API G

Nesta aula, exploraremos as diversas APIs oferecidas pelo Google. As APIs do Google fornecem uma ampla gama de serviços, desde mapas e autenticação até armazenamento em nuvem e processamento de dados. Vamos discutir o que são APIs, como elas podem ser usadas para melhorar seus aplicativos Flutter e apresentar uma visão geral das APIs mais populares do Google, como Google Maps, Google Drive, Google Calendar e Google Sheets.

Visão Geral das APIs do Google

O que são APIs?

API (Application Programming Interface) é um conjunto de definições e protocolos usados para desenvolver e integrar o software de aplicativos. As APIs permitem que diferentes sistemas se comuniquem entre si, facilitando a troca de dados e a funcionalidade.

Principais APIs do Google

1. Google Maps API: Utilizada para incorporar mapas interativos, rotas e locais em aplicativos.

2. Google Drive API: Permite o gerenciamento de arquivos e armazenamento na nuvem.

3. Google Calendar API: Utilizada para acessar e gerenciar eventos de calendário.

4. Google Sheets API: Permite a leitura e escrita de dados em planilhas Google.

Benefícios das APIs do Google

Escalabilidade: As APIs do Google são projetadas para lidar com grandes volumes de dados e tráfego.

Segurança: Integração com autenticação segura, como OAuth 2.0.

Funcionalidade Ampla: Acesso a uma vasta gama de serviços e ferramentas.

Nesta aula, aprenderemos como configurar e autenticar o uso das APIs do Google no seu aplicativo Flutter. Discutiremos como criar um projeto no Google Cloud Console, obter credenciais de API, e configurar a autenticação usando OAuth 2.0. Ao final desta aula, você estará pronto para integrar qualquer API do Google ao seu aplicativo Flutter.

Configurando e Autenticando APIs do Google

Passos para Configurar e Autenticar APIs

1. Criar um Projeto no Google Cloud Console

- Acesse o Google Cloud Console.

- Crie um novo projeto ou use um existente.

2. Habilitar a API Desejada

- No painel do Google Cloud Console, navegue até a biblioteca de APIs.

- Pesquise e habilite a API que deseja usar (por exemplo, Google Drive API).

3. Obter Credenciais de API

- Vá para “Credenciais” no menu de navegação.

- Clique em “Criar credenciais” e selecione “ID do Cliente OAuth 2.0”.

- Configure a tela de consentimento OAuth.

- Baixe o arquivo JSON com as credenciais.

4. Adicionar Dependências ao pubspec.yaml

```
dependencies:
  flutter:
    sdk: flutter
  googleapis: latest_version
  googleapis_auth: latest_version
  http: ^0.13.3
```

5. Configurar Autenticação no Flutter

```
import ‘package:flutter/material.dart’;
import ‘package:googleapis/drive/v3.dart’ as drive;
import ‘package:googleapis_auth/auth_io.dart’;
import ‘package:http/http.dart’ as http;
void main() {
  runApp(MyApp());
}
```

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomeScreen(),
    );
  }
}
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}
class _HomeScreenState extends State<HomeScreen> {
  final _clientId = ClientId('YOUR_CLIENT_ID', 'YOUR_CLIENT_SECRET');
  final _scopes = [drive.DriveApi.driveFileScope];
  Future<void> _authenticate() async {
    await clientViaUserConsent(_clientId, _scopes, (url) {
      // Abra o URL em um navegador para o usuário fazer login
      print('Por favor, faça login em: $url');
```

```
  }).then((authClient) {
    // Use o authClient para acessar a API do Google
    final driveApi = drive.DriveApi(authClient);
    _listFiles(driveApi);
  }).catchError((e) {
    print('Erro na autenticação: $e');
  });
}

Future<void> _listFiles(drive.DriveApi driveApi) async {
  final fileList = await driveApi.files.list();
  fileList.files?.forEach((file) {
    print('Encontrado arquivo: ${file.name}');
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Autenticação Google API'),
```

```
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: _authenticate,
          child: Text('Autenticar com Google'),
        ),
      ),
    );
  }
}
```

Explicação: Este exemplo mostra como configurar a autenticação OAuth 2.0 no Flutter, permitindo que o usuário faça login e conceda permissões ao aplicativo.

Nesta aula, vamos aprender a usar a API Google Drive no Flutter. Veremos como listar arquivos, fazer upload e download de arquivos no Google Drive. Esta integração é útil para aplicativos que precisam gerenciar arquivos na nuvem.

Usando a API Google Drive no Flutter

Listar Arquivos no Google Drive

1. Adicionar Dependências ao pubspec.yaml

dependencies:

```
  flutter:
    sdk: flutter
  googleapis: latest_version
  googleapis_auth: latest_version
  http: ^0.13.3
```

2. Código para Listar Arquivos

```
import 'package:flutter/material.dart';
import 'package:googleapis/drive/v3.dart' as drive;
import 'package:googleapis_auth/auth_io.dart';
import 'package:http/http.dart' as http;
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomeScreen(),
    );
```

```
  }
}
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}
class _HomeScreenState extends State<HomeScreen> {
  final _clientId = ClientId('YOUR_CLIENT_ID', 'YOUR_CLIENT_SECRET');
  final _scopes = [drive.DriveApi.driveFileScope];
  List<String> _files = [];
  Future<void> _authenticate() async {
    await clientViaUserConsent(_clientId, _scopes, (url) {
      // Abra o URL em um navegador para o usuário fazer login
      print('Por favor, faça login em: $url');
    }).then((authClient) {
      // Use o authClient para acessar a API do Google
      final driveApi = drive.DriveApi(authClient);
      _listFiles(driveApi);
    }).catchError((e) {
```

```
    print('Erro na autenticação: $e');
  });
}

Future<void> _listFiles(drive.DriveApi driveApi) async {
  final fileList = await driveApi.files.list();
  setState(() {
    _files = fileList.files?.map((file) => file.name).toList() ?? [];
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Google Drive API'),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
```

```
        ElevatedButton(
          onPressed: _authenticate,
          child: Text('Autenticar com Google'),
        ),
        Expanded(
          child: ListView.builder(
            itemCount: _files.length,
            itemBuilder: (context, index) {
              return ListTile(
                title: Text(_files[index]),
              );
            },
          ),
        ),
      ],
    ),
  ),
);
}
```

```
}
```

Explicação: Este código autentica o usuário e lista os arquivos armazenados no Google Drive, exibindo-os em uma lista.

Upload de Arquivos para o Google Drive

1. Código para Upload de Arquivos

```
Future<void> _uploadFile(drive.DriveApi driveApi) async {
  final file = drive.File();
  file.name = 'upload.txt';
  final fileContent = 'Conteúdo do arquivo de upload.';
  final media = drive.Media(Stream.value(fileContent.codeUnits).asBroadcastStream(), fileContent.length);
  final result = await driveApi.files.create(file, uploadMedia: media);
  print('Arquivo enviado: ${result.name}');
}
```

2. Adicionar Função de Upload ao Botão

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
```

```
  title: Text('Google Drive API'),
),
body: Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      ElevatedButton(
        onPressed: _authenticate,
        child: Text('Autenticar com Google'),
      ),
      ElevatedButton(
        onPressed: () => _uploadFile(driveApi),
        child: Text('Upload de Arquivo'),
      ),
      Expanded(
        child: ListView.builder(
          itemCount: _files.length,
          itemBuilder: (context, index) {
            return ListTile(
```

```
                title: Text(_files[index]),
              );
            },
          ),
        ),
      ],
    ),
  ),
 );
}
```

Explicação: Este código adiciona a funcionalidade de upload de arquivos ao Google Drive, permitindo que o usuário envie arquivos diretamente do aplicativo Flutter.

Nesta aula, aprenderemos a integrar outras APIs do Google ao seu aplicativo Flutter. Veremos exemplos de como usar a API do Google Calendar para gerenciar eventos e a API do Google Sheets para ler e escrever dados em planilhas. A integração de várias APIs do Google pode aumentar significativamente a funcionalidade do seu aplicativo.

Integrando Outras APIs do Google

Usando a API do Google Calendar

1. Adicionar Dependências ao pubspec.yaml

```
dependencies:
  flutter:
    sdk: flutter
  googleapis: latest_version
  googleapis_auth: latest_version
  http: ^0.13.3
```

2. Código para Listar Eventos do Google Calendar

```
import 'package:flutter/material.dart';
import 'package:googleapis/calendar/v3.dart' as calendar;
import 'package:googleapis_auth/auth_io.dart';
import 'package:http/http.dart' as http;
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomeScreen(),
```

```
    );
  }
}
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}
class _HomeScreenState extends State<HomeScreen> {
  final _clientId = ClientId('YOUR_CLIENT_ID', 'YOUR_CLIENT_SECRET');
  final _scopes = [calendar.CalendarApi.calendarReadonlyScope];
  List<String> _events = [];
  Future<void> _authenticate() async {
    await clientViaUserConsent(_clientId, _scopes, (url) {
      // Abra o URL em um navegador para o usuário fazer login
      print('Por favor, faça login em: $url');
    }).then((authClient) {
      // Use o authClient para acessar a API do Google
      final calendarApi = calendar.CalendarApi(authClient);
      _listEvents(calendarApi);
```

```
    }).catchError((e) {
      print('Erro na autenticação: $e');
    });
  }
  Future<void> _listEvents(calendar.CalendarApi calendarApi) async {
    final eventList = await calendarApi.events.list('primary');
    setState(() {
      _events = eventList.items?.map((event) => event.summary ?? 'Sem título').toList() ?? [];
    });
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Google Calendar API'),
      ),
      body: Center(
        child: Column(
```

```
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
       ElevatedButton(
        onPressed: _authenticate,
        child: Text('Autenticar com Google'),
       ),
       Expanded(
        child: ListView.builder(
         itemCount: _events.length,
         itemBuilder: (context, index) {
          return ListTile(
           title: Text(_events[index]),
          );
         },
        ),
       ),
      ],
     ),
    ),
```

```
    );
  }
}
```

Explicação: Este código autentica o usuário e lista os eventos do Google Calendar, exibindo-os em uma lista.

Usando a API do Google Sheets

1. Adicionar Dependências ao pubspec.yaml

dependencies:

```
 flutter:
   sdk: flutter
 googleapis: latest_version
 googleapis_auth: latest_version
 http: ^0.13.3
```

2. Código para Ler Dados do Google Sheets

```
import 'package:flutter/material.dart';
import 'package:googleapis/sheets/v4.dart' as sheets;
import 'package:googleapis_auth/auth_io.dart';
import 'package:http/http.dart' as http;
void main() {
```

```
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomeScreen(),
    );
  }
}
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}
class _HomeScreenState extends State<HomeScreen> {
  final _clientId = ClientId('YOUR_CLIENT_ID', 'YOUR_CLIENT_SECRET');
  final _scopes = [sheets.SheetsApi.spreadsheetsReadonlyScope];
  List<List<Object>> _values = [];
  Future<void> _authenticate() async {
```

```
    await clientViaUserConsent(_clientId, _scopes, (url) {
      // Abra o URL em um navegador para o usuário fazer login
      print('Por favor, faça login em: $url');
    }).then((authClient) {
      // Use o authClient para acessar a API do Google
      final sheetsApi = sheets.SheetsApi(authClient);
      _readSheet(sheetsApi);
    }).catchError((e) {
      print('Erro na autenticação: $e');
    });
  }

  Future<void> _readSheet(sheets.SheetsApi sheetsApi) async {
    final response = await sheetsApi.spreadsheets.values.get('YOUR_SHEET_ID', 'Sheet1!A1:E10');
    setState(() {
      _values = response.values ?? [];
    });
  }

  @override
```

```
Widget build(BuildContext context) {
 return Scaffold(
  appBar: AppBar(
   title: Text('Google Sheets API'),
  ),
  body: Center(
   child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
     ElevatedButton(
      onPressed: _authenticate,
      child: Text('Autenticar com Google'),
     ),
     Expanded(
      child: ListView.builder(
       itemCount: _values.length,
       itemBuilder: (context, index) {
        return ListTile(
         title: Text(_values[index].join(', ')),
```

```
                );
              },
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

Explicação: Este código autentica o usuário e lê dados de uma planilha do Google Sheets, exibindo-os em uma lista.

Esses exemplos e explicações fornecem uma base sólida para iniciantes em Flutter aprenderem a integrar e usar várias APIs do Google em seus aplicativos. Para mais detalhes, consulte a documentação oficial do Flutter: Flutter Documentation.

Materiais Extras

Você pode realizar o download do arquivo contendo os materiais extras utilizados ao longo das aulas por meio do seguinte link: https://drive.google.com/file/d/1mg7lqMl8Pt2zl0rHIsFS0Qew00YN-sEX/view?usp=sharing.

Conteúdo Bônus

Para aprofundar seus conhecimentos sobre APIs do Google e aprender a utilizar a API do Google Tag Manager, recomendamos o seguinte material:

Nome da obra: “Guia do Desenvolvedor da API do Google Tag Manager v2”

Autor: Google Developers

Plataforma: Site oficial do Google Developers

Este conteúdo gratuito oferece instruções detalhadas e exemplos práticos de como integrar e manipular a API do Google Tag Manager em seus projetos, permitindo um gerenciamento mais eficiente de tags e melhorias na implementação de ferramentas de análise e marketing.

Referências Bibliográficas

BOYLESTAD, R. L.; NASHELSKY, L. Dispositivos Eletrônicos e Teoria de Circuitos. 11. ed. Pearson, 2013.

DEITEL, P. J.; DEITEL, H. M. Ajax, Rich Internet Applications e Desenvolvimento Web para Programadores. Pearson, 2008.

DUARTE, W. Delphi para Android e iOS: Desenvolvendo Aplicativos Móveis. Brasport, 2015.

FELIX, R.; SILVA, E. L. da. Arquitetura para Computação Móvel. 2. ed. Pearson, 2019.

LEE, V.; SCHNEIDER, H.; SCHELL, R. Aplicações Móveis: Arquitetura, Projeto e Desenvolvimento. Pearson, 2005.

MARINHO, A. L.; CRUZ, J. L. da. Desenvolvimento de Aplicações para Internet. 2. ed. Pearson, 2019.

MOLETTA, A. Você na Tela: Criação Audiovisual para a Internet. Summus, 2019.

SILVA, D. (Org.) Desenvolvimento para dispositivos móveis. Pearson, 2017.

Ir para exercício