

Avaliação-07

Ana Beatriz Carvalho de Menezes

Flutter Networking with http Package

Fontes

+ Adicionar

🔍 Descobrir

Selecionar todas as fontes

HTTP Requests and REST API | Networki...

✓

Network Request - Flutter Explained - Le...

✓

Networking Requests in Flutter | Day 10 - ...

✓

Networking | Flutter

✓

Flutter Networking with http Package

4 fontes

Flutter Networking with http Package

4 fontes

O texto é um excerto da documentação oficial do Flutter, uma estrutura de UI de código aberto. Ele se concentra principalmente em tarefas de rede, como fazer solicitações HTTP usando o pacote http, e destaca configurações específicas da plataforma necessárias para Android e macOS. Além disso, a documentação abrange uma vasta gama de outros tópicos essenciais para o desenvolvimento de aplicativos com Flutter, incluindo gerenciamento de estado, manipulação de entrada do usuário, testes, depuração e desempenho. A estrutura geral do site indica um recurso abrangente para desenvolvedores, oferecendo tutoriais, guias de arquitetura, referências de API e suporte para integração com serviços como Firebase e Google Maps.

🔖 Salvar nota

📄

📄 Adicionar nota

🔊 Resumo em Áudio

🗺️ Mapa mental

Gostaria de um questionário de 20 questões de múltipla escolha - itens a, b, c, d -, sobre o tema Networking em Flutter, e por fim, responda-o e justifique.

4 fontes

Que funcionalidades e plataformas o pacote 'http' do Flutter suporta para networking?

🔍 >

O NotebookLM pode gerar respostas incorretas. Por isso, cheque o conteúdo.

Resposta do NotebookLM:

Com certeza! Abaixo, você encontrará um questionário de 20 perguntas de múltipla escolha sobre "Networking em Flutter", seguido pelas respostas e suas justificativas, baseadas nas informações fornecidas nos materiais de origem.

Questionário: Networking em Flutter

1. Qual é o principal objetivo do "networking" em um aplicativo móvel, segundo as fontes?

- a) Apenas para enviar dados para servidores.
- b) Para permitir que o aplicativo exiba apenas conteúdo estático.
- c) Para buscar e enviar dados de/para servidores, APIs ou serviços externos, e gerenciar a comunicação em tempo real para fornecer informações atualizadas e interatividade.
- d) Para substituir completamente o armazenamento de dados local.

2. Qual pacote é recomendado pelas fontes para lidar com requisições HTTP em Flutter?

- a) `dio`.
- b) `http`.
- c) `chopper`.
- d) `retrofit`.

3. Quais são os quatro tipos principais de requisições HTTP mencionados nas fontes?

- a) Connect, Trace, Options, Head.
- b) Get, Send, Update, Remove.
- c) Get, Post, Put, Delete.
- d) Fetch, Submit, Modify, Erase.

4. Para que serve uma requisição `GET` em HTTP?

- a) Para enviar dados para um servidor.
- b) Para atualizar dados em um servidor.
- c) Para excluir dados de um servidor.
- d) Para buscar dados de um servidor.

5. Como é feito o processo de conversão de uma URL string para um objeto `Uri` ao realizar uma requisição HTTP com o pacote `http`?

- a) Usando `Uri.fromText()`.
- b) Usando `Uri.parse()`.

- c) Usando `Uri.fromString()`.
- d) Usando `Url.toUri()`.

6. Por que é importante usar `async` e `await` ao fazer chamadas de API em Flutter?

- a) Para garantir que o aplicativo não congele enquanto espera pela resposta da API.
- b) Para tornar a função síncrona.
- c) Para habilitar a depuração de rede.
- d) Para permitir que o aplicativo execute várias operações de UI simultaneamente.

7. Qual código de status HTTP indica uma resposta bem-sucedida de um servidor?

- a) 404.
- b) 500.
- c) 200.
- d) 401.

8. Como os dados JSON recebidos de uma API são convertidos em um objeto Dart para ser utilizado no aplicativo?

- a) Usando `json.encode()`.
- b) Usando `json.toDart()`.
- c) Usando `json.decode()`.
- d) Usando `response.toString()`.

9. Ao exibir uma lista de dados dinâmicos em Flutter, qual widget é especificamente mencionado como adequado para esta tarefa?

- a) `Column`.
- b) `Container`.
- c) `ListView.builder`.
- d) `GridView`.

10. Qual é a funcionalidade de uma requisição `POST`?

- a) Buscar dados de um servidor.
- b) Enviar dados para um servidor.
- c) Atualizar dados em um servidor.
- d) Deletar dados de um servidor.

11. Ao enviar dados via `POST` em formato JSON, qual cabeçalho (`header`) é especificado para indicar o tipo de conteúdo?

- a) `Accept: application/json`.
- b) `Content-Type: text/plain`.
- c) `Content-Type: application/json`.
- d) `Authorization: Bearer <token>`.

12. O que significa "REST" em REST API, segundo as fontes?

- a) Real-time Event Streaming Technology.
- b) Representational State Transfer.
- c) Resource Exchange Standardized Transfer.
- d) Remote Execution Service Technology.

13. Qual conceito de REST API descreve o fato de que cada requisição do cliente ao servidor deve conter todas as informações necessárias para entender e processar a requisição?

- a) Recursos.
- b) Métodos HTTP.
- c) Stateless (Sem estado).
- d) CRUD.

14. Qual é uma das melhores práticas para lidar com erros em requisições de rede em Flutter, de acordo com as fontes?

- a) Ignorar erros e tentar a requisição novamente.
- b) Exibir mensagens de erro amigáveis para o usuário e configurar um tempo limite (`timeout`) para as requisições.
- c) Apenas registrar os erros no console.
- d) Bloquear o aplicativo até que a requisição seja bem-sucedida.

15. Qual é uma forma de mostrar um indicador de carregamento (`loading sign`) em um aplicativo Flutter enquanto os dados da API estão sendo buscados?

- a) Usar um `SizedBox` vazio.
- b) Usar um `CircularProgressIndicator`.
- c) Usar um `ElevatedButton`.
- d) Usar um `Text` widget.

16. Em Flutter, para exibir dados em um formato de grade com um número fixo de colunas, qual widget pode ser usado?

- a) `ListView`.
- b) `Column`.
- c) `GridView.builder` com `SliverGridDelegateWithFixedCrossAxisCount`.
- d) `SingleChildScrollView`.

17. Onde o pacote `http` deve ser adicionado em um projeto Flutter para ser utilizado?

- a) No arquivo `main.dart`.
- b) Dentro da seção `dependencies` do arquivo `pubspec.yaml`.
- c) Diretamente no código como um `import`.
- d) No arquivo `AndroidManifest.xml`.

18. Para que serve o método `fromJson` em um modelo Dart ao trabalhar com JSON de APIs?

- a) Para converter um objeto Dart em JSON.
- b) Para criar um novo objeto Dart a partir de um objeto JSON (mapa de chave-valor).
- c) Para serializar dados para envio.
- d) Para validar o esquema JSON.

19. Ao lidar com grandes conjuntos de dados retornados por APIs, quais são as melhores práticas sugeridas pelas fontes?

- a) Desativar a manipulação de erros.
- b) Usar mecanismos de cache e considerar paginação.
- c) Buscar todos os dados de uma vez, independentemente do tamanho.
- d) Exibir apenas os primeiros 10 itens.

20. No contexto de networking em Flutter, o que um `Future` representa?

- a) Um erro que ocorrerá no futuro.
- b) Um valor que será recebido em algum momento posterior, ou seja, representa uma computação assíncrona que pode produzir um resultado ou um erro.
- c) Uma variável que nunca mudará.
- d) Uma função que retorna `void`.

Respostas e Justificativas:

1. **c) Para buscar e enviar dados de/para servidores, APIs ou serviços externos, e gerenciar a comunicação em tempo real para fornecer informações atualizadas e interatividade.**

- **Justificativa:** As fontes afirmam que "networking é um dos componentes centrais de quase todo aplicativo moderno. O aplicativo precisa buscar dados de servidores, APIs ou serviços externos para fornecer aos usuários as informações mais atualizadas" e que "comunicar com APIs é crucial para tornar os aplicativos interativos e ricos em dados".

2. **b) `http`.**

- **Justificativa:** Várias fontes indicam que "uma das formas mais comuns e fáceis de usar é o pacote HTTP", e mencionam explicitamente a instalação e uso do pacote `http` de pub.dev para fazer requisições HTTP.

3. **c) `Get`, `Post`, `Put`, `Delete`.**

- **Justificativa:** As fontes listam claramente os quatro tipos de requisições HTTP em Flutter como "Get request to fetch data from a server, then we have Post request to send the data to a server, then we have Put request to update

data on the server and finally we have Delete request to delete data from the server".

4. **d) Para buscar dados de um servidor.**

- **Justificativa:** O material afirma que o `get request` é usado "to fetch data from a server".

5. **b) Usando `Uri.parse()`.**

- **Justificativa:** As fontes demonstram o uso de `URI.parse()` para converter uma URL string no formato `Uri` necessário para as requisições HTTP, tanto em requisições GET quanto POST.

6. **a) Para garantir que o aplicativo não congele enquanto espera pela resposta da API.**

- **Justificativa:** É explicado que as requisições de rede são tarefas assíncronas e que `await` é usado para "esperar até que a resposta não chegue", e que a função precisa ser marcada com `async` para que se possa `await` a resposta, evitando o congelamento do aplicativo.

7. **c) 200.**

- **Justificativa:** As fontes indicam que "if response status code is 200 then we have successfully got the response" e "if status code is equal to 200 then print that post created successfully", e também que "status code 200 is our favorite one because it is okay everything works well".

8. **c) Usando `json.decode()`.**

- **Justificativa:** É explicitado que `json.decode` é usado para "pass the Json response into a Dart object", e que "this JSON is nothing for Dart it doesn't understand that it's just a string, so there is a package called JSON decode from the dart convert package".

9. **c) `ListView.builder`.**

- **Justificativa:** As fontes mostram `ListView.builder` sendo usado para "display data dynamically in the apps UI" e afirmam que Flutter fornece `ListView` (e `GridView`) para "display data in a structured way".

10. **b) Enviar dados para um servidor.**

- **Justificativa:** As fontes definem `post request` como o método "to send the data to a server", e demonstram seu uso para "sending data to the server using a post request".

11. **c) `Content-Type: application/json`.**

- **Justificativa:** Ao enviar dados via POST, as fontes instruem a adicionar `headers` e especificar "the content type as application J", e explicitam que "the header specify that the data is being sent as Json".

12. b) Representational State Transfer.

- **Justificativa:** As fontes afirmam que "REST stands for Representational State Transfer".

13. c) Stateless (Sem estado).

- **Justificativa:** O conceito de stateless é descrito como "each request from the client to the server must contain all the information needed to understand and process the request".

14. b) Exibir mensagens de erro amigáveis para o usuário e configurar um tempo limite (timeout) para as requisições.

- **Justificativa:** As fontes listam soluções para cenários de erro, como "gracefully handle server Errors By showing userfriendly messages" e "set a timeout duration for our API request to avoid waiting indefinitely".

15. b) Usar um CircularProgressIndicator.

- **Justificativa:** É demonstrado como adicionar um `CircularProgressIndicator` para mostrar um sinal de carregamento "until the data is not loaded".

16. c) GridView.builder com SliverGridDelegateWithFixedCrossAxisCount.

- **Justificativa:** As fontes mostram o uso de `GridView.builder` e a configuração de seu `gridDelegate` para `SliverGridDelegateWithFixedCrossAxisCount` para "set the column count" e exibir dados em formato de grade.

17. b) Dentro da seção dependencies do arquivo pubspec.yaml.

- **Justificativa:** As fontes instruem "open the pubspec. AML file and here inside this dependency section add the HTTP dependency", e mostram o processo de obtenção do pacote `http` de `pub.dev` e sua adição às `dependencies` no `pubspec.yaml`.

18. b) Para criar um novo objeto Dart a partir de um objeto JSON (mapa de chave-valor).

- **Justificativa:** O construtor `fromJson` é explicado como uma forma de "transform this string that is a map between a key a string and something else into this person", e é usado para criar objetos `Person` a partir de dados JSON.

19. b) Usar mecanismos de cache e considerar paginação.

- **Justificativa:** As fontes recomendam "use caching mechanism when dealing with large data and consider pagination when working with apis that return large data set".

20. b) Um valor que será recebido em algum momento posterior, ou seja, representa uma computação assíncrona que pode produzir um resultado ou um erro.

- **Justificativa:** É explicado que a requisição HTTP `get` "is a future", e que as funções assíncronas "return a future of something". As fontes também mencionam o uso de `FutureBuilder` para "receive a network request and show something", o que implica que um `Future` representa um resultado a ser obtido posteriormente.