WEB API – Entrevista : Questões e Respostas

1 - O que é uma API ?

Uma API (Application Programming Interface) é um intermediário de software que permite que dois aplicativos se comuniquem entre si. Ela inclui várias definições de subrotinas, registros e ferramentas para criar software de aplicativo.

Exemplos de API : API do Google Maps, API de publicidade da Amazon, API do Twitter, API do YouTube etc.

2. Quais são as principais diferenças entre uma API e um web service ou serviço da Web?

- Todos os web services ou serviços da Web são APIs, mas nem todas as APIs são serviços da Web.

- Os web services ou serviços da Web podem não conter todas as especificações e não podem executar todas as tarefas que as APIs executariam.

- Um serviço da Web usa apenas três estilos de uso: SOAP, REST e XML-RPC para comunicação, enquanto a API pode ser exposta de várias maneiras.

- Um serviço da Web sempre precisa de uma rede para operar, enquanto as APIs não precisam de uma rede para operação.

3. Quais são os limites do uso da API?

Muitas APIs têm um certo limite configurado pelo provedor. Assim, tente estimar seu uso e entender como isso afetará o custo total da oferta. Se isso será um problema, depende em grande parte de como os dados são aproveitados.

4. Cite alguns estilos arquitetônicos para criar uma Web API ?

Temos quatro estilos de arquitetura comuns para criar Web APIs:

- HTTP para comunicação entre cliente e servidor
- XML / JSON como linguagem de formatação
- URI simples como o endereço dos serviços
- Comunicação sem estado

5. Quem pode usar uma API da Web?

Uma Web API pode ser consumida por quaisquer clientes que suportem verbos HTTP, como GET, PUT, DELETE, POST. Como os serviços da Web API não exigem configuração, eles podem ser facilmente usados por qualquer cliente. De fato, até mesmo dispositivos portáteis, como dispositivos móveis, podem facilmente usar a Web API, que é, sem dúvida, a maior vantagem dessa tecnologia.

6. O que é um teste de API ?

O teste de API é um tipo de teste de software que determina se as APIs desenvolvidas atendem às expectativas relacionadas à funcionalidade, confiabilidade, desempenho e segurança do aplicativo.

7. Quais são as vantagens do teste de API ?

A seguir algumas das vantagens de realizar o teste da API :

1- Teste da funcionalidade principal: o teste da API fornece acesso ao aplicativo sem uma interface do usuário. O núcleo e o nível de código das funcionalidades do aplicativo serão testados e avaliados antes dos testes de GUI. Isso ajudará a detectar os problemas menores que podem se tornar maiores durante o teste da GUI.

2- Tempo efetivo: o teste de API geralmente consome menos tempo do que o teste de GUI funcional. Os elementos da web no teste de GUI devem ser pesquisados, o que torna o processo de teste mais lento. Particularmente, a automação de testes de API requer menos código, de modo que possa fornecer uma cobertura de teste melhor e mais rápida em comparação com a automação de teste de GUI. Isso resultará na economia de custos do projeto de teste.

3- Independente de idioma: no teste de API, os dados são trocados usando XML ou JSON. Esses modos de transferência são completamente independentes do idioma, permitindo que os usuários selecionem qualquer linguagem de código ao adotem serviços de teste de automação para o projeto.

4- Integração fácil com GUI: os testes de API permitem testes altamente integráveis, o que é particularmente útil se você deseja executar testes de GUI funcionais após o teste da API. Por exemplo, a integração simples permitiria que novas contas de usuários fossem criadas dentro do aplicativo antes do início de um teste de GUI.

8. Quais protocolos comuns podem ser usados em testes de API?

Muitos protocolos estão agora disponíveis para serem usados em testes de API, como JMS, REST, HTTP, UDDI e SOAP.

9. Qual é o ambiente de teste da API?

O ambiente de teste da API é um pouco completo e requer a configuração do banco de dados e do servidor, dependendo dos requisitos de software. Nenhuma GUI (Graphical User Interface) está disponível nesta etapa de teste.

Quando o processo de instalação é concluído, a API é verificada para a operação correta. Durante todo o processo, a API chamada do ambiente original é configurada com parâmetros diferentes para estudar os resultados do teste.

10. Quais são os princípios de padrão de Teste de API ?

Os cinco princípios mais importantes de um padrão de teste de API são:

1- Configuração: Criar objetos, iniciar serviços, inicializar dados, etc.
2- Execução: Etapas para aplicar a API ou o cenário, incluindo o registro em log
3- Verificação: Avaliar o resultado da execução
4- Relatórios: Aprovação, falha ou bloqueio
5- Limpeza: Estado de pré-teste

11. Quais são os tipos comuns de testes da API?

Embora existam testes especializados, a maioria dos testes se encaixa amplamente nas seguintes nove categorias que você deve lembrar antes de participar de uma entrevista de teste de API.

- Teste de validação
- Teste funcional
- Teste de interface do usuário
- Teste de carga
- Tempo de Execução / Detecção de Erros
- Testes de segurança
- Teste de penetração
- Teste de fuzz
- Interoperabilidade e teste de conformidade com WS

12. Qual é o procedimento para realizar testes de API?

- Escolha o conjunto para adicionar o caso de teste da API
- Escolha o modo de desenvolvimento de teste
- Exigir o desenvolvimento de casos de teste para os métodos de API necessários
- Configure os parâmetros de controle do aplicativo e, em seguida, teste as condições
- Configurar validação de método
- Execute o teste da API

- Verificar relatórios de teste e filtrar casos de teste da API
- Organizar todos os casos de teste da API

13. O que deve ser verificado ao realizar testes de API?

Durante o processo de teste da API, uma requisição é enviada à API com os dados conhecidos. Desta forma, você pode analisar a resposta de validação. Ao testar uma API, você deve considerar:

- Precisão de dados
- Validação de esquema
- Códigos de status http
- Tipo de dados, validações, ordem e integridade
- Verificações de autorização
- Implementação do tempo limite de resposta
- Códigos de erro no caso de a API retornar e
- Testes não funcionais, como testes de desempenho e segurança

14. O que é documentação da API?

A documentação da API é uma redação técnica completa e precisa que fornece instruções sobre como usar e integrar-se efetivamente a uma API. É um manual de referência compacto que contém todas as informações necessárias para trabalhar com a API e ajuda a responder a todas as perguntas de testes da API com detalhes sobre funções, classes, tipos de retorno, argumentos e também exemplos e tutoriais.

15. Quais são os modelos de documentação da API que são comumente usados?

Existem vários modelos de documentação da API disponíveis para ajudar a tornar todo o processo simples e direto:

- Swagger
- Miredot
- Ardósia
- FlatDoc
- Modelo de API
- RestDoc
- Especificação da API de serviço da web

16. Ao escrever o documento da API, o que deve ser considerado?

- Fonte do conteúdo
- Plano de documento ou esboço
- Layout de entrega
- Informações necessárias para todas as funções no documento
- Programas de criação automática de documentos

17. O que é o REST?

REST (Representational State Transfer) é um estilo de arquitetura para o desenvolvimento de serviços da Web que exploram a onipresença do protocolo HTTP e usa o método HTTP para definir ações. Ele gira em torno do conceito no qual cada componente é um recurso que pode ser acessado por meio de uma interface compartilhada usando métodos HTTP padrão.

Na arquitetura REST, um servidor REST fornece acesso a recursos e acessos a clientes REST e disponibiliza esses recursos. Aqui, cada recurso é identificado por URIs ou IDs globais, e o REST usa várias maneiras de representar um recurso, como texto, JSON e XML. XML e JSON são hoje as representações mais populares de recursos.

WEB API – Entrevista : Questões e Respostas

18. O que é um Web Services RESTFul?

Existem dois tipos principais de web services :

O SOAP (Simple Object Access Protocol) - um método baseado em XML para expor serviços da web.

E os serviços da Web desenvolvidos no estilo REST que são referidos como serviços da Web RESTful. Esses serviços da web usam métodos HTTP para implementar o conceito de arquitetura REST.

Um serviço web RESTful geralmente define um URI, um serviço de identificador uniforme de recursos, fornece representação de recursos como JSON e um conjunto de métodos HTTP.

19. O que é um "recurso" no REST?

A arquitetura REST trata qualquer conteúdo como um recurso, que pode ser arquivos de texto, páginas HTML, imagens, vídeos ou informações comerciais dinâmicas.

O REST Server fornece acesso a recursos e os modifica, onde cada recurso é identificado por URIs / IDs globais.

20. Qual é a maneira mais popular de representar um recurso no REST?

O REST usa diferentes representações para definir um recurso como texto, JSON e XML.

XML e JSON são as representações mais populares de recursos.

21. Qual protocolo é usado pelos serviços da Web RESTful?

Os serviços da Web RESTful usam o protocolo HTTP como um meio de comunicação entre o cliente e o servidor.

22. Quais são algumas das principais características do REST?

As principais características do REST são :

- REST é stateless, portanto, o SERVER não possui status (ou dados da sessão)

- Com uma API REST bem aplicada, o servidor pode ser reiniciado entre duas chamadas, pois todos os dados são transferidos para o servidor

- O serviço da Web usa o método POST principalmente para executar operações, enquanto o REST usa GET para acessar recursos.

23. O que é o serviço de mensagens em serviços da Web RESTful?

Os serviços da Web RESTful usam o protocolo HTTP como uma ferramenta de comunicação entre o cliente e o servidor. A técnica que quando o cliente envia uma mensagem na forma de uma solicitação HTTP, o servidor envia de volta a resposta HTTP é chamada de mensagens. Essas mensagens compreendem dados de mensagens e metadados, isto é, informações sobre a própria mensagem.

24. Quais são os principais componentes de uma solicitação HTTP?

Uma solicitação HTTP contém cinco elementos principais:

- Uma Action mostrando métodos HTTP como GET, PUT, POST, DELETE.
- Uniform Resource Identifier (URI), que é o identificador do recurso no servidor.
- A Versão HTTP, que indica a versão HTTP, por exemplo, HTTP v1.1.
- O Request Header, que transporta metadados (como pares de valor-chave) para a mensagem HTTP Request. Os metadados podem ser do tipo cliente (ou navegador), formato suportado pelo cliente, formato do formato do corpo da mensagem, configurações de cache e assim por diante.
- O Request Body, que indica o conteúdo da mensagem ou a representação do recurso.

WEB API – Entrevista : Questões e Respostas

25. Quais são os métodos HTTP mais usados pelo REST?

GET é usado apenas para solicitar dados de um recurso especificado. As solicitações de obtenção podem ser armazenadas em cache e marcadas como favoritos. Ele permanece no histórico do navegador. As solicitações GET nunca devem ser usadas ao lidar com dados confidenciais

O POST é usado para enviar dados para um servidor para criar/atualizar um recurso. As solicitações POST nunca são armazenadas em cache, marcadas e não permanecem no histórico do navegador.

PUT substitui todas as representações atuais do recurso de destino com a carga útil da solicitação.

DELETE remove o recurso especificado.

OPTIONS é usado para descrever as opções de comunicação para o recurso de destino.

HEAD pede uma resposta idêntica à de uma solicitação GET, mas sem o corpo da resposta.

26. A solicitação GET pode ser usada em vez de PUT para criar um recurso?

O método PUT ou POST deve ser usado para criar um recurso. GET é usado apenas para solicitar dados de um recurso especificado.

27. Existe alguma diferença entre as operações PUT e POST?

As operações PUT e POST são bastante semelhantes, exceto os termos do resultado gerado por eles.

A operação PUT é idempotente, portanto, você pode armazenar em cache a resposta enquanto as respostas à operação POST não podem ser armazenadas em cache e, se tentar novamente N vezes, você terá N recursos com N URIs diferentes criadas no servidor.

28. O que é URI? Qual é o objetivo principal dos serviços da Web baseados em REST e qual é o seu formato?

URI significa Uniform Resource Identifier (Identificador Uniforme de Recursos). É uma cadeia de caracteres projetada para identificação inequívoca de recursos e extensibilidade por meio do esquema de URI.

O objetivo de um URI é localizar um recurso no servidor que hospeda o serviço da web.

O formato de um URI é <protocol>: // <service-name> / <ResourceType> / <ResourceID>.

29. O que é payload em serviços da Web RESTFul?

A “carga útil” ou payload são os dados que você está interessado em transportar. Isso é diferenciado das coisas que envolvem os dados para transporte, como os cabeçalhos de solicitação / resposta HTTP / S, autenticação etc.

30. Qual é o limite superior para uma carga útil passar no método POST?

O <GET> acrescenta dados ao URL do serviço. Porém, seu tamanho não deve exceder o tamanho máximo do URL. No entanto, <POST> não tem esse limite.

Então, teoricamente, um usuário pode passar dados ilimitados como a carga útil para o método POST. Mas, se considerarmos um caso de uso real, o envio de POST com carga útil grande consumirá mais largura de banda. Levará mais tempo e apresentará desafios de desempenho ao seu servidor. Portanto, um usuário deve agir de acordo.

31. Qual é o mecanismo de armazenamento em cache?

O armazenamento em cache é apenas a prática de armazenar dados temporariamente e recuperar dados de um armazenamento de alto desempenho (geralmente memória), de forma explícita ou implícita.

Quando um mecanismo de armazenamento em cache está em vigor, ele ajuda a melhorar a velocidade de entrega armazenando uma cópia do ativo que você solicitou e depois acessando a cópia em cache em vez do original.

32. Quais as melhores práticas a serem seguidas ao projetar serviços da Web RESTful?

Para criar um serviço Web RESTful seguro, existem algumas práticas recomendadas ou alguns pontos que devem ser considerados. Estes são explicados da seguinte forma:

- Cada entrada no servidor deve ser validada.
- A entrada deve ser bem formada.
- Nunca passe dados confidenciais por meio de URL.
- Para qualquer sessão, o usuário deve ser autenticado.
- Apenas mensagens de erro HTTP devem ser usadas para indicar qualquer falha.
- Use o formato de mensagem que seja facilmente entendido e seja exigido pelo cliente.
- O Identificador de Recurso Unificado deve ser descritivo e de fácil compreensão.

33. O que são códigos de status HTTP? Dê alguns exemplos.

Os códigos de status HTTP são basicamente a representação do status da tarefa que foi executada no servidor, com o modo de alguns códigos. Todo código tem seu próprio significado.

Alguns dos códigos de status HTTP com seus significados são os seguintes:

- Código 200: isso indica sucesso.
- Código 201: indica que o recurso foi criado com sucesso.
- Código 204: indica que não há conteúdo no corpo da resposta.
- Código 404: Isso indica que não há nenhum método disponível.

34. Descreva os principais características dos web services RESTful.

Todos os serviços da Web RESTful devem ter os seguintes recursos e características :

- Ser baseado na representação do servidor do cliente.
- Usar o protocolo HTTP para executar funções como buscar dados do serviço da web, recuperar recursos, executar qualquer consulta, etc.
- A comunicação entre o servidor e o cliente é realizada através do meio conhecido como "mensagens".
- Realizar o endereçamento de recursos disponíveis no servidor por meio de URIs.
- Ser baseado no conceito sem estado, onde cada solicitação do cliente e a resposta são independentes da outra, com garantia total de fornecer as informações necessárias.
- Usar o conceito de armazenamento em cache.
- Funcionar na interface Uniforme.

35. Enumere as diferenças entre SOAP e REST

SOAP:

SOAP é um protocolo através do qual dois computadores se comunicam compartilhando documento XML
SOAP permite apenas XML
Leituras baseadas em SOAP não podem ser armazenadas em cache
SOAP é como um aplicativo de desktop personalizado, intimamente conectado ao servidor
SOAP é mais lento que REST
Ele é executado em HTTP, mas envelopa a mensagem

REST :

Rest é uma arquitetura de serviço e design para arquiteturas de software baseadas em rede
O REST suporta muitos formatos de dados diferentes

WEB API – Entrevista : Questões e Respostas

Leituras de REST podem ser armazenadas em cache

Um cliente REST é mais parecido com um navegador; ele sabe como padronizar métodos e um aplicativo tem que caber dentro dele

REST é mais rápido que o SOAP

Ele usa os cabeçalhos HTTP para conter informações meta