Questionário: Networking em Flutter
Matheus Cardoso Tabosa Braga

1. Qual pacote Flutter é o principal e recomendado para emitir requisições HTTP?
A) flutter_http
B) dio
**C)** http
D) retrofit

2. Qual é o propósito principal de uma requisição HTTP GET no Flutter?
A) Enviar dados para criar um novo recurso.
**B)** Recuperar dados de uma API.
C) Atualizar dados existentes.
D) Deletar um recurso.

3. Qual é o propósito principal de uma requisição HTTP POST no Flutter?
A) Solicitar informações de um servidor.
B) Remover dados existentes.
**C)**Enviar dados para criar um novo recurso.
D) Realizar uma conexão de WebSocket.

4. Qual código de status HTTP geralmente indica uma requisição GET bem-sucedida?
A) 404 Not Found B)
500 Internal Server Error
**C)** 200 OK
D) 204 No Content

5. Para uma requisição POST que cria um novo recurso com sucesso, qual código de status HTTP é tipicamente esperado?
A) 200 OK
**B)** 201 Created
C) 400 Bad Request
D) 202 Accepted

6. Qual função Dart é usada para converter uma string JSON (corpo da resposta HTTP) em um objeto Dart utilizável?
A) jsonEncode()
B) jsonParse()
**C)** jsonDecode()
D) stringToJson()

7. Qual função Dart é usada para converter um objeto Dart em uma string JSON, tipicamente para ser enviada no corpo de uma requisição HTTP POST?
A) jsonDecode()
B) jsonStringify()
**C)** jsonEncode()
D) objectToJson()

8. As requisições HTTP em Dart/Flutter são consideradas operações de qual tipo?
A) Síncronas
B) Bloqueadoras
**C)** Assíncronas
D) Sequenciais

9. Em Dart, o resultado de uma operação assíncrona (como uma requisição HTTP) é frequentemente encapsulado em qual objeto?
A) Stream
**B)** Future
C) Observable
D) Callback

10. Onde a permissão de uso da internet deve ser declarada para aplicativos Flutter no Android para que eles possam realizar operações de rede?
A) No arquivo pubspec.yaml
B) No arquivo build.gradle
**C)** No arquivo AndroidManifest.xml
D) Diretamente no código Dart principal

11. Para aplicativos Flutter no macOS, onde o acesso à rede deve ser permitido?
A) No arquivo Info.plist
**B)** Nos arquivos *.entitlements
C) No arquivo pubspec.yaml
D) Automaticamente pelo sistema

12. Qual widget Flutter é comumente usado para construir a interface do usuário com base nos diferentes estados de um Future (carregando, com erro ou com dados)?
A) StreamBuilder
B) ListView.builder
**C)** FutureBuilder
D) StatefulBuilder

13. Para exibir listas de itens de forma eficiente no Flutter, como os resultados de uma requisição GET, qual widget é recomendado?
A) Column
B) SingleChildScrollView
C) ListView
**D)** ListView.builder

14. Qual widget Flutter pode ser usado para adicionar a funcionalidade de "puxar para atualizar" (pull to refresh) a uma lista?
A) RefreshController
**B)** RefreshIndicator
C) PullToRefreshWidget
D) ScrollRefresh

15. Ao usar Riverpod, que tipo de provedor é tipicamente usado para expor dados assíncronos, como uma lista de usuários obtida de uma API?
A) Provider
B) StateProvider
**C)** FutureProvider
D) ChangeNotifierProvider

16. Em Riverpod, qual método é usado dentro do método build de um widget para "observar" um provedor e reconstruir o widget quando o estado do provedor muda?
A) ref.read
B) ref.listen
**C)** ref.watch
D) ref.select

17. Em Riverpod, qual método é usado para interagir com um provedor e realizar uma ação, como enviar dados via um método POST, especialmente em um contexto de evento (fora do build direto)?
A) ref.watch
**B)** ref.read
C) ref.stream
D) ref.observe

18. Qual cabeçalho HTTP é comumente definido para requisições POST para informar ao servidor que o corpo da requisição é formatado como JSON?
A) Accept: application/json
B) Authorization: Bearer
C) Content-Length
**D)** Content-Type: application/json

19. Qual ferramenta online é sugerida para gerar rapidamente classes Dart (modelos) a partir de uma estrutura JSON fornecida por uma API?
A) JSON to Dart Converter
B) Postman
**C)** QuickType
D) Dart Model Generator

20. Qual método HTTP é usado para atualizar dados existentes em uma API?
A) GET
B) POST
**C)** PUT
D) DELETE

---

Gabarito e Relação com as Fontes
Aqui estão as respostas corretas para o questionário, com as referências diretas às fontes que as suportam:
1. C) http: O pacote http é a maneira mais simples e recomendada para emitir requisições HTTP, sendo suportado em todas as plataformas.

2. B) Recuperar dados de uma API: O método GET é usado para recuperar dados de uma API.
3. C) Enviar dados para criar um novo recurso: O método POST é utilizado para enviar dados e criar novos recursos em uma API.
4. C) 200 OK: Para uma requisição GET bem-sucedida, o código de status HTTP esperado é 200 OK.
5. B) 201 Created: Para uma requisição POST que cria um novo recurso com sucesso, o código de status HTTP esperado é 201 Created.
6. C) jsonDecode(): Para converter uma string JSON em um objeto Dart utilizável, a função jsonDecode() do pacote dart:convert é empregada.
7. C) jsonEncode(): Ao enviar dados via POST, o corpo da requisição geralmente precisa ser codificado em JSON usando jsonEncode().
8. C) Assíncronas: As requisições HTTP são operações assíncronas, significando que elas não bloqueiam a execução do programa.
9. B) Future: Em Dart/Flutter, o resultado de uma operação assíncrona é geralmente encapsulado em um objeto Future.
10. C) No arquivo AndroidManifest.xml: Aplicativos Android devem declarar a permissão de uso da internet no arquivo AndroidManifest.xml.
11. B) Nos arquivos *.entitlements: Aplicativos macOS devem permitir o acesso à rede nos arquivos *.entitlements relevantes.
12. C) FutureBuilder: Para exibir dados assíncronos na interface do usuário, widgets como o FutureBuilder são comumente usados, permitindo construir a UI com base nos diferentes estados do Future.
13. D) ListView.builder: Para exibir listas de itens de forma eficiente, o widget ListView.builder é eficiente, pois constrói os itens apenas conforme eles se tornam visíveis.
14. B) RefreshIndicator: O widget RefreshIndicator pode ser usado para adicionar a funcionalidade "pull to refresh" a uma lista.
15. C) FutureProvider: Sistemas de gerenciamento de estado como Riverpod oferecem provedores específicos como FutureProvider para lidar com dados assíncronos de forma mais integrada.
16. C) ref.watch: O ref.watch é usado no Riverpod para "assistir" a um provedor e reconstruir o widget quando o estado do provedor muda.
17. B) ref.read: O ref.read é usado para interagir com o provedor e enviar dados, por exemplo, ao chamar um método POST.
18. D) Content-Type: application/json: É comum definir o cabeçalho Content-Type: application/json para indicar que o corpo da requisição POST é um JSON.
19. C) QuickType: Ferramentas como o QuickType são sugeridas para gerar rapidamente classes (modelos) Dart a partir de um JSON.
20. C) PUT: O método PUT é usado para atualizar dados existentes.