
Questionário: Networking em Flutter e Dart

Instruções: Para cada questão, selecione a única opção correta.

1. Qual a principal razão para o uso de networking em aplicações mobile modernas com Flutter?

    - A) Para exibir conteúdo estático diretamente do armazenamento local [Não é a principal razão do networking]
    - B) Para realizar cálculos complexos no lado do cliente [Não é o propósito do networking]
    - C) Para buscar conteúdo dinâmico de servidores, APIs ou serviços externos
    - D) Para gerenciar o estado da aplicação offline sem acesso à internet [Networking lida com comunicação *com* servidores, não diretamente com gerenciamento de estado offline intrínseco]
2. Qual tipo de requisição HTTP é usado para enviar dados a um servidor, por exemplo, para criar um novo recurso como uma postagem de blog?

    - A) GET
    - B) PUT
    - C) DELETE
    - D) POST
3. Qual pacote é amplamente recomendado e fácil de usar para realizar requisições HTTP (como GET, POST, PUT, DELETE) em aplicações Flutter, sendo uma biblioteca baseada em Future?

    - A) `web_socket_channel`
    - B) `sqflite`
    - C) `http`
    - D) `WorkManager`
4. Como se converte uma resposta JSON (geralmente acessada via `response.body`) em um objeto Dart utilizável após uma requisição HTTP bem-sucedida?

    - A) Usando o método `http.read()`
    - B) Usando `json.decode()` da biblioteca `dart:convert`
    - C) Aplicando `await` diretamente no `response.body`
    - D) Utilizando o widget `StreamBuilder` para parsing
5. O que significa a sigla REST no contexto de REST API, que é um estilo arquitetural para comunicação entre sistemas?

    - A) Real-time Event State Transfer
    - B) Representational State Transfer
    - C) Remote Endpoint Service Type
    - D) Resource Exchange Standardized Protocol

6. Uma característica fundamental das REST APIs é serem "stateless" (sem estado). O que isso implica para a comunicação entre cliente e servidor?

    - A) Cada requisição do cliente ao servidor deve conter todas as informações necessárias para entender e processar a requisição.
    - B) O servidor mantém o estado da sessão do cliente entre as requisições, facilitando a interação.
    - C) As requisições são sempre feitas em tempo real e não podem ser armazenadas.
    - D) O cliente não precisa armazenar nenhum dado localmente, pois o servidor gerencia tudo.
7. Em Dart e Flutter, como as operações assíncronas, como requisições de rede, são tipicamente gerenciadas, especialmente quando se espera por uma resposta?

    - A) Usando as palavras-chave `Sync` e `Wait`.
    - B) Usando as palavras-chave `async` e `await` em conjunto com `Future`.
    - C) Através de um loop síncrono contínuo que bloqueia a thread principal.
    - D) Implementando `Thread.sleep()` para pausar a execução até a resposta.
8. Qual widget é frequentemente usado em Flutter para construir a interface do usuário com base em dados que são buscados de forma assíncrona, como de uma API REST?

    - A) `StatelessWidget`
    - B) `GestureDetector`
    - C) `FutureBuilder`
    - D) `Scaffold`
9. Como se pode lidar com exceções de timeout em requisições HTTP em Flutter, prevenindo esperas indefinidas por uma resposta do servidor?

    - A) Apenas ignorando o erro e permitindo que a aplicação falhe.
    - B) Adicionando um `.timeout()` à requisição `http.get` ou `http.post` e usando um bloco `on TimeoutException`.
    - C) Chamando `Navigator.pop()` para fechar a tela atual em caso de timeout.
    - D) Registrando uma tarefa periódica com `WorkManager()` para verificar a conexão.
10. Qual widget é comumente usado para indicar visualmente ao usuário que dados estão sendo carregados de uma fonte remota, como uma API, antes que a UI completa seja exibida?

    - A) `Text`
    - B) `Image.asset`
    - C) `CircularProgressIndicator`
    - D) `FloatingActionButton`

11. Para comunicação em tempo real usando WebSockets em Flutter, qual pacote fornece as ferramentas necessárias para conectar, ouvir e enviar mensagens?

    - A) `http`
    - B) `sqflite`
    - C) `web_socket_channel`
    - D) `cloud_firestore`

12. Qual a principal diferença fundamental entre um `Future` e um `Stream` no Dart, especialmente no contexto de networking?

    - A) `Future` é usado para operações síncronas, enquanto `Stream` é para operações assíncronas.
    - B) `Future` representa uma única resposta assíncrona, enquanto `Stream` pode entregar muitos eventos (mensagens) ao longo do tempo.
    - C) `Future` só pode ser usado para requisições GET, e `Stream` apenas para requisições POST.
    - D) `Stream` é a base para o gerenciamento de estado, enquanto `Future` é para widgets.

13. O que são "Background Tasks" em aplicações Flutter, e qual pacote é mencionado para sua implementação em plataformas Android e iOS?

    - A) Tarefas executadas apenas enquanto o aplicativo está em primeiro plano; `connectivity_plus`.
    - B) Atividades executadas em segundo plano, mesmo com o aplicativo fechado; `WorkManager`.
    - C) Renderização de elementos complexos da UI em threads separadas; `FutureBuilder`.
    - D) Sincronização de dados em tempo real quando a conexão de internet é instável; `http`.

14. Para habilitar tarefas em segundo plano no iOS usando o pacote `WorkManager`, qual arquivo deve ser modificado e qual chave deve ser adicionada à seção `<array>` de `UIBackgroundModes`?

    - A) `AppDelegate.m`; `background_fetch_enabled`
    - B) `pubspec.yaml`; `ios_background_tasks: true`
    - C) `info.plist`; `fetch` e `processing`
    - D) `main.dart`; `Workmanager.enableBackground`

15. De acordo com as fontes, qual é a frequência mínima para a execução de uma tarefa periódica no Android ao usar o `WorkManager`?

    - A) 5 segundos
    - B) 1 minuto
    - C) 15 minutos
    - D) 1 hora

16. Qual é uma recomendação crucial para a segurança de rede em todas as aplicações mobile, incluindo Flutter, para proteger contra ataques man-in-the-middle (MITM)?

    - A) Permitir tráfego HTTP para maior compatibilidade com servidores legados.
    - B) Implementar Transport Layer Security (TLS) pinning (fixação de certificado).
    - C) Usar apenas bibliotecas de terceiros sem revisar suas permissões.
    - D) Desabilitar a validação de certificados SSL para evitar erros de conexão.
17. Por que a arquitetura do Flutter torna o bypass de TLS pinning mais desafiador para atacantes, em comparação com aplicações nativas que dependem do stack TLS do sistema operacional?

    - A) Porque o Flutter não suporta requisições HTTP, apenas WebSockets. [Não é verdade]
    - B) Porque a biblioteca TLS e os componentes de rede do Flutter são integrados diretamente ao Flutter Engine.
    - C) Porque o Flutter utiliza exclusivamente a interface de rede nativa do dispositivo.
    - D) Porque o Flutter Engine não tem vulnerabilidades de segurança conhecidas.
18. Qual é uma desvantagem de segurança notável da ofuscação de código em Dart, conforme as fontes, que pode levar à exposição de informações sensíveis?

    - A) O ofuscador integrado do Dart não ofusca strings, o que pode expor chaves de API ou outros segredos estáticos in-app.
    - B) A ofuscação impede que o aplicativo se conecte corretamente à internet.
    - C) A ofuscação aumenta significativamente o tamanho final do aplicativo.
    - D) A ofuscação é um recurso experimental e não confiável para segurança.
19. Qual é uma desvantagem mencionada do Flutter para desenvolvedores, relacionada à linguagem de programação Dart?

    - A) Dart é uma linguagem de alto desempenho, mas extremamente difícil de aprender.
    - B) Dart é uma linguagem interpretada, o que resulta em um desempenho de UI lento.
    - C) Dart não suporta os princípios da Programação Orientada a Objetos.
    - D) A necessidade de aprender uma nova linguagem (Dart), embora seja considerada fácil de aprender.
20. Qual engine de renderização o Flutter utiliza para garantir que a renderização da UI funcione da mesma maneira em cada plataforma, unificando a experiência?

    - A) React Native Engine
    - B) Hermes Engine
    - C) Skia rendering engine
    - D) WebKit

Gabarito:

1. C
2. D
3. C
4. B
5. B
6. A
7. B
8. C
9. B
10. C
11. C
12. B
13. B
14. C
15. C
16. B
17. B
18. A
19. D
20. C