.application.name=_________________

# Port d'écoute
server.port=_________________

# Configuration H2 Database
spring.datasource.url=_________________
spring.datasource.username=_________________
spring.datasource.password=_________________
spring.h2.console.enabled=_________________
```

Exercice 2 : Annotations

32. Annotez correctement cette classe Controller (écrivez les annotations manquantes):

Exercice 3 : Architecture

33. Dessinez (ou décrivez textuellement) l'architecture complète d'un système e-commerce avec 4 microservices (Product, Order, Customer, Payment), un API Gateway, les ports de chaque service et les flèches de communication.

Exercice 4 : Scénario réel

34. Un client veut créer une commande. Décrivez étape par étape le flux de communication entre les différents services.

Exercice 5 : Cas pratique - Découpage

35. Vous devez créer une application de gestion de bibliothèque. Proposez un découpage en microservices avec le nom de chaque microservice, ses responsabilités (ce qu'il fait) et ce qu'il NE fait PAS.

PARTIE 4 : Vrai ou Faux

10 points (1 point par question)

36. Dans une architecture microservices, un service peut accéder directement à la base de données d'un autre service.

■ Vrai ■ Faux

37. L'API Gateway peut gérer l'authentification et l'autorisation de manière centralisée.

■ Vrai ■ Faux

38. REST est toujours plus performant que gRPC.

■ Vrai ■ Faux

39. Chaque microservice doit être déployable indépendamment.

■ Vrai ■ Faux

40. L'annotation @Entity est utilisée dans la couche Controller.

■ Vrai ■ Faux

41. Le fichier application.properties peut contenir les URLs des autres microservices.

■ Vrai ■ Faux

42. Un découpage trop fin des microservices peut augmenter la complexité.

■ Vrai ■ Faux

43. gRPC utilise le format JSON pour communiquer.

■ Vrai ■ Faux

44. Spring Boot combine automatiquement @Configuration, @EnableAutoConfiguration, et @ComponentScan avec @SpringBootApplication.

■ Vrai ■ Faux

45. Dans une architecture microservices, tous les services doivent utiliser le même langage de programmation.

■ Vrai ■ Faux

PARTIE 5 : Questions de Réflexion

5 points (1 point par question)

46. Quels sont les inconvénients d'une architecture microservices par rapport à un monolithe ?

47. Comment gérer les transactions distribuées dans une architecture microservices ? (ex: si le paiement échoue après la création de la commande)

48. Expliquez pourquoi on dit que l'API Gateway est un "single point of failure". Comment peut-on mitiger ce risque ?

49. Dans quel cas préféreriez-vous une communication asynchrone (message queue) plutôt que synchrone (REST) ?

50. Quelle stratégie utiliseriez-vous pour gérer les versions d'API dans un système microservices ?

CORRIGÉ - RÉPONSES

PARTIE 1 : QCM - Réponses

1. B) Un style architectural où l'application est découpée en petits services indépendants
2. B) Chaque microservice a sa propre base de données
3. B) Point d'entrée unique qui route les requêtes vers les bons services
4. C) Domain-Driven Design (DDD)
5. B) Via des API REST, gRPC, ou message queues
6. C) 8080
7. D) Toutes les réponses ci-dessus
8. A) REST utilise JSON, gRPC utilise du binaire (Protocol Buffers)
9. B) Pour la communication externe (API publique, frontend → backend)
10. C) Client → API Gateway → Microservice → API Gateway → Client
11. B) Model, Repository, Service, Controller
12. C) Service
13. C) Dans src/main/resources/
14. C) La configuration (port, DB, URLs des services)
15. B) server.port=8080
16. C) @SpringBootApplication
17. B) @RestController
18. C) @GetMapping
19. B) @PathVariable
20. C) @Service

PARTIE 2 : Questions Ouvertes - Réponses

21. Monolithe : une seule application contenant toutes les fonctionnalités. Microservices : application découpée en services indépendants, chacun avec une responsabilité claire, déployables séparément.

22. Pour garantir l'indépendance des services, éviter les couplages forts, permettre le scaling indépendant et respecter le principe de responsabilité unique.

23. Routage vers les bons services, Sécurité centralisée (authentification/autorisation), Rate limiting, Logging/Monitoring, Load balancing.

24. Product Service (gestion catalogue), Order Service (gestion commandes), Customer Service (gestion clients), Payment Service (traitement paiements).

25. Chaque microservice gère son propre domaine métier sans se préoccuper des autres. Séparation claire des responsabilités.

26. Synchrone : le client attend la réponse (ex: REST, gRPC). Asynchrone : le client n'attend pas (ex: message queue, événements).

27. Client → envoie requête HTTP → API Gateway → route vers microservice → microservice traite → retourne réponse → API Gateway → Client.

28. Avantages : granularité fine, scaling précis. Inconvénients : complexité accrue, trop de communication réseau, overhead de gestion.

29. Model : définit les entités/données. Repository : accès aux données (DB). Service : logique métier. Controller : gère les endpoints REST.

30. Pour l'injection automatique de dépendances. Spring injecte automatiquement les beans nécessaires.

PARTIE 3 : Exercices Pratiques - Réponses

Exercice 31 - Configuration :

```
spring.application.name=product-service
server.port=8081
spring.datasource.url=jdbc:h2:mem:productdb
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.enabled=true
```

Exercice 32 - Annotations :

```
@RestController
@RequestMapping("/api/products")
@Autowired (pour productService)
@GetMapping
@GetMapping("/{id}") et @PathVariable Long id
@PostMapping et @RequestBody Product product
```

Exercice 33 - Architecture :

Client → API Gateway (8080) → Product Service (8081), Order Service (8082), Customer Service (8083), Payment Service (8084)

Exercice 34 - Scénario :

1) Client → POST /api/orders vers API Gateway, 2) API Gateway → Order Service, 3) Order Service vérifie stock (appel à Product Service), 4) Order Service valide client (appel à Customer Service), 5) Order Service traite paiement (appel à Payment Service), 6) Réponse renvoyée au client

Exercice 35 - Bibliothèque :

Book Service (catalogue livres), User Service (gestion utilisateurs), Loan Service (gestion emprunts/retours), Notification Service (rappels/alertes)

PARTIE 4 : Vrai ou Faux - Réponses

36. FAUX - Les services ne peuvent pas accéder directement aux DB des autres

37. VRAI

38. FAUX - gRPC est généralement plus performant que REST

39. VRAI

40. FAUX - @Entity est utilisé dans la couche Model

41. VRAI

42. VRAI

43. FAUX - gRPC utilise Protocol Buffers (binaire)

44. VRAI

45. FAUX - Chaque service peut utiliser un langage différent

PARTIE 5 : Questions de Réflexion - Réponses

46. Complexité accrue, overhead réseau, débogage plus difficile, nécessite expertise DevOps, coûts d'infrastructure plus élevés.

47. Pattern Saga (choreography ou orchestration), compensation en cas d'échec, eventual consistency, event sourcing.

48. Si l'API Gateway tombe, tout le système est inaccessible. Mitigation : redondance (plusieurs instances), load balancer, health checks, failover automatique.

49. Quand la réponse immédiate n'est pas nécessaire, pour découpler les services, pour gérer les pics de charge, pour la résilience (retry automatique).

50. Versioning dans l'URL (/v1/, /v2/), headers HTTP (Accept-Version), backwards compatibility, dépréciation progressive.

BARÈME

Partie	Questions	Points
QCM	Q1-Q20	40 points
Questions Ouvertes	Q21-Q30	20 points
Exercices Pratiques	Q31-Q35	25 points
Vrai/Faux	Q36-Q45	10 points
Réflexion	Q46-Q50	5 points
TOTAL	**50 questions**	**100 points**