



```
import 'package:flutter/material.dart';
import 'package:video_player/video_player.dart';

void main() => runApp(const VideoApp());

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

  @override
```

```

    State<VideoScreen> createState() => _VideoScreenState();
}

class _VideoScreenState extends State<VideoScreen> {
    late final VideoPlayerController _controller;
    late Future<void> _initFuture;

    bool _muted = false;
    bool _looping = true;
    bool _fullscreen = false;

    List<Uri> videos = [
        Uri.parse("https://video-
previews.elements.envatousercontent.com/9376b5da-eeec-4497-81ec-
24974097b70c/watermarked_preview/watermarked_preview.mp4"),
        Uri.parse(" "),
    ];

    @override
    void initState() {
        // TODO: implement initState
        super.initState();

        _controller = VideoPlayerController.networkUrl(
            videos[0],
            videoPlayerOptions: VideoPlayerOptions(mixWithOthers: true),
        );

        _initFuture = _controller.initialize().then((_) {
            setState(() {});
        });

        _controller.setLooping(_looping);
        _controller.setVolume(1.0);

        /* ValueListenableBuilder'e taşındı.
        _controller.addListener(() {      setState(() {});
            if (_controller.value.hasError)      debugPrint("Video Error:
${_controller.value.errorDescription}");      });      */

    @override
    void dispose() {
        _controller.dispose();
        super.dispose();
    }

    String _format(Duration d) {
        String two(int n) => n.toString().padLeft(2, '0');
        final h = d.inHours;
        final m = d.inMinutes;
        final s = d.inSeconds;
        return h > 0 ? "${two(h)}:${two(m)}:${two(s)}" : "${two(m)}:${two(s)}";
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                backgroundColor: Colors.indigo,
                title: Center(

```

```

        child: Text(
          "Video Oynatici",
          style: TextStyle(fontWeight: FontWeight.bold, fontSize: 20),
        ),
      ),
    ),
  ),
  body: FutureBuilder(
    future: _initFuture,
    builder: (context, snapshot) {
      if (snapshot.connectionState != ConnectionState.done) {
        return const Center(child: CircularProgressIndicator());
      }

      final value = _controller.value;

      if (value.hasError) {
        return Center(child: Text("Hata: ${value.errorDescription}"));
      }

      return Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          AspectRatio(
            aspectRatio: value.aspectRatio == 0
              ? 16 / 9
              : value.aspectRatio,
            child: VideoPlayer(_controller),
          ),
          ValueListenableBuilder(
            valueListenable: _controller,
            builder: (context, VideoPlayerValue value, child) {
              if (value.hasError) {
                return Center(
                  child: Text("Hata: ${value.errorDescription}"),
                );
              }
              return controlCard(value);
            },
          ),
        ],
      );
    },
  ),
);
}

```

```

Card controlCard(VideoPlayerValue value) {
  return Card(
    elevation: 4,
    margin: EdgeInsetsGeometry.symmetric(horizontal: 2, vertical: 2),
    color: Colors.blueAccent,
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Row(
          children: [
            SizedBox(width: 5),
            Text(_format(value.position)),
            Expanded(
              child: SliderTheme(

```

```

        data: SliderTheme.of(context).copyWith(
          trackHeight: 3,
          thumbShape: const RoundSliderThumbShape(
            enabledThumbRadius: 6,
          ),
          overlayShape: const RoundSliderOverlayShape(
            overlayRadius: 12,
          ),
        ),
        child: Slider(
          value: value.position.inMilliseconds.toDouble().clamp(
            0,
            value.duration.inMilliseconds.toDouble(),
          ),
          min: 0,
          max: value.duration.inMilliseconds == 0
            ? 0
            : value.duration.inMilliseconds.toDouble(),
          onChanged: (ms) async {
            await _controller.seekTo(
              Duration(milliseconds: ms.toInt()),
            );
            setState(() {});
          },
        ),
      ),
      Text(_format(value.duration)),
      SizedBox(width: 5),
    ],
  ),
  Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      /// Oynat
      IconButton(
        iconSize: 22,
        padding: EdgeInsets.all(3),
        constraints: const BoxConstraints(),
        onPressed: () {
          setState(() {
            value.isPlaying ? _controller.pause() :
            _controller.play();
          });
        },
        icon: value.isPlaying
          ? Icon(Icons.pause)
          : Icon(Icons.play_arrow),
      ),

      /// Geri Sar
      IconButton(
        iconSize: 22,
        padding: EdgeInsets.all(3),
        constraints: const BoxConstraints(),
        onPressed: () {
          final back = value.position - Duration(seconds: 10);
          _controller.seekTo(
            back > Duration(seconds: 0) ? back : Duration.zero,
          );
        },
      ),
    ],
  ),
);

```

```

        },
        icon: Icon(Icons.replay_10),
      ),

      /// İleri Sar
      IconButton(
        iconSize: 22,
        padding: EdgeInsets.all(3),
        constraints: const BoxConstraints(),
        onPressed: () {
          final forward = value.position + Duration(seconds: 10);
          _controller.seekTo(
            forward < value.duration ? forward : value.duration,
          );
        },
        icon: Icon(Icons.forward_10),
      ),

      /// Ses
      IconButton(
        iconSize: 22,
        padding: EdgeInsets.all(3),
        constraints: const BoxConstraints(),
        onPressed: () async {
          setState(() => _muted = !_muted);
          await _controller.setVolume(_muted ? 0.0 : 1.0);
        },
        icon: _muted ? Icon(Icons.volume_off) :
Icon(Icons.volume_up),
      ),

      /// Loop
      IconButton(
        iconSize: 22,
        padding: EdgeInsets.all(3),
        constraints: const BoxConstraints(),
        onPressed: () {
          setState(() => _looping = !_looping);
          _controller.setLooping(_looping);
        },
        icon: _looping ? Icon(Icons.repeat_one) :
Icon(Icons.repeat),
        color: _looping ? Colors.red : Colors.white,
      ),

      /// Full Ekran
      IconButton(
        iconSize: 22,
        padding: EdgeInsets.all(3),
        constraints: const BoxConstraints(),
        onPressed: () {
          setState(() => _fullscreen = !_fullscreen);
        },
        icon: Icon(Icons.fullscreen),
      ),
    ],
  ),
),
),
);

```

```
    }  
  }  
  
  class VideoApp extends StatelessWidget {  
    const VideoApp({super.key});  
  
    @override  
    Widget build(BuildContext context) {  
      return MaterialApp(  
        debugShowCheckedModeBanner: false,  
        theme: ThemeData(useMaterial3: true, colorSchemeSeed:  
Colors.blueAccent),  
        home: VideoScreen(),  
      );  
    }  
  }  
}
```