🚀 Exemplos de Uso - ARBORIS AI Gemini Service

📸 Exemplo 1: Análise de Imagem de Planta

```dart
import 'dart:io';
import 'package:arboris_genesis/services/gemini_service.dart';
import 'package:arboris_genesis/services/auth_service.dart';

Future<void> analisarPlanta() async {
  // 1. Obter imagem (de câmera ou galeria)
  final imagePath = '/path/to/plant_image.jpg';
  final imageFile = File(imagePath);

  // 2. Obter userId do usuário autenticado
  final userId = AuthService.instance.currentUser?.uid ?? 'guest';

  // 3. Analisar imagem com Gemini
  try {
    final analysis = await GeminiService.instance.analyzeImage(
      imageFile,
      userId: userId,
    );

    // 4. Usar os resultados
    print('🌿 Espécie: ${analysis.species}');
    print('📊 Confiança: ${(analysis.confidence * 100).toStringAsFixed(1)}%');
    print('💚 Saúde: ${analysis.healthStatus}');
    print('📖 Descrição: ${analysis.description}');

    // Características
    print('\n🔍 Características:');
    for (final char in analysis.characteristics) {
      print('  • $char');
    }

    // Informações adicionais
    if (analysis.additionalInfo['benefits'] != null) {
      print('\n✨ Benefícios:');
      for (final benefit in analysis.additionalInfo['benefits']) {
        print('  • $benefit');
      }
    }

  } catch (e) {
    print('❌ Erro ao analisar: $e');
  }
}
```

💬 Exemplo 2: Chat Básico

```dart
import 'package:arboris_genesis/services/gemini_service.dart';
import 'package:arboris_genesis/services/auth_service.dart';

Future<void> chatBasico() async {
  final userId = AuthService.instance.currentUser?.uid ?? 'guest';

  try {
    // Fazer uma pergunta
    final response = await GeminiService.instance.generateResponse(
      'Quais são os benefícios das árvores nativas?',
      userId: userId,
    );

    print('🤖 Resposta da IA:');
    print(response);

  } catch (e) {
    print('❌ Erro no chat: $e');
  }
}
```

🌊 Exemplo 3: Chat com Streaming (Tempo Real)

```dart
import 'package:arboris_genesis/services/gemini_service.dart';
import 'package:arboris_genesis/services/auth_service.dart';

Future<void> chatComStreaming() async {
  final userId = AuthService.instance.currentUser?.uid ?? 'guest';

  print('🤖 Resposta da IA (streaming):');

  try {
    // Receber resposta em tempo real
    await for (final chunk in GeminiService.instance.streamResponse(
      'Conte-me sobre a importância da Mata Atlântica',
      userId: userId,
    )) {
      // Cada 'chunk' é um pedaço da resposta
      print(chunk); // Sem quebra de linha para efeito de digitação
    }

    print('\n✅ Resposta completa!');

  } catch (e) {
    print('❌ Erro no streaming: $e');
  }
}
```

🔄 Exemplo 4: Chat com Contexto (Sessão)

```dart
import 'package:arboris_genesis/services/gemini_service.dart';
import 'package:arboris_genesis/services/auth_service.dart';

Future<void> chatComContexto() async {
  final userId = AuthService.instance.currentUser?.uid ?? 'guest';

  // Iniciar sessão de chat (mantém contexto)
  GeminiService.instance.startChatSession();

  try {
    // Primeira mensagem
    print('👤 Usuário: Olá! Pode me falar sobre ipês?');
    var response = await GeminiService.instance.sendChatMessage(
      'Olá! Pode me falar sobre ipês?',
      userId: userId,
    );
    print('🤖 IA: $response\n');

    // Segunda mensagem (com contexto da primeira)
    print('👤 Usuário: Qual a diferença entre ipê amarelo e roxo?');
    response = await GeminiService.instance.sendChatMessage(
      'Qual a diferença entre ipê amarelo e roxo?',
      userId: userId,
    );
    print('🤖 IA: $response\n');

    // Terceira mensagem (ainda com contexto)
    print('👤 Usuário: Como posso cultivar um em casa?');
    response = await GeminiService.instance.sendChatMessage(
      'Como posso cultivar um em casa?',
      userId: userId,
    );
    print('🤖 IA: $response\n');

    // Limpar sessão quando terminar
    GeminiService.instance.clearChatSession();

  } catch (e) {
    print('❌ Erro no chat: $e');
  }
}
```

🔄 Exemplo 5: Chat com Streaming em Sessão

```dart
import 'package:arboris_genesis/services/gemini_service.dart';
import 'package:arboris_genesis/services/auth_service.dart';

Future<void> chatStreamComContexto() async {
  final userId = AuthService.instance.currentUser?.uid ?? 'guest';

  // Iniciar sessão
  GeminiService.instance.startChatSession();

  // Lista de perguntas
  final perguntas = [
    'Quais plantas são boas para purificar o ar?',
    'Qual dessas plantas é mais fácil de cuidar?',
    'Posso colocá-las no quarto?',
  ];

  for (final pergunta in perguntas) {
    print('\n👤 Usuário: $pergunta');
    print('🤖 IA: ');

    try {
      // Streaming com contexto
      await for (final chunk in GeminiService.instance.sendChatMessageStream(
        pergunta,
        userId: userId,
      )) {
        print(chunk);
      }

      print('\n');

    } catch (e) {
      print('❌ Erro: $e');
    }
  }

  GeminiService.instance.clearChatSession();
}
```

📚 Exemplo 6: Buscar Histórico de Análises

```dart
import 'package:arboris_genesis/services/gemini_service.dart';
import 'package:arboris_genesis/services/auth_service.dart';

Future<void> buscarHistoricoAnalises() async {
  final userId = AuthService.instance.currentUser?.uid ?? 'guest';

  try {
    final analyses = await GeminiService.instance.getPlantAnalysisHistory(userId);

    print('📊 Histórico de Análises (${analyses.length} itens):\n');

    for (var i = 0; i < analyses.length; i++) {
      final analysis = analyses[i];
      print('${i + 1}. ${analysis.species}');
      print('   Data: ${analysis.timestamp}');
      print('   Confiança: ${(analysis.confidence * 100).toStringAsFixed(1)}%');
      print('   Saúde: ${analysis.healthStatus}');
      print('');
    }

  } catch (e) {
    print('❌ Erro ao buscar histórico: $e');
  }
}
```

💬 Exemplo 7: Buscar Histórico de Chat

```dart
import 'package:arboris_genesis/services/gemini_service.dart';
import 'package:arboris_genesis/services/auth_service.dart';

Future<void> buscarHistoricoChat() async {
  final userId = AuthService.instance.currentUser?.uid ?? 'guest';

  try {
    final messages = await GeminiService.instance.getChatHistory(userId);

    print('💬 Histórico de Chat (${messages.length} mensagens):\n');

    for (var i = 0; i < messages.length; i++) {
      final msg = messages[i];
      print('${i + 1}. [${msg.timestamp}]');
      print('   👤 Usuário: ${msg.message}');
      print('   🤖 IA: ${msg.response.substring(0, 100)}...');
      print('');
    }

  } catch (e) {
    print('❌ Erro ao buscar histórico: $e');
  }
}
```

🧪 Exemplo 8: Testar Conexão com Gemini

```dart
import 'package:arboris_genesis/services/gemini_service.dart';

Future<void> testarConexao() async {
  print('🔌 Testando conexão com Gemini...');

  final isConnected = await GeminiService.instance.testConnection();

  if (isConnected) {
    print('✅ Conexão estabelecida com sucesso!');
  } else {
    print('❌ Falha ao conectar com Gemini');
    print('   Verifique:');
    print('   • Conexão com internet');
    print('   • API key está correta');
    print('   • Cotas da API não excedidas');
  }
}
```

🎯 Exemplo 9: Widget Flutter Completo - Análise de Planta

```dart
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'dart:io';
import 'package:arboris_genesis/services/gemini_service.dart';
import 'package:arboris_genesis/services/auth_service.dart';

class PlantAnalysisWidget extends StatefulWidget {
  const PlantAnalysisWidget({super.key});

  @override
  State<PlantAnalysisWidget> createState() => _PlantAnalysisWidgetState();
}

class _PlantAnalysisWidgetState extends State<PlantAnalysisWidget> {
  final ImagePicker _picker = ImagePicker();
  bool _isAnalyzing = false;
  String? _result;

  Future<void> _analyzePlant() async {
    // Capturar imagem da galeria
    final XFile? image = await _picker.pickImage(source: ImageSource.gallery);
    if (image == null) return;

    setState(() {
      _isAnalyzing = true;
      _result = null;
    });

    try {
      // Analisar com Gemini
      final userId = AuthService.instance.currentUser?.uid ?? 'guest';
      final analysis = await GeminiService.instance.analyzeImage(
        File(image.path),
        userId: userId,
      );

      setState(() {
        _result = '${analysis.species}\n'
            'Confiança: ${(analysis.confidence * 100).toStringAsFixed(1)}%\n'
            'Saúde: ${analysis.healthStatus}';
      });
    } catch (e) {
      setState(() {
        _result = 'Erro: $e';
      });
    } finally {
      setState(() {
        _isAnalyzing = false;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        ElevatedButton(
          onPressed: _isAnalyzing ? null : _analyzePlant,
          child: Text(_isAnalyzing ? 'Analisando...' : 'Analisar Planta'),
        ),
        if (_result != null)
          Padding(
```

```
            padding: const EdgeInsets.all(16),
            child: Text(_result!),
          ),
        ],
      );
    }
  }
```

🎯 Exemplo 10: Widget Flutter Completo - Chat Simples

```dart
import 'package:flutter/material.dart';
import 'package:arboris_genesis/services/gemini_service.dart';
import 'package:arboris_genesis/services/auth_service.dart';

class SimpleChatWidget extends StatefulWidget {
  const SimpleChatWidget({super.key});

  @override
  State<SimpleChatWidget> createState() => _SimpleChatWidgetState();
}

class _SimpleChatWidgetState extends State<SimpleChatWidget> {
  final TextEditingController _controller = TextEditingController();
  final List<String> _messages = [];
  bool _isLoading = false;

  Future<void> _sendMessage() async {
    final message = _controller.text.trim();
    if (message.isEmpty) return;

    setState(() {
      _messages.add('Você: $message');
      _controller.clear();
      _isLoading = true;
    });

    try {
      final userId = AuthService.instance.currentUser?.uid ?? 'guest';
      final response = await GeminiService.instance.generateResponse(
        message,
        userId: userId,
      );

      setState(() {
        _messages.add('IA: $response');
      });
    } catch (e) {
      setState(() {
        _messages.add('Erro: $e');
      });
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Expanded(
          child: ListView.builder(
            itemCount: _messages.length,
            itemBuilder: (context, index) => ListTile(
              title: Text(_messages[index]),
            ),
          ),
        ),
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Row(
```

```
            children: [
              Expanded(
                child: TextField(
                  controller: _controller,
                  decoration: const InputDecoration(
                    hintText: 'Digite sua mensagem...',
                  ),
                  onSubmitted: (_) => _sendMessage(),
                ),
              ),
              IconButton(
                icon: _isLoading
                    ? const CircularProgressIndicator()
                    : const Icon(Icons.send),
                onPressed: _isLoading ? null : _sendMessage,
              ),
            ],
          ),
        ),
      ],
    );
  }
}
```

🎓 Dicas e Boas Práticas

1. Sempre Inicialize o GeminiService

```
@override
void initState() {
  super.initState();
  GeminiService.instance.initialize();
}
```

2. Use Try-Catch para Tratamento de Erros

```
try {
  final response = await GeminiService.instance.generateResponse(...);
} catch (e) {
  // Mostrar erro ao usuário
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text('Erro: $e')),
  );
}
```

3. Otimize Imagens Antes de Enviar

```
final XFile? image = await _picker.pickImage(
  source: ImageSource.gallery,
  maxWidth: 1920,    // Limita largura
  maxHeight: 1080,   // Limita altura
  imageQuality: 85,  // Comprime (0-100)
);
```

4. Use Streaming para Melhor UX

```dart
// Em vez de esperar resposta completa
final response = await GeminiService.instance.generateResponse(...);

// Use streaming para resposta em tempo real
await for (final chunk in GeminiService.instance.streamResponse(...)) {
  // Atualiza UI a cada chunk
  setState(() => responseText += chunk);
}
```

5. Limpe Sessões de Chat Quando Necessário

```dart
// Ao sair da tela de chat
@override
void dispose() {
  GeminiService.instance.clearChatSession();
  super.dispose();
}
```

🚨 Troubleshooting Comum

Problema: “API key inválida”

Solução: Verifique se a API key em `gemini_service.dart` está correta.

Problema: “Timeout na requisição”

Solução: Imagens muito grandes podem demorar. Use `maxWidth` e `maxHeight`.

Problema: “Firestore permission denied”

Solução: Configure regras de segurança do Firestore para permitir escrita.

Problema: “Câmera não disponível no emulador”

Solução: Use `ImageSource.gallery` em vez de `ImageSource.camera`.

📖 Referências

- GeminiService Docs (./README_GEMINI_IMPLEMENTATION.md)
- Google Generative AI (https://pub.dev/packages/google_generative_ai)
- Gemini API Guide (https://ai.google.dev/docs)