Institut TECCART

420-TT-RCW

Guide Technique sur les Concepts d'API, d'Architecture Client-Serveur, de Micro-services, de Webservices et de Design Patterns

Descriptif du document

Cette ressource est un guide d'entretien exhaustif pour les professionnels de l'informatique, en particulier ceux qui travaillent avec les API, l'architecture client-serveur, les micro-services, les webservices et les design patterns comme MVC. Il contient des questions d'entretien détaillées avec des réponses sur ces concepts. Les questions visent à évaluer la compréhension de ces concepts clés, tandis que les réponses servent de guide de référence pour les candidats se préparant pour des entretiens.

Last update: 2023-07-30

Questions

1. **Qu'est-ce qu'une API ?**

   Une API (Application Programming Interface) est un ensemble de règles et de protocoles pour la construction et l'interaction de logiciels. Elle permet aux différents logiciels de communiquer entre eux.

2. **Qu'est-ce qu'une REST API?**

   REST (Representational State Transfer) est un style architectural pour la conception de réseaux d'applications. Une REST API est une API qui adhère à ces principes et utilise le protocole HTTP pour les opérations de création, de lecture, de mise à jour et de suppression de données.

3. **Qu'est-ce qu'une API RESTful?**

   Une API RESTful est une API qui suit les principes de REST. Elle doit permettre aux clients de manipuler les ressources de l'API à travers un ensemble prédéfini d'opérations stateless, souvent mappées sur les méthodes HTTP comme GET, POST, PUT, DELETE.

4. **Comment utilise-t-on Flask pour créer une API RESTful?**

   Flask est un framework web Python qui peut être utilisé pour créer des API RESTful. Vous pouvez définir différentes routes pour correspondre aux différentes méthodes HTTP, et manipuler les données à travers ces routes.

5. **Pouvez-vous donner un exemple d'une route GET avec Flask?** Oui, voici un exemple :

```
python
from flask import Flask, jsonify
app = Flask(__name__)
@app.route('/api/data', methods=['GET'])
def get_data():
     return jsonify({'name': 'John', 'age': 30})
```

6. **Comment Django peut-il être utilisé pour créer une API RESTful?**

   Django est un autre framework web Python qui peut être utilisé pour créer des API RESTful. En utilisant l'extension Django REST Framework, vous pouvez définir des vues basées sur les classes qui correspondent aux différentes méthodes HTTP.

7. **Pouvez-vous donner un exemple d'une vue POST avec Django REST Framework?** Oui, voici un exemple :

```
python
from rest_framework import status
from rest_framework.response import Response
from rest_framework.views import APIView

class SampleView(APIView):
def post(self, request, format=None):
         data = request.data
         return Response(data, status=status.HTTP_201_CREATED)
```

8. **Qu'est-ce que Node.js et comment peut-il être utilisé pour créer une API RESTful?**

   Node.js est un environnement d'exécution JavaScript côté serveur qui peut être utilisé pour créer des API RESTful. Vous pouvez utiliser des frameworks comme Express.js pour définir des routes correspondant aux différentes méthodes HTTP.

9. **Pouvez-vous donner un exemple d'une route PUT avec Express.js?** Oui, voici un exemple :

```
javascript
const express = require('express');
const app = express();
app.put('/api/data/:id', (req, res) => {
// Mettez à jour la ressource ici
res.send(`Resource ${req.params.id} updated`);
});
```

10. **Qu'est-ce que l'architecture client-serveur?**

    L'architecture client-serveur est un modèle de design de système où le serveur fournit des ressources ou des services, et le client accède à ces services. Le client fait des requêtes au serveur, qui traite ces requêtes et renvoie les réponses appropriées.

11. **Comment l'architecture client-serveur est-elle utilisée dans le web?**

    Sur le web, un navigateur agit comme un client qui fait des requêtes à un serveur web pour des pages web ou d'autres ressources web. Le serveur web traite ces requêtes et renvoie les réponses appropriées.

12. **Qu'est-ce que l'architecture microservices?**

    L'architecture microservices est un style architectural qui structure une application comme une collection de services faiblement couplés. Chaque service est autonome et peut être déployé, mis à l'échelle et redémarré indépendamment.

13. **Quels sont les avantages de l'architecture microservices par rapport à une architecture monolithique?**

Les microservices offrent une meilleure modularité, facilitent le déploiement et la mise à l'échelle de composants individuels, permettent l'utilisation de différentes technologies pour différents services et peuvent améliorer la résilience du système en isolant les échecs.

14. **Qu'est-ce qu'un webservice?**

    Un webservice est un service offert par une application électronique à une autre application électronique, communiquant avec elle via le réseau. Les webservices sont souvent implémentés en utilisant des API REST ou SOAP.

15. **Quelle est la différence entre un webservice et une API?**

    Toutes les APIs ne sont pas des webservices. Un webservice est une API qui est accessible via le réseau et utilise un protocole de communication réseau comme HTTP.

16. **Qu'est-ce que le modèle MVC?**

    MVC (Model-View-Controller) est un patron de conception qui sépare une application en trois composants interagissant : le Modèle (qui gère les données), la Vue (qui gère la présentation des données) et le Contrôleur (qui gère les interactions entre le Modèle et la Vue).

17. **Comment le modèle MVC est-il utilisé dans Django?**

    Django suit une version modifiée du modèle MVC appelée MVT (Model-View-Template). Le Modèle correspond au modèle de Django, la Vue à la vue de Django, et le Contrôleur à l'infrastructure de Django elle-même qui déroute les requêtes HTTP vers les vues appropriées.

18. **Qu'est-ce que le patron de conception Singleton?**

    Le Singleton est un patron de conception qui garantit qu'une classe n'a qu'une seule instance et fournit un point d'accès global à cette instance.

19. **Qu'est-ce que le patron de conception Factory?**

    Le Factory Pattern est un patron de conception qui fournit une interface pour créer des objets dans une super-classe, mais permet aux sous-classes de modifier le type d'objets qui seront créés.

20. **Pouvez-vous donner un exemple de Factory Pattern en Python?**
    Oui, voici un exemple simple :

```
python
class AnimalFactory:
def create_animal(self, type):
      if type == 'Dog':
              return Dog()
      elif type == 'Cat':
              return Cat()
      else:
              raise ValueError('Invalid type')
```

21. **Qu'est-ce que le patron de conception Observer?**

    L'Observer Pattern est un patron de conception où un objet (l'observé) maintient une liste d'autres objets (les observateurs) et les notifie automatiquement de tout changement d'état.

22. **Qu'est-ce qu'une architecture serverless?**

    L'architecture serverless est un modèle où les développeurs peuvent construire et exécuter des applications et des services sans avoir à gérer l'infrastructure. Les services serverless sont déclenchés par des événements spécifiques et s'exécutent uniquement lorsque nécessaire.

23. **Comment le modèle MVC est-il utilisé dans le développement d'applications mobiles?**

    Dans le développement d'applications mobiles, le modèle (Modèle) gère les données, les règles métier et la logique, la vue (Vue) est responsable de l'affichage des données à l'utilisateur, et le contrôleur (Contrôleur) est le pont qui relie le modèle et la vue.

24. **Qu'est-ce qu'une API GraphQL?**

    GraphQL est une API open-source qui fournit un cadre pour les demandes de données client-servants. Contrairement aux API REST, qui ont des endpoints fixes, GraphQL permet aux clients de demander exactement ce dont ils ont besoin, ce qui peut réduire la quantité de données qui doivent être transférées.

25. **Quelle est la différence entre une API publique et une API privée?**

    Une API publique est accessible à tous, tandis qu'une API privée est destinée à être utilisée uniquement en interne par l'entreprise qui l'a créée.

26. **Qu'est-ce que l'architecture client-serveur dans le contexte d'une API?**

L'architecture client-serveur dans le contexte d'une API fait référence à un modèle de communication où le client (par exemple, une application Web) envoie des demandes à un serveur via l'API, et le serveur renvoie des réponses.

27. **Qu'est-ce qu'un endpoint dans une API?**

    Un endpoint dans une API est une URL spécifique où une API peut être accédée. Chaque endpoint représente une fonction spécifique de l'API.

28. **Qu'est-ce qu'un microservice dans une architecture de microservices?**

    Un microservice dans une architecture de microservices est une application logicielle autonome qui fonctionne comme un service indépendant et qui communique avec d'autres services via une API.

29. **Qu'est-ce qui différencie l'architecture microservices de l'architecture monolithique?**

    Dans une architecture monolithique, tous les composants de l'application sont interdépendants et fonctionnent comme une seule unité, tandis que dans une architecture microservices, chaque service est indépendant et peut être développé, déployé et mis à l'échelle indépendamment des autres.

30. **Comment les microservices communiquent-ils entre eux?**

    Les microservices communiquent généralement entre eux à travers des protocoles HTTP/REST ou des services de messagerie asynchrones, en utilisant des formats de message légers comme JSON.

31. **Qu'est-ce qu'un webservice?**

    Un webservice est une application ou un service accessible par le biais d'un réseau, généralement sur Internet, qui communique avec d'autres programmes ou applications en utilisant des technologies comme HTTP, XML, SOAP, etc.

32. **Qu'est-ce que SOAP en termes de webservices?**

    SOAP (Simple Object Access Protocol) est un protocole standardisé qui permet l'échange de données structurées dans l'implémentation de webservice. Il peut fonctionner sur n'importe quel protocole et est généralement utilisé avec HTTP pour les communications sur le web.

33. **Qu'est-ce que REST en termes de webservices?**

    REST (Representational State Transfer) est un style d'architecture pour développer des applications réseau. Les webservices RESTful utilisent des méthodes HTTP standard pour créer, lire, mettre à jour et supprimer des données.

34. **Quelle est la différence entre un webservice et une API?**

Un webservice est une forme d'API qui fonctionne sur un réseau pour faciliter l'interaction entre les machines. Cependant, toutes les API ne sont pas des webservices. Les API peuvent également permettre la communication entre différentes parties d'une application sur une seule machine.

35. **Qu'est-ce qu'un serveur dans une architecture client-serveur?**

    Dans une architecture client-serveur, un serveur est une entité qui fournit des services ou des ressources à d'autres programmes ou appareils, appelés clients. Les serveurs reçoivent des demandes de clients, traitent ces demandes et renvoient les résultats.

36. **Qu'est-ce qu'un client dans une architecture client-serveur?**

    Dans une architecture client-serveur, un client est un programme ou un appareil qui demande des services ou des ressources à un serveur. Les clients initient des demandes de communication avec les serveurs.

37. **Qu'est-ce qu'une architecture de microservices distribués?**

    Une architecture de microservices distribués est un style d'architecture où une application est divisée en une collection de services autonomes qui sont déployés et exécutés sur différents serveurs ou environnements.

38. **Comment les problèmes de sécurité sont-ils gérés dans une architecture de microservices?**

    Dans une architecture de microservices, les problèmes de sécurité peuvent être gérés en utilisant des technologies comme les API sécurisées, l'authentification et l'autorisation, le cryptage, et en suivant les meilleures pratiques de sécurité comme le principe du moindre privilège.

39. **Qu'est-ce qu'un service Web RESTful?**

    Un service Web RESTful est un service Web qui suit les principes de l'architecture REST. Il utilise des méthodes HTTP standard pour exposer des fonctionnalités et est généralement basé sur des URL pour l'accès aux ressources et des formats de données standard (comme JSON ou XML) pour le transfert de données.

40. **Qu'est-ce qu'un service Web SOAP?**

    Un service Web SOAP est un service Web qui utilise le protocole SOAP pour échanger des données structurées sur le web. Contrairement aux services Web RESTful, les services Web SOAP utilisent des opérations définies dans des fichiers WSDL et peuvent fonctionner sur n'importe quel protocole de transport.

41. **Qu'est-ce que JSON et comment est-il utilisé dans les services Web?**

JSON (JavaScript Object Notation) est un format de données léger qui est facile pour les humains à lire et à écrire et facile pour les machines à analyser et à générer. Dans les services Web, JSON est souvent utilisé pour échanger des données entre le client et le serveur.

42. **Qu'est-ce que XML et comment est-il utilisé dans les services Web?**

    XML (eXtensible Markup Language) est un langage de balisage qui définit un ensemble de règles pour l'encodage de documents d'une manière qui est à la fois lisible par l'homme et lisible par la machine. Dans les services Web, XML est souvent utilisé pour structurer les données dans les demandes et les réponses.

43. **Qu'est-ce que le protocole HTTP et comment est-il utilisé dans les services Web?**

    HTTP (Hypertext Transfer Protocol) est un protocole qui définit comment les messages sont formatés et transmis sur le web, et quelles actions les serveurs web et les navigateurs doivent prendre en réponse à diverses commandes. Dans les services Web, HTTP est généralement utilisé comme protocole de transport pour les demandes et les réponses.

44. **Qu'est-ce que le patron de conception Builder?**

    Le Builder Pattern est un patron de conception qui fournit une solution pour construire un objet complexe en utilisant des étapes simples. Il sépare la construction d'un objet complexe de sa représentation, de sorte que le même processus de construction peut créer différentes représentations.

45. **Qu'est-ce que le patron de conception Prototype?**

    Le Prototype Pattern est un patron de conception qui spécifie les types d'objets à créer en utilisant une instance prototype, et crée de nouveaux objets en copiant ce prototype.

46. **Comment Node.js peut-il être utilisé dans une architecture microservices?**

    Node.js peut être utilisé pour créer des services individuels dans une architecture microservices. Chaque service peut être une application Node.js autonome qui communique avec les autres services via des API RESTful, par exemple.

47. **Comment Flask et Django peuvent-ils être utilisés dans une architecture microservices?**

    Flask et Django, étant des frameworks web Python, peuvent être utilisés pour créer des services individuels dans une architecture microservices. Chaque service peut être une application Flask ou Django autonome qui communique avec les autres services via des API RESTful.

48. **Qu'est-ce qu'un service Web SOAP?**

SOAP (Simple Object Access Protocol) est un protocole de messagerie qui permet aux programmes fonctionnant sur des systèmes d'exploitation distincts de communiquer entre eux via le web. Un service Web SOAP utilise ce protocole pour échanger des informations.

49. **Quelle est la différence entre REST et SOAP?**

    REST est un style architectural, tandis que SOAP est un protocole. REST utilise généralement le protocole HTTP et peut utiliser différents formats de données, tandis que SOAP utilise son propre protocole et utilise le format XML pour les données.

50. **Qu'est-ce que l'architecture basée sur les événements?**

    L'architecture basée sur les événements est un modèle architectural où les flux de travail sont déclenchés par des événements spécifiques. Dans ce modèle, un morceau de logiciel envoie des événements (changements d'état) aux autres sans savoir qui les reçoit.

51. **Qu'est-ce que le patron de conception Strategy?**

    Le Strategy Pattern est un patron de conception qui permet de sélectionner un algorithme à l'exécution. Il définit une famille d'algorithmes, encapsule chacun d'eux et les rend interchangeables.

52. **Qu'est-ce que le patron de conception Composite?**

    Le Composite Pattern est un patron de conception qui permet de traiter les objets individuels et les compositions d'objets de manière uniforme. Il est utile pour représenter des structures d'objets hiérarchiques.

53. **Qu'est-ce que le patron de conception Adapter?**

    L'Adapter Pattern est un patron de conception qui permet aux classes ayant des interfaces incompatibles de travailler ensemble. Il enveloppe l'objet avec une interface différente.

54. **Qu'est-ce que le patron de conception Bridge?**

    Le Bridge Pattern est un patron de conception qui sépare une abstraction de son implémentation afin que les deux puissent être modifiées indépendamment.

55. **Qu'est-ce que le patron de conception Decorator?**

    Le Decorator Pattern est un patron de conception qui permet d'ajouter de nouveaux comportements à des objets en les plaçant à l'intérieur d'objets décorateurs.

56. **Qu'est-ce que le patron de conception Command?**

    Le Command Pattern est un patron de conception qui encapsule une demande en tant qu'objet, ce qui permet aux utilisateurs de paramétrer des files d'attente, des demandes et des opérations.

57. **Qu'est-ce que le patron de conception State?**

    Le State Pattern est un patron de conception qui permet à un objet de modifier son comportement lorsque son état interne change.

58. **Qu'est-ce que le patron de conception Visitor?**

    Le Visitor Pattern est un patron de conception qui permet d'ajouter de nouvelles opérations virtuelles à une famille de classes sans modifier les classes elles-mêmes.

59. **Qu'est-ce que le patron de conception Memento?**

    Le Memento Pattern est un patron de conception qui fournit un moyen de restaurer l'état d'un objet à un état précédent.

60. **Qu'est-ce que le patron de conception Iterator?**

    L'Iterator Pattern est un patron de conception qui fournit un moyen d'accéder séquentiellement aux éléments d'un objet complexe sans exposer ses détails d'implémentation.

61. **Qu'est-ce que le patron de conception Template?**

    Le Template Pattern est un patron de conception qui définit le squelette d'un algorithme dans une méthode, en repoussant certaines étapes à des sous-classes.

62. **Qu'est-ce que le patron de conception Observer?**

    L'Observer Pattern est un patron de conception qui définit une dépendance d'un à plusieurs, de manière que lorsqu'un objet change d'état, tous ses dépendants soient notifiés et mis à jour automatiquement.

63. **Qu'est-ce que le patron de conception Singleton?**

    Le Singleton Pattern est un patron de conception qui garantit qu'une classe a une seule instance, et fournit un point d'accès global à cette instance.