Data and Backend/Networking

Aluno: João Gabriel Aguiar de Senna
Professor: Ricardo Duarte Taveira

Questionário:

1. Qual pacote do Flutter é o mais simples e amplamente recomendado para realizar requisições HTTP cross-platform (Android, iOS, macOS, Windows, Linux e web)?
A) dart:io
B) web_socket_channel
C) http
D) dart:html

2. Para qual propósito principal os WebSockets são utilizados em aplicações Flutter?
A) Para armazenar dados localmente no dispositivo.
B) Para realizar comunicação bidirecional com um servidor sem a necessidade de polling.
C) Para enviar dados de uma única vez para o servidor.
D) Para gerenciar o estado da aplicação de forma síncrona.

3. Ao implementar requisições HTTP em um aplicativo Flutter, por que é desaconselhável usar diretamente dart:io ou dart:html?
A) Eles são mais complexos de integrar com widgets Flutter.
B) Eles não oferecem suporte para JSON.
C) Essas bibliotecas são dependentes de plataforma e ligadas a uma única implementação, enquanto o pacote http é cross-platform.
D) Elas causam vazamento de memória em dispositivos móveis.

4. Qual permissão é necessária adicionar ao arquivo AndroidManifest.xml de um aplicativo Android para permitir o acesso à internet e, consequentemente, a capacidade de buscar ou enviar dados?
A) <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
B) <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
C) <uses-permission android:name="android.permission.INTERNET" />
D) <uses-permission android:name="android.permission.CAMERA" />

5. Qual é a classe fundamental do Dart para trabalhar com operações assíncronas que representa um valor ou erro que estará disponível em algum momento no futuro?
A) Stream
B) Future
C) Sync
D) Response

6. Para exibir dados assíncronos (como os obtidos de uma requisição HTTP Future) na tela do Flutter, qual widget é comumente utilizado e facilita o trabalho com fontes de dados assíncronas?
A) SynchronousBuilder
B) WidgetBuilder
C) FutureBuilder
D) StatefulBuilder

7. Ao buscar dados da internet, qual método do pacote http é utilizado para fazer uma requisição GET a uma URL específica, como https://jsonplaceholder.typicode.com/albums/1?
A) http.post()
B) http.delete()
C) http.get()
D) http.put()

8. Qual método do pacote http é utilizado para enviar dados a um servidor e criar um novo recurso (por exemplo, um novo álbum em https://jsonplaceholder.typicode.com/albums), geralmente envolvendo um corpo de requisição com dados JSON?
A) http.get()
B) http.put()
C) http.patch()
D) http.post()

9. Para remover dados de um servidor, qual método do pacote http é empregado, que requer o id do recurso a ser excluído?
A) http.get()
B) http.delete()
C) http.post()
D) http.patch()

10. Ao enviar dados (requisição POST) para o servidor, qual é o código de status HTTP que o servidor deve retornar para indicar que o recurso foi criado com sucesso, conforme o exemplo dado nas fontes?
A) 200 OK
B) 404 Not Found
C) 201 CREATED
D) 500 Internal Server Error

11. Qual das seguintes afirmações sobre o tratamento de erros em requisições HTTP é verdadeira, segundo as fontes, especialmente quando o FutureBuilder é utilizado?
A) Se o servidor não retornar um status 200 OK ou 201 CREATED, a função de requisição deve retornar null.
B) Em caso de erro (ex: 404 Not Found), a função de requisição deve lançar uma exceção para que snapshot.hasError no FutureBuilder seja verdadeiro e uma mensagem de erro possa ser exibida.

C) O FutureBuilder sempre exibe um spinner de carregamento, independentemente do status da requisição.
D) snapshot.hasData retorna true mesmo quando o servidor retorna um erro.

12. Qual é a principal vantagem de chamar métodos de API (como fetchAlbum()) dentro de initState() ou didChangeDependencies() em um StatefulWidget?
A) Permite que a chamada à API seja executada repetidamente a cada rebuild do widget.
B) Garante que o Future seja executado apenas uma vez e então armazenado em cache para reconstruções subsequentes, evitando chamadas repetidas da API no método build().
C) Reduz o tempo de inicialização do aplicativo.
D) Torna a lógica de rede mais fácil de depurar.

13. O que é um Stream no Dart, conforme descrito no contexto de comunicação com WebSockets?
A) Um tipo de Future que retorna apenas uma resposta assíncrona.
B) Uma classe que permite ouvir eventos assíncronos de uma fonte de dados e pode entregar muitos eventos ao longo do tempo, ao contrário de um Future.
C) Uma função para converter JSON em objetos Dart.
D) Um widget de interface do usuário para exibir listas.

14. Qual package é fornecido para conectar-se a um servidor WebSocket em Flutter?
A) http
B) flutter_socket
C) web_socket_channel
D) socket_io_client

15. Para enviar dados a um servidor WebSocket após estabelecer uma conexão, qual método é usado no sink fornecido pelo WebSocketChannel?
A) channel.sink.receive()
B) channel.sink.get()
C) channel.sink.send()
D) channel.sink.add()

16. Após a comunicação com um WebSocket ser concluída, qual método é usado para fechar a conexão?
A) channel.close()
B) channel.disconnect()
C) channel.sink.dispose()
D) channel.sink.close()

17. O que é o StreamSink fornecido pelo WebSocketChannel?
A) Uma classe para receber mensagens do servidor.
B) Uma maneira geral de adicionar eventos síncronos ou assíncronos a uma fonte de dados, usada para enviar mensagens ao servidor.
C) Um gerenciador de estado para WebSockets.
D) Um objeto para analisar o JSON recebido.

18. O que a ferramenta mockapi.io permite que um desenvolvedor faça, conforme mencionado nas fontes?
A) Criar bancos de dados SQL complexos.
B) Desenvolver interfaces de usuário sem código.
C) Fazer APIs RESTful básicas sem qualquer código, populando-as opcionalmente com dados fictícios.
D) Analisar o tráfego de rede em tempo real.
19. No contexto de modelagem de dados e serialização/desserialização JSON em Flutter, o que o pacote Freezed (e suas anotações como @freezed e @JsonKey) auxilia o desenvolvedor a fazer?
A) Gerar automaticamente código UI para formulários.
B) Gerenciar autenticação de usuários.
C) Gerar modelos de dados com métodos como toJson e fromJson, e lidar com convenções de nomenclatura JSON (snake_case para camelCase).
D) Conectar-se a WebSockets de forma segura.

20. Para um aplicativo macOS se comunicar com a web, qual entitlement precisa ser incluído nos arquivos *.entitlements?
A) <key>com.apple.security.network.server</key><true/>
B) <key>com.apple.security.app-sandbox</key><true/>
C) <key>com.apple.security.network.client</key><true/>
D) <key>com.apple.security.personal-information.location</key><true/>

Gabarito:

1. Qual pacote do Flutter é o mais simples e amplamente recomendado para realizar requisições HTTP cross-platform (Android, iOS, macOS, Windows, Linux e web)?
• Gabarito: C) http
• Explicação: O pacote http é a ferramenta recomendada para realizar requisições HTTP no Flutter, pois é projetado para ser cross-platform (funciona em diversas plataformas como Android, iOS, macOS, Windows, Linux e web). As bibliotecas dart:io e dart:html não são aconselhadas para requisições HTTP gerais, pois são dependentes de plataforma e ligadas a uma única implementação.

2. Para qual propósito principal os WebSockets são utilizados em aplicações Flutter?
• Gabarito: B) Para realizar comunicação bidirecional com um servidor sem a necessidade de polling.
• Explicação: Os WebSockets são empregados para estabelecer uma comunicação bidirecional e contínua entre o cliente (aplicativo Flutter) e o servidor. Diferentemente do polling, onde o cliente precisa fazer requisições repetidas, os WebSockets permitem que o servidor envie dados ao cliente a qualquer momento e vice-versa, de forma assíncrona, através de Streams.

3. Ao implementar requisições HTTP em um aplicativo Flutter, por que é desaconselhável usar diretamente dart:io ou dart:html?

• Gabarito: C) Essas bibliotecas são dependentes de plataforma e ligadas a uma única implementação, enquanto o pacote http é cross-platform.
• Explicação: É expressamente recomendado evitar o uso direto de dart:io ou dart:html para requisições HTTP porque essas bibliotecas são dependentes da plataforma específica. O pacote http, por outro lado, oferece uma solução multiplataforma e unificada para lidar com essas requisições.

4. Qual permissão é necessária adicionar ao arquivo AndroidManifest.xml de um aplicativo Android para permitir o acesso à internet e, consequentemente, a capacidade de buscar ou enviar dados?
• Gabarito: C) <uses-permission android:name="android.permission.INTERNET" />
• Explicação: Para que um aplicativo Android possa realizar acesso à internet (buscar ou enviar dados), é necessário adicionar a permissão <uses-permission android:name="android.permission.INTERNET" /> ao arquivo AndroidManifest.xml.

5. Qual é a classe fundamental do Dart para trabalhar com operações assíncronas que representa um valor ou erro que estará disponível em algum momento no futuro?
• Gabarito: B) Future
• Explicação: Em Dart, a classe Future é a representação de uma operação assíncrona que, em algum momento no futuro, resultará em um único valor ou um erro. Ela é usada para lidar com operações que podem levar tempo para serem concluídas, como requisições de rede.

6. Para exibir dados assíncronos (como os obtidos de uma requisição HTTP Future) na tela do Flutter, qual widget é comumente utilizado e facilita o trabalho com fontes de dados assíncronas?
• Gabarito: C) FutureBuilder
• Explicação: O widget FutureBuilder é a ferramenta padrão do Flutter para construir a interface do usuário com base no estado de um Future. Ele permite exibir um spinner de carregamento enquanto os dados estão sendo buscados, os dados quando disponíveis (snapshot.hasData), ou uma mensagem de erro (snapshot.hasError).

7. Ao buscar dados da internet, qual método do pacote http é utilizado para fazer uma requisição GET a uma URL específica, como https://jsonplaceholder.typicode.com/albums/1?
• Gabarito: C) http.get()
• Explicação: O método http.get() é utilizado para realizar requisições HTTP do tipo GET, que são usadas para solicitar dados de um servidor a partir de uma URL específica.

8. Qual método do pacote http é utilizado para enviar dados a um servidor e criar um novo recurso (por exemplo, um novo álbum em https://jsonplaceholder.typicode.com/albums), geralmente envolvendo um corpo de requisição com dados JSON?
• Gabarito: D) http.post()
• Explicação: Para enviar dados a um servidor e criar um novo recurso, o método comumente empregado é o http.post(). Este método geralmente inclui um corpo de requisição (request body) que contém os dados a serem enviados, muitas vezes no formato JSON.

9. Para remover dados de um servidor, qual método do pacote http é empregado, que requer o id do recurso a ser excluído?
• Gabarito: B) http.delete()
• Explicação: O método http.delete() é utilizado para remover recursos de um servidor na internet. Ele requer o identificador (ID) do recurso que se deseja excluir.
10. Ao enviar dados (requisição POST) para o servidor, qual é o código de status HTTP que o servidor deve retornar para indicar que o recurso foi criado com sucesso, conforme o exemplo dado nas fontes?
• Gabarito: C) 201 CREATED
• Explicação: Embora as fontes fornecidas explicitamente demonstrem o código de status 200 OK para uma requisição GET bem-sucedida, o padrão HTTP para indicar que um recurso foi criado com sucesso em uma requisição POST é o status 201 CREATED. O questionário, sendo parte das fontes, sugere este como o código esperado para tal operação.

11. Qual das seguintes afirmações sobre o tratamento de erros em requisições HTTP é verdadeira, segundo as fontes, especialmente quando o FutureBuilder é utilizado?
• Gabarito: B) Em caso de erro (ex: 404 Not Found), a função de requisição deve lançar uma exceção para que snapshot.hasError no FutureBuilder seja verdadeiro e uma mensagem de erro possa ser exibida.
• Explicação: As fontes indicam que, se o servidor não retornar um status de sucesso (como 200 OK para GET), a função de requisição deve lançar uma Exception. Isso permite que o FutureBuilder capture o erro através de snapshot.hasError e exiba uma mensagem apropriada ao usuário.

12. Qual é a principal vantagem de chamar métodos de API (como fetchAlbum()) dentro de initState() ou didChangeDependencies() em um StatefulWidget?
• Gabarito: B) Garante que o Future seja executado apenas uma vez e então armazenado em cache para reconstruções subsequentes, evitando chamadas repetidas da API no método build().
• Explicação: Chamar métodos de API dentro de initState() ou didChangeDependencies() garante que a operação assíncrona seja executada apenas uma vez durante o ciclo de vida inicial do widget. Isso evita que a requisição seja feita repetidamente a cada reconstrução (rebuild) do widget, otimizando o desempenho e a eficiência do aplicativo.

13. O que é um Stream no Dart, conforme descrito no contexto de comunicação com WebSockets?
• Gabarito: B) Uma classe que permite ouvir eventos assíncronos de uma fonte de dados e pode entregar muitos eventos ao longo do tempo, ao contrário de um Future.
• Explicação: No Dart, um Stream é uma sequência de eventos assíncronos que podem ser entregues ao longo do tempo. Diferente de um Future, que retorna um único resultado, um Stream pode emitir múltiplos valores ou erros, sendo ideal para comunicação contínua como a de WebSockets.

14. Qual package é fornecido para conectar-se a um servidor WebSocket em Flutter?
• Gabarito: C) web_socket_channel
• Explicação: O pacote web_socket_channel é a biblioteca fornecida no Flutter para estabelecer e gerenciar a comunicação com servidores WebSocket.

15. Para enviar dados a um servidor WebSocket após estabelecer uma conexão, qual método é usado no sink fornecido pelo WebSocketChannel?
• Gabarito: D) channel.sink.add()
• Explicação: Após estabelecer uma conexão WebSocket, para enviar dados ao servidor, utiliza-se o método add() no sink do WebSocketChannel (channel.sink.add(data)). O StreamSink é a interface geral para adicionar eventos a uma fonte de dados.

16. Após a comunicação com um WebSocket ser concluída, qual método é usado para fechar a conexão?
• Gabarito: D) channel.sink.close()
• Explicação: Para encerrar a conexão com um servidor WebSocket de forma limpa, o método utilizado é channel.sink.close().

17. O que é o StreamSink fornecido pelo WebSocketChannel?
• Gabarito: B) Uma maneira geral de adicionar eventos síncronos ou assíncronos a uma fonte de dados, usada para enviar mensagens ao servidor.
• Explicação: O StreamSink é uma interface que permite adicionar eventos (síncronos ou assíncronos) a uma fonte de dados. No contexto do WebSocketChannel, ele é especificamente utilizado para enviar mensagens do cliente para o servidor.

18. O que a ferramenta mockapi.io permite que um desenvolvedor faça, conforme mencionado nas fontes?
• Gabarito: C) Fazer APIs RESTful básicas sem qualquer código, populando-as opcionalmente com dados fictícios.
• Explicação: A ferramenta mockapi.io é descrita como um recurso que permite aos desenvolvedores criar APIs RESTful básicas sem a necessidade de escrever código, oferecendo a opção de popular essas APIs com dados fictícios para fins de teste e desenvolvimento.

19. No contexto de modelagem de dados e serialização/desserialização JSON em Flutter, o que o pacote Freezed (e suas anotações como @freezed e @JsonKey) auxilia o desenvolvedor a fazer?
• Gabarito: C) Gerar modelos de dados com métodos como toJson e fromJson, e lidar com convenções de nomenclatura JSON (snake_case para camelCase).
• Explicação: O pacote Freezed, juntamente com suas anotações (@freezed, @JsonKey), auxilia o desenvolvedor na geração automática de modelos de dados com funcionalidades como serialização (toJson) e desserialização (fromJson) JSON. Ele também pode ajudar a lidar com convenções de nomenclatura, como a conversão de snake_case (comum em JSON) para camelCase (comum em Dart).

20. Para um aplicativo macOS se comunicar com a web, qual entitlement precisa ser incluído nos arquivos
• Gabarito: C) <key>com.apple.security.network.client</key><true/>
• Explicação: Para que aplicativos macOS tenham permissão de acesso à rede e possam se comunicar com a web, é necessário incluir o entitlement <key>com.apple.security.network.client</key><true/> nos arquivos .entitlements do projeto.