Atividade A1 - Parte 2

Academico:	**José A. Q. C. Gomes** `@JoseComparotto`	**RA: 398439413098**
Curso:	Ciência da Computação	Noturno
Professor:	Luiz Augusto Rodrigues `@profluizao`	
Disciplina:	Programação Orientada a Objetos II	2024.2 - 6º Semestre
Universidade:	Anhanguera-Uniderp - Matriz	Campo Grande, MS
Atividade:	Atividade A1 - Parte 2	Data: 19/08/2024

Gabarito

Questão	**Resposta**	**Justificativa**
1	C	De acordo com a documentação da MDN Web Docs (2023), o HTTP (Hypertext Transfer Protocol) opera na camada de aplicação do modelo OSI e utiliza o modelo cliente-servidor para a troca de mensagens.
2	C	De acordo com a MDN Web Docs, "Em respostas, o Content-Type diz para o cliente qual é o tipo de conteúdo que a resposta, de fato, tem" (Mozilla Contributors, 2023).
3	B	Embora o HTTP opere na camada de aplicação, o HTTPS utiliza TLS/SSL para garantir a segurança da comunicação, e esse processo ocorre entre a camada de transporte (TCP) e a camada de aplicação. De acordo com a MDN Web Docs, "HTTPS (HTTP Secure) é uma versão do protocolo HTTP criptografado. É normalmente usado SSL ou TLS para criptografar toda a comunicação entre um cliente e um servidor" (Mozilla Contributors, 2023).
4	D	De acordo com a MDN Web Docs, "O código HTTP 200 OK é a resposta de status de sucesso que indica que a requisição foi bem sucedida." (Mozilla Contributors, 2023).
5	A	O código de status HTTP 400 Bad Request é apropriado quando a solicitação do cliente está malformada ou falta dados essenciais, e o servidor não consegue processar a solicitação devido a essas falhas. Segundo a MDN Web Docs, "O código de status HTTP 400 Bad Request indica que o servidor não pode ou não irá processar a solicitação devido a um erro do cliente" (Mozilla Contributors, 2023).
6	B	Segundo a MDN Web Docs, "O código de resposta de status de redirecionamento 301 Moved Permanently do protocolo HTTP indica que o recurso requisitado foi movido permanentemente para a URL dada pelo cabeçalho Localização headers." (Mozilla Contributors, 2023).

Questão	Resposta	Justificativa
7	A	O método HTTP PUT é utilizado para enviar uma nova representação de um recurso existente ou criar um novo recurso em uma URL específica, substituindo a representação anterior. Por outro lado, o método POST é utilizado para criar um novo recurso no servidor, geralmente enviando dados que serão processados para criar uma nova entrada. De acordo com a MDN Web Docs, "O método de requisição HTTP PUT cria um novo recurso ou substitui uma representação do recurso de destino com os novos dados. A diferença entre PUT e POST é que PUT é idempotente, ou seja, chamá-lo várias vezes terá o mesmo efeito" (Mozilla Contributors, 2023).
8	D	De acordo com a MDN Web Docs, "O método HTTP HEAD solicita os cabeçalhos retornados de um recurso específico que foi requisitado por um método HTTP GET. Tal solicitação pode ser feita antes de baixar um grande recurso para economizar largura de banda, por exemplo. Uma resposta para um método HEAD não deve ter um corpo. Se tiver, deve ser ignorado." (Mozilla Contributors, 2023).
9	D	De acordo com a MDN Web Docs, "O método HTTP POST envia dados ao servidor. [...] Uma solicitação POST geralmente é enviada por meio de um formulário HTML e resulta em uma alteração no servidor." (Mozilla Contributors, 2023).
10	**N.D.A.**	A RFC 5789 especifica que: > The PATCH method requests that a set of changes described in the request entity be applied to the resource identified by the Request-URI. The set of changes is represented in a format called a 'patch document' identified by a media type. [...] PATCH is neither safe nor idempotent as defined by [RFC2616], Section 9.1. A PATCH request can be issued in such a way as to be idempotent, which also helps prevent bad outcomes from collisions between two PATCH requests on the same resource in a similar time frame. (RFC 5789, Section 2). Portanto, a afirmação correta seria: **PATCH** é utilizado para aplicar modificações parciais a um recurso. Apesar de não ser garantidamente idempotente, o método pode ser implementado de forma a ser idempotente, mas isso não é uma exigência obrigatória. **PUT** é utilizado para substituir um recurso inteiro e é garantidamente idempotente, significando que múltiplas requisições PUT idênticas terão o mesmo efeito que uma única requisição. Logo, com base na descrição fornecida, **nenhuma das alternativas é completamente correta.** A questão pode ser considerada ambígua e pode precisar ser revisada ou anulada, pois a descrição se refere a um comportamento de PATCH que não é garantido por padrão.

Questão	Resposta	Justificativa
11	C	De acordo com a MDN Web Docs, "O método de requisição HTTP PATCH aplica modificações parciais a um recurso. O método HTTP PUT permite apenas substituições completas de um documento."(Mozilla Contributors, 2023).
12	B	Segundo Fielding (2000), no estilo arquitetural "client-stateless-server" (CSS), que é derivado do modelo cliente-servidor, o servidor não armazena o estado da sessão entre as requisições. Isso significa que cada solicitação enviada pelo cliente ao servidor deve conter todas as informações necessárias para ser processada, sem depender de qualquer contexto previamente armazenado no servidor. O estado da sessão é mantido exclusivamente no lado do cliente. Fielding explica: > The client-stateless-server style derives from client-server with the additional constraint that no session state is allowed on the server component. Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is kept entirely on the client. (Fielding, 2000, p. 47)
13	A & C	Ao projetar endpoints para uma API RESTful, é crucial seguir práticas que assegurem uma arquitetura clara e eficiente, conforme os princípios estabelecidos por Roy Fielding em sua dissertação. Fielding define um recurso como um "mapeamento conceitual para um conjunto de entidades", e a semântica desse mapeamento deve permanecer constante para garantir a distinção entre recursos (Fielding, 2000). Neste contexto, a utilização de métodos HTTP distintos para operações CRUD em um único endpoint (Alternativa A) e a adoção de URLs hierárquicas para representar recursos e suas coleções (Alternativa C) são práticas que alinham-se com as diretrizes de Fielding. O uso de métodos HTTP apropriados para diferentes operações permite uma interação clara e semântica com os recursos, enquanto URLs hierárquicas proporcionam uma estrutura organizada e intuitiva para a identificação dos recursos e suas coleções.
14	C	De acordo com o Modelo de Maturidade de Richardson (2008), APIs no Nível 1 são caracterizadas por utilizar um único método HTTP, frequentemente POST, e um único endpoint para todas as operações. Nesse nível, a API não explora amplamente o uso dos métodos HTTP para diferenciar operações (como GET, PUT, DELETE) nem adota URLs hierárquicas para representar diferentes recursos. Essas práticas são desenvolvidas nos níveis seguintes do modelo, que introduzem uma estrutura mais detalhada e uma representação mais clara dos recursos.

Questão	Resposta	Justificativa
15	D	A limitação das APIs no Nível 1 do Modelo de Maturidade de Richardson (2008) está no uso inadequado dos métodos HTTP e das URLs para representar operações em recursos. Nessa abordagem, é comum que as operações de CRUD sejam realizadas utilizando um único método HTTP (como POST) para todas as ações, ou que diferentes operações sejam mapeadas para diferentes URLs (exemplo: /produtos/update para atualização), o que viola as práticas RESTful.
16	C	O Nível 2 do Modelo de Maturidade de Richardson se distingue do Nível 1 pela adoção correta dos métodos HTTP (GET, POST, PUT, DELETE) para suas respectivas operações e pelo uso de códigos de status HTTP adequados. No Nível 1, as APIs podem até usar diferentes métodos HTTP, mas geralmente de forma inconsistente ou inadequada, e com URLs que não seguem padrões RESTful. No Nível 2, essas práticas são ajustadas para aderir aos princípios REST, proporcionando uma interface mais clara e semântica.
17	B	No Nível 2 do Modelo de Maturidade de Richardson, a API se diferencia do Nível 1 principalmente pela adoção correta dos métodos HTTP (GET, POST, PUT, DELETE), que são aplicados conforme a semântica de cada operação. Além disso, as URLs são estruturadas de maneira a refletir a hierarquia e a natureza dos recursos, facilitando a identificação e manipulação dos mesmos. No Nível 1, essa distinção não é feita de forma consistente, resultando em uma API menos clara e padronizada.
18	C	No Nível 2 do Modelo de Maturidade de Richardson, o uso de métodos HTTP distintos melhora significativamente a clareza e a semântica das operações na API. Cada método HTTP (GET, POST, PUT, DELETE) está claramente associado a uma operação específica sobre os recursos, o que não só facilita o entendimento do comportamento da API como também promove uma interação mais intuitiva e alinhada aos princípios REST. Isso contrasta com o uso de um único método HTTP para todas as operações, que pode levar a ambiguidades e a uma estrutura menos organizada.
19	B	No Nível 3 do Modelo de Maturidade de Richardson, a principal característica que diferencia esse nível é a implementação de HATEOAS. Isso permite que a API forneça links dinâmicos nas respostas, facilitando a navegação e descoberta de recursos pelo cliente, sem depender de conhecimento prévio da API. Essa abordagem torna a API mais autossuficiente e verdadeiramente RESTful.
20	C	A principal vantagem de HATEOAS é permitir que o cliente descubra e interaja dinamicamente com recursos e ações disponíveis, utilizando os links fornecidos na resposta da API. Isso torna a API mais flexível e autoexplicativa, reduzindo a necessidade de o cliente ter um conhecimento prévio fixo sobre a estrutura da API.

Referências

- Mozilla Contributors. (2023). HTTP. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP>. Acesso em 24 de Agosto de 2024.
- Mozilla Contributors. (2023). Content-Type. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/Content-Type>. Acesso em 24 de Agosto de 2024.
- Mozilla Contributors. (2023). HTTPS. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Glossary/HTTPS>. Acesso em 24 de Agosto de 2024.
- Mozilla Contributors. (2023). HTTP Status 200. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/200>. Acesso em 24 de Agosto de 2024.
- Mozilla Contributors. (2023). HTTP Status 400. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/400>. Acesso em 24 de Agosto de 2024.
- Mozilla Contributors. (2023). HTTP Status 301. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/301>. Acesso em 24 de Agosto de 2024.
- Mozilla Contributors. (2023). HTTP Method PUT. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/PUT>. Acesso em 24 de Agosto de 2024.
- Mozilla Contributors. (2023). HTTP Method HEAD. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/HEAD>. Acesso em 24 de Agosto de 2024.
- Mozilla Contributors. (2023). HTTP Method POST. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/POST>. Acesso em 24 de Agosto de 2024.
- RFC 5789 - PATCH Method for HTTP. (2010). Disponível em: <https://datatracker.ietf.org/doc/html/rfc5789#section-2>. Acesso em 24 de Agosto de 2024.
- Mozilla Contributors. (2023). HTTP Method PATCH. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/PATCH>. Acesso em 26 de Agosto de 2024.
- Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>. Acesso em 26 de Agosto de 2024.
- Richardson, Leonard (2008). "Justice Will Take Us Millions of Intricate Moves". Act Three: The Maturity Heuristic. Disponível em: <https://www.crummy.com/writing/speaking/2008-QCon/act3.html>. Acesso em 26 de Agosto de 2024.

Questões

Questão 01

Qual das seguintes afirmativas sobre o protocolo HTTP (Hypertext Transfer Protocol) está correta?

Alternativas

- (A) HTTP é um protocolo de comunicação que funciona apenas na camada de rede do modelo OSI.
- (B) O HTTP permite a comunicação entre cliente e servidor usando pacotes de dados criptografados por padrão.

- ***(C) O HTTP utiliza o modelo cliente-servidor para a troca de mensagens e opera tipicamente na camada de aplicação do modelo OSI.***
- (D) O HTTP é um protocolo que requer uma conexão constante e de longa duração entre cliente e servidor.
- (E) O HTTP é um protocolo orientado à conexão que garante a entrega de pacotes na ordem correta.

Resposta

```
(C) O HTTP utiliza o modelo cliente-servidor para a troca de mensagens e opera
tipicamente na camada de aplicação do modelo OSI.
```

Justificativa

```
De acordo com a documentação da MDN Web Docs (2023), o HTTP (Hypertext Transfer
Protocol) opera na camada de aplicação do modelo OSI e utiliza o modelo
cliente-servidor para a troca de mensagens.
```

Questão 02

Qual dos seguintes cabeçalhos HTTP é usado para especificar o tipo de mídia do corpo da resposta, informando ao cliente o formato dos dados recebidos?

Alternativas

- (A) Content-Length
- (B) Accept
- ***(C) Content-Type***
- (D) Authorization
- (E) Location

Resposta

```
(C) Content-Type
```

Justificativa

```
De acordo com a MDN Web Docs, "Em respostas, o Content-Type diz para o cliente
qual é o tipo de conteúdo que a resposta, de fato, tem" (Mozilla Contributors,
```

```
2023).
```

Questão 03

Qual das seguintes afirmações sobre HTTPS (Hypertext Transfer Protocol Secure) é verdadeira?

Alternativas

- (A) HTTPS é uma extensão do HTTP que utiliza a criptografia AES para proteger a comunicação, mas não fornece autenticação do servidor.
- ***(B) HTTPS opera sobre a camada de transporte do modelo OSI e utiliza criptografia SSL/TLS para garantir a segurança da comunicação entre cliente e servidor.***
- (C) HTTPS é um protocolo que utiliza a criptografia somente para o armazenamento de dados no servidor, e não para a transmissão de dados entre cliente e servidor.
- (D) HTTPS é um protocolo de comunicação que é mais rápido que HTTP porque elimina a necessidade de criptografia e autenticação.
- (E) HTTPS não é compatível com certificados digitais e, portanto, não pode validar a identidade do servidor.

Resposta

```
(B) HTTPS opera sobre a camada de transporte do modelo OSI e utiliza
criptografia SSL/TLS para garantir a segurança da comunicação entre cliente e
servidor.
```

Justificativa

```
Embora o HTTP opere na camada de aplicação, o HTTPS utiliza TLS/SSL para
garantir a segurança da comunicação, e esse processo ocorre entre a camada de
transporte (TCP) e a camada de aplicação. De acordo com a MDN Web Docs, "HTTPS
(HTTP Secure) é uma versão do protocolo HTTP criptografado. É normalmente usado
SSL ou TLS para criptografar toda a comunicação entre um cliente e um servidor"
(Mozilla Contributors, 2023).
```

Questão 04

Qual dos seguintes códigos de status HTTP indica que a solicitação foi bem-sucedida e que a resposta contém a representação solicitada do recurso?

Alternativas

- (A) 301 Moved Permanently

- (B) 404 Not Found
- (C) 500 Internal Server Error
- ***(D) 200 OK***
- (E) 401 Unauthorized

Resposta

```
(D) 200 OK
```

Justificativa

```
De acordo com a MDN Web Docs, "O código HTTP 200 OK é a resposta de status de
sucesso que indica que a requisição foi bem sucedida." (Mozilla Contributors,
2023).
```

Questão 05

Qual dos seguintes códigos de status HTTP é apropriado para uma resposta quando um cliente envia uma solicitação que está faltando dados essenciais, e o servidor não pode processar a solicitação devido a essa falta de informações?

Alternativas

- ***(A) 400 Bad Request***
- (B) 403 Forbidden
- (C) 422 Unprocessable Entity
- (D) 405 Method Not Allowed
- (E) 409 Conflict

Resposta

```
(A) 400 Bad Request
```

Justificativa

```
O código de status HTTP 400 Bad Request é apropriado quando a solicitação do
cliente está malformada ou falta dados essenciais, e o servidor não consegue
```

```
processar a solicitação devido a essas falhas. Segundo a MDN Web Docs, "O
código de status HTTP 400 Bad Request indica que o servidor não pode ou não irá
processar a solicitação devido a um erro do cliente" (Mozilla Contributors,
2023).
```

Questão 06

Em um cenário onde um cliente tenta acessar um recurso que foi removido e o servidor deseja informar ao cliente que o recurso foi movido permanentemente para uma nova URL, qual código de status HTTP deve ser retornado?

Alternativas

- (A) 410 Gone
- ***(B) 301 Moved Permanently***
- (C) 302 Found
- (D) 307 Temporary Redirect
- (E) 403 Forbidden

Resposta

```
(B) 301 Moved Permanently
```

Justificativa

```
Segundo a MDN Web Docs, "O código de resposta de status de redirecionamento 301
Moved Permanently do protocolo HTTP indica que o recurso requisitado foi movido
permanentemente para a URL dada pelo cabeçalho Localização headers."
(Mozilla Contributors, 2023).
```

Questão 07

No contexto do protocolo HTTP, qual das seguintes afirmações é verdadeira sobre os métodos de requisição e suas respectivas semânticas?

Alternativas

- ***(A) O método HTTP PUT é utilizado para enviar uma nova representação de um recurso existente, e o método POST é utilizado para criar um novo recurso no servidor.***
- (B) O método HTTP GET é usado para enviar dados ao servidor, enquanto o método DELETE é utilizado para recuperar informações de um recurso especificado

- (C) O método HTTP PATCH é utilizado para substituir completamente o recurso de destino, enquanto o método OPTIONS é usado para modificar parcialmente um recurso.
- (D) O método HTTP HEAD é semelhante ao método GET, mas não retorna o corpo da resposta, enquanto o método PUT é utilizado para atualizar parcialmente um recurso no servidor.
- (E) O método HTTP TRACE é utilizado para depurar a comunicação entre cliente e servidor, retornando o conteúdo da requisição no corpo da resposta, enquanto o método PUT é usado para enviar uma nova representação de um recurso ao servidor.

Resposta

```
(A) O método HTTP PUT é utilizado para enviar uma nova representação de um
recurso existente, e o método POST é utilizado para criar um novo recurso no
servidor.
```

Justificativa

```
O método HTTP PUT é utilizado para enviar uma nova representação de um recurso
existente ou criar um novo recurso em uma URL específica, substituindo a
representação anterior. Por outro lado, o método POST é utilizado para criar um
novo recurso no servidor, geralmente enviando dados que serão processados para
criar uma nova entrada. De acordo com a MDN Web Docs, "O método de requisição
HTTP PUT cria um novo recurso ou substitui uma representação do recurso de
destino com os novos dados. A diferença entre PUT e POST é que PUT é
idempotente, ou seja, chamá-lo várias vezes terá o mesmo efeito"(Mozilla
Contributors, 2023).
```

Questão 08

Qual dos seguintes métodos HTTP é utilizado para obter informações sobre um recurso sem modificar seu estado e é semelhante ao método GET, mas não inclui o corpo da resposta?

Alternativas

- (A) PUT
- (B) POST
- (C) OPTIONS
- ***(D) HEAD***
- (E) PATCH

Resposta

```
(D) HEAD
```

Justificativa

```
De acordo com a MDN Web Docs, "O método HTTP HEAD solicita os cabeçalhos
retornados de um recurso específico que foi requisitado por um método HTTP GET.
Tal solicitação pode ser feita antes de baixar um grande recurso para
economizar largura de banda, por exemplo. Uma resposta para um método HEAD não
deve ter um corpo. Se tiver, deve ser ignorado." (Mozilla Contributors, 2023).
```

Questão 09

Qual dos seguintes métodos HTTP é projetado para enviar dados ao servidor para criar um novo recurso e, geralmente, deve ser usado quando se deseja submeter dados para processamento, como em um formulário de web?

Alternativas

- (A) GET
- (B) DELETE
- (C) PUT
- ***(D) POST***
- (E) OPTIONS

Resposta

```
(D) POST
```

Justificativa

```
De acordo com a MDN Web Docs, "O método HTTP POST envia dados ao servidor.
[...] Uma solicitação POST geralmente é enviada por meio de um formulário HTML
e resulta em uma alteração no servidor." (Mozilla Contributors, 2023).
```

Questão 10

Considere os seguintes métodos HTTP. Qual deles é corretamente descrito pela seguinte afirmação: "Este método é utilizado para aplicar modificações parciais a um recurso existente e deve ser idempotente, significando que realizar a mesma operação múltiplas vezes deve resultar no mesmo estado final do recurso"?

Alternativas

- (A) PUT
- (B) POST
- (C) DELETE
- (D) PATCH
- (E) OPTIONS

Resposta

```
(N.D.A.) Nenhuma das Alternativas
```

Justificativa

```
A RFC 5789 especifica que:

    "The PATCH method requests that a set of changes described in the request
    entity be applied to the resource identified by the Request-URI. The set of
    changes is represented in a format called a 'patch document' identified by
    a media type. [...] PATCH is neither safe nor idempotent as defined by
    [RFC2616], Section 9.1. A PATCH request can be issued in such a way as to
    be idempotent, which also helps prevent bad outcomes from collisions
    between two PATCH requests on the same resource in a similar time frame."
    (RFC 5789, Section 2).

Portanto, a afirmação correta seria:

    PATCH é utilizado para aplicar modificações parciais a um recurso. Apesar
    de não ser garantidamente idempotente, o método pode ser implementado de
    forma a ser idempotente, mas isso não é uma exigência obrigatória.

    PUT é utilizado para substituir um recurso inteiro e é garantidamente
    idempotente, significando que múltiplas requisições PUT idênticas terão o
    mesmo efeito que uma única requisição.

Logo, com base na descrição fornecida, nenhuma das alternativas é completamente
correta. A questão pode ser considerada ambígua e pode precisar ser revisada ou
anulada, pois a descrição se refere a um comportamento de PATCH que não é
garantido por padrão.
```

Questão 11

Considere os métodos HTTP a seguir e suas descrições. Qual afirmação corretamente descreve a diferença entre os métodos PUT e PATCH em termos de suas operações e características?

Alternativas

- (A) O método PUT é usado para criar um novo recurso em uma localização especificada, enquanto o método PATCH substitui completamente o recurso existente com uma nova representação.
- (B) O método PATCH é utilizado para substituir um recurso completo no servidor, enquanto o método PUT é usado para enviar uma modificação parcial ao recurso existente.
- ***(C) O método PUT substitui completamente o recurso existente com uma nova representação fornecida pelo cliente, enquanto o método PATCH aplica alterações parciais ao recurso existente.***
- (D) O método PATCH cria um novo recurso no servidor com base nos dados enviados, enquanto o método PUT é usado para recuperar informações sobre um recurso sem alterá-lo.
- (E) O método PUT retorna o estado atual do recurso após a modificação, enquanto o método PATCH não fornece nenhum feedback sobre o estado do recurso após a operação.

Resposta

```
(C) O método PUT substitui completamente o recurso existente com uma nova
representação fornecida pelo cliente, enquanto o método PATCH aplica
alterações parciais ao recurso existente.
```

Justificativa

```
De acordo com a MDN Web Docs, "O método de requisição HTTP PATCH aplica
modificações parciais a um recurso.O método HTTP PUT permite apenas
substituições completas de um documento."(Mozilla Contributors, 2023).
```

Questão 12

Qual das seguintes afirmações descreve corretamente o conceito de "stateless" em uma API RESTful?

Alternativas

- (A) Em uma API RESTful, "stateless" significa que o servidor armazena o estado da sessão do cliente entre diferentes requisições.
- ***(B) Em uma API RESTful, "stateless" implica que o cliente deve enviar todas as informações necessárias para processar a requisição em cada solicitação, pois o servidor não armazena nenhum estado entre as requisições.***
- (C) Em uma API RESTful, "stateless" significa que o servidor mantém o estado global da aplicação, mas não o estado específico do cliente.
- (D) Em uma API RESTful, "stateless" permite que o servidor armazene dados temporários para melhorar o desempenho das requisições frequentes do mesmo cliente.

- (E) Em uma API RESTful, "stateless" indica que a comunicação entre cliente e servidor é realizada usando uma conexão persistente durante a duração da sessão.

Resposta

```
(B) Em uma API RESTful, "stateless" implica que o cliente deve enviar todas as
informações necessárias para processar a requisição em cada solicitação,
pois o servidor não armazena nenhum estado entre as requisições.
```

Justificativa

```
Segundo Fielding (2000), no estilo arquitetural "client-stateless-server"
(CSS), que é derivado do modelo cliente-servidor, o servidor não armazena o
estado da sessão entre as requisições. Isso significa que cada solicitação
enviada pelo cliente ao servidor deve conter todas as informações necessárias
para ser processada, sem depender de qualquer contexto previamente armazenado
no servidor. O estado da sessão é mantido exclusivamente no lado do cliente.
Fielding explica:
    "The client-stateless-server style derives from client-server with the
    additional constraint that no session state is allowed on the server
    component. Each request from client to server must contain all of the
    information necessary to understand the request, and cannot take
    advantage of any stored context on the server. Session state is kept
    entirely on the client." (Fielding, 2000, p. 47)
```

Questão 13

Qual das seguintes características é uma boa prática recomendada ao projetar endpoints em uma API RESTful?

Alternativas

- ***(A) Utilizar métodos HTTP diferentes para operações CRUD em um único endpoint.***
- (B) Usar URLs dinâmicas e variáveis de consulta para criar endpoints de recursos, permitindo flexibilidade ilimitada na estrutura dos recursos.
- ***(C) Projetar endpoints que representam recursos e suas coleções usando URLs hierárquicas e manter a consistência no uso dos métodos HTTP.***
- (D) Permitir que todos os endpoints aceitem e retornem todos os formatos de dados possíveis, sem especificar um formato padrão.
- (E) Implementar um sistema de autenticação e autorização dentro dos próprios endpoints de recursos, ao invés de utilizar um mecanismo centralizado.

Resposta

```
(A) Utilizar métodos HTTP diferentes para operações CRUD em um único
endpoint.
(C) Projetar endpoints que representam recursos e suas coleções usando URLs
hierárquicas e manter a consistência no uso dos métodos HTTP.
```

Justificativa

```
Ao projetar endpoints para uma API RESTful, é crucial seguir práticas que
assegurem uma arquitetura clara e eficiente, conforme os princípios
estabelecidos por Roy Fielding em sua dissertação. Fielding define um recurso
como um "mapeamento conceitual para um conjunto de entidades", e a semântica
desse mapeamento deve permanecer constante para garantir a distinção entre
recursos (Fielding, 2000).

Neste contexto, a utilização de métodos HTTP distintos para operações CRUD em
um único endpoint (Alternativa A) e a adoção de URLs hierárquicas para
representar recursos e suas coleções (Alternativa C) são práticas que
alinham-se com as diretrizes de Fielding. O uso de métodos HTTP apropriados
para diferentes operações permite uma interação clara e semântica com os
recursos, enquanto URLs hierárquicas proporcionam uma estrutura organizada e
intuitiva para a identificação dos recursos e suas coleções.
```

Questão 14

No contexto do Modelo de Maturidade de Richardson, qual das seguintes características é típica de uma API que está no Nível 1 de maturidade?

Alternativas

- (A) A API utiliza múltiplos métodos HTTP (GET, POST, PUT, DELETE) e fornece URLs hierárquicas para representar diferentes recursos.
- (B) A API retorna códigos de status HTTP apropriados para indicar o resultado das operações e utiliza URLs para representar recursos e suas coleções.
- ***(C) A API faz uso de um único método HTTP (geralmente POST) e um único endpoint para todas as operações, sem distinguir entre diferentes tipos de operações.***
- (D) A API implementa HATEOAS, fornecendo links dinâmicos que permitem ao cliente descobrir e navegar entre diferentes recursos e operações.
- (E) A API permite a autenticação e autorização baseada em tokens e implementa suporte para consultas avançadas e paginação de resultados.

Resposta

```
(C) A API faz uso de um único método HTTP (geralmente POST) e um único
endpoint para todas as operações, sem distinguir entre diferentes tipos de
operações.
```

Justificativa

```
De acordo com o Modelo de Maturidade de Richardson (2008), APIs no Nível 1
são caracterizadas por utilizar um único método HTTP, frequentemente POST,
para todas as operações. Nesse nível, a API não explora amplamente o uso dos
métodos HTTP para diferenciar operações (como GET, PUT, DELETE) nem adota
URLs hierárquicas para representar diferentes recursos. Essas práticas são
desenvolvidas nos níveis seguintes do modelo, que introduzem uma estrutura
mais detalhada e uma representação mais clara dos recursos.
```

Questão 15

Qual é uma limitação comum das APIs que operam no Nível 1 do Modelo de Maturidade de Richardson?

Alternativas

- (A) Falta de suporte para operações de CRUD (Criar, Ler, Atualizar, Excluir) através de métodos HTTP distintos e URLs específicas.
- (B) Implementação de HATEOAS, fornecendo links dinâmicos que permitem ao cliente navegar entre diferentes recursos e operações.
- (C) Uso de múltiplos métodos HTTP para operações diferentes e fornecimento de URLs hierárquicas para representar recursos e suas coleções.
- ***(D) Utilização de URLs genéricas e um único método HTTP para todas as operações, que pode dificultar a identificação e a manipulação específica dos recursos.***
- (E) Retorno de códigos de status HTTP apropriados e consistentes para indicar o resultado das operações.

Resposta

```
(D) Utilização de URLs genéricas e um único método HTTP para todas as
operações, que pode dificultar a identificação e a manipulação específica dos
recursos.
```

Justificativa

```
A limitação das APIs no Nível 1 do Modelo de Maturidade de Richardson (2008)
está no uso inadequado dos métodos HTTP e das URLs para representar operações
em recursos. Nessa abordagem, é comum que as operações de CRUD sejam
realizadas utilizando um único método HTTP (como POST) para todas as ações,
ou que diferentes operações sejam mapeadas para diferentes URLs (exemplo:
/produtos/update para atualização), o que viola as práticas RESTful.
```

Questão 16

No Modelo de Maturidade de Richardson para APIs RESTful, qual é a principal característica que distingue o Nível 2 do Nível 1?

Alternativas

- (A) No Nível 2, a API usa URLs bem definidas para recursos e métodos HTTP, enquanto no Nível 1 a API utiliza URLs genéricas e um único método HTTP.
- (B) No Nível 2, a API implementa autenticação e autorização baseada em tokens, enquanto no Nível 1 a API não utiliza nenhum mecanismo de segurança.
- ***(C) No Nível 2, a API utiliza métodos HTTP corretamente (GET, POST, PUT, DELETE) e usa códigos de status HTTP apropriados para indicar o resultado das operações, enquanto no Nível 1 a API faz uso de um único método HTTP para todas as operações.***
- (D) No Nível 2, a API utiliza apenas um formato de resposta, enquanto no Nível 1 a API é capaz de lidar com múltiplos formatos de resposta, como JSON e XML.
- (E) No Nível 2, a API implementa suporte para consultas avançadas e paginação de resultados, enquanto no Nível 1 a API não oferece suporte a essas funcionalidades.

Resposta

```
(C) No Nível 2, a API utiliza métodos HTTP corretamente (GET, POST, PUT,
DELETE) e usa códigos de status HTTP apropriados para indicar o resultado das
operações, enquanto no Nível 1 a API faz uso de um único método HTTP para
todas as operações.
```

Justificativa

```
O Nível 2 do Modelo de Maturidade de Richardson se distingue do Nível 1 pela
adoção correta dos métodos HTTP (GET, POST, PUT, DELETE) para suas
respectivas operações e pelo uso de códigos de status HTTP adequados. No
Nível 1, as APIs podem até usar diferentes métodos HTTP, mas geralmente de
forma inconsistente ou inadequada, e com URLs que não seguem padrões RESTful.
No Nível 2, essas práticas são ajustadas para aderir aos princípios REST,
proporcionando uma interface mais clara e semântica.
```

Questão 17

No Nível 2 do Modelo de Maturidade de Richardson, qual das seguintes práticas é uma característica distintiva que separa esse nível do Nível 1?

Alternativas

- (A) A API utiliza um único método HTTP (geralmente POST) para todas as operações e um único endpoint para todos os recursos.
- ***(B) A API usa URLs hierárquicas para representar recursos e suas coleções e utiliza métodos HTTP distintos (GET, POST, PUT, DELETE) para diferentes tipos de operações.***
- (C) A API fornece links dinâmicos e informações de navegação para que os clientes descubram e naveguem entre diferentes recursos e operações.
- (D) A API permite a realização de consultas avançadas e paginação de resultados para otimizar a recuperação de grandes conjuntos de dados.
- (E) A API utiliza um único formato de resposta e não faz negociação de formatos com base nas preferências do cliente.

Resposta

```
(B) A API usa URLs hierárquicas para representar recursos e suas coleções e
utiliza métodos HTTP distintos (GET, POST, PUT, DELETE) para diferentes tipos
de operações.
```

Justificativa

```
No Nível 2 do Modelo de Maturidade de Richardson, a API se diferencia do
Nível 1 principalmente pela adoção correta dos métodos HTTP (GET, POST, PUT,
DELETE), que são aplicados conforme a semântica de cada operação. Além disso,
as URLs são estruturadas de maneira a refletir a hierarquia e a natureza dos
recursos, facilitando a identificação e manipulação dos mesmos. No Nível 1,
essa distinção não é feita de forma consistente, resultando em uma API menos
clara e padronizada.
```

Questão 18

No Nível 2 do Modelo de Maturidade de Richardson, qual é a principal vantagem do uso de métodos HTTP distintos (GET, POST, PUT, DELETE) em comparação com o uso de um único método HTTP para todas as operações?

Alternativas

- (A) Permite a autenticação e autorização detalhada em cada operação individual.
- (B) Facilita a definição de formatos de resposta múltiplos para diferentes tipos de dados.
- ***(C) Melhora a clareza e a semântica das operações, permitindo que cada método HTTP represente uma ação específica sobre os recursos.***
- (D) Oferece suporte a consultas avançadas e paginação automática dos resultados.
- (E) Reduz a complexidade da API ao permitir que todos os dados sejam enviados e recebidos em um único formato.

Resposta

```
(C) Melhora a clareza e a semântica das operações, permitindo que cada
método HTTP represente uma ação específica sobre os recursos.
```

Justificativa

```
No Nível 2 do Modelo de Maturidade de Richardson, o uso de métodos HTTP
distintos melhora significativamente a clareza e a semântica das operações na
API. Cada método HTTP (GET, POST, PUT, DELETE) está claramente associado a
uma operação específica sobre os recursos, o que não só facilita o
entendimento do comportamento da API como também promove uma interação mais
intuitiva e alinhada aos princípios REST. Isso contrasta com o uso de um
único método HTTP para todas as operações, que pode levar a ambiguidades e a
uma estrutura menos organizada.
```

Questão 19

No Nível 3 do Modelo de Maturidade de Richardson, qual é a principal característica que diferencia esse nível de Níveis anteriores (1 e 2)?

Alternativas

- (A) A utilização de métodos HTTP distintos (GET, POST, PUT, DELETE) para diferentes tipos de operações sobre os recursos.
- ***(B) A implementação de HATEOAS (Hypermedia as the Engine of Application State), permitindo que o cliente descubra e navegue entre os recursos dinamicamente através de links fornecidos na resposta.***
- (C) A definição de URLs hierárquicas para representar recursos e suas coleções de forma clara e organizada.
- (D) O suporte para múltiplos formatos de resposta e negociação de conteúdo com base nas preferências do cliente.

- (E) A implementação de autenticação e autorização em nível de recurso específico, utilizando tokens de segurança.

Resposta

```
(B) A implementação de HATEOAS (Hypermedia as the Engine of Application
State), permitindo que o cliente descubra e navegue entre os recursos
dinamicamente através de links fornecidos na resposta.
```

Justificativa

```
No Nível 3 do Modelo de Maturidade de Richardson, a principal característica
que diferencia esse nível é a implementação de HATEOAS. Isso permite que a
API forneça links dinâmicos nas respostas, facilitando a navegação e
descoberta de recursos pelo cliente, sem depender de conhecimento prévio da
API. Essa abordagem torna a API mais autossuficiente e verdadeiramente
RESTful.
```

Questão 20

Qual das seguintes abordagens é uma vantagem principal do uso de HATEOAS (Hypermedia as the Engine of Application State) no Nível 3 do Modelo de Maturidade de Richardson?

Alternativas

- (A) Permite que a API se adapte automaticamente a diferentes métodos HTTP para operações de CRUD.
- (B) Facilita a comunicação de recursos e suas relações através de URLs estáticas definidas previamente no cliente.
- ***(C) Oferece ao cliente a capacidade de descobrir e interagir com recursos e ações disponíveis dinamicamente, com base nos links fornecidos na resposta da API.***
- (D) Reduz a necessidade de uso de formatos de resposta variados, fixando um formato único para todas as respostas.
- (E) Implementa um sistema de cache robusto para melhorar o desempenho das requisições em APIs com grande volume de dados.

Resposta

```
(C) Oferece ao cliente a capacidade de descobrir e interagir com recursos e
ações disponíveis dinamicamente, com base nos links fornecidos na resposta da
API.
```

Justificativa

```
A principal vantagem de HATEOAS é permitir que o cliente descubra e interaja
dinamicamente com recursos e ações disponíveis, utilizando os links
fornecidos na resposta da API. Isso torna a API mais flexível e
autoexplicativa, reduzindo a necessidade de o cliente ter um conhecimento
prévio fixo sobre a estrutura da API.
```