Ödev 8

Created	@December 28, 2021 3:36 PM
Açıklama	Bir mobil uygulama geliştirerek arka planda Node.js kullanarak oluşturduğunuz bir web socket fonksiyonuna bir textboxa girilen değeri butona basıldığında gönderiniz. Gönderdiğiniz mesaja karşın sunucu tarafından üretilen cevap mesajını aynı ara yüzünde başka bir textbox bileşenine yazdırınız.
Link	https://www.youtube.com/watch?v=JBNNeqzALf0

Uygulamanın Youtube Linki : https://www.youtube.com/watch?v=JBNNeqzALf0

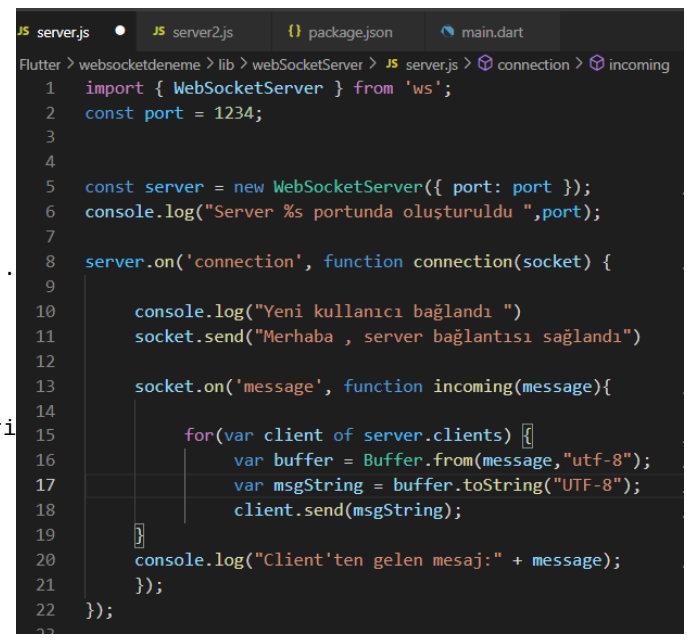
Uygulamamızda server ve client tarafı var. Client tarafından (mobil uygulama) gönderdiğimiz mesajı server (web - node.js) ‘a ileteceğiz ve server’dan tekrar client’e göndereceğiz.

Bunun için websocket teknolojisini kullanarak TCP üzerinden mesajlarımızı iletebileceğiz. Çift yönlü bir veri alışverişi yapabiliriz .Şimdi ilk olarak server’ı nasıl kuracağımıza bakalım.

Server

Server’ı node.js ile yazıyoruz. Hangi porttan server yaratılacağı önemli .

```
import { WebSocketServer } from 'ws';
const port = 1234;


const server = new WebSocketServer({ port: port });    // server oluşturuldu
console.log("Server %s portunda oluşturuldu ",port);

server.on('connection', function connection(socket) {  // servera bağlantı olduğunda ...

     console.log("Yeni kullanıcı bağlandı ")
     socket.send("Merhaba , server bağlantısı sağlandı")

     socket.on('message', function incoming(message){  // gelen mesaj için içeridekileri

          for(var client of server.clients) {               // Her bir client'e
               var buffer = Buffer.from(message,"utf-8");   // mesajları buffer ile
               var msgString = buffer.toString("UTF-8");    // düzelterek
               client.send(msgString);                      // client'e gönderir.
     }
     console.log("Client'ten gelen mesaj:" + message); // konsola aynı mesaj yazdırılır.
     });
});
```



İlk olarak server değişkenine WebSocketServer kullanarak 1234 portundan yeni bir server oluşturduk. Bu değişken üzerinden server.on ile bağlanma durumunda neler yapılacağını oluşturduk. Bağlanma durumu olduğunda socket parametresi ile mesajı alacağımız ve geri ileteceğimiz kısmı oluşturduk. Gönderilen mesajlar string tipinde iletilmesi için Buffer kullandık. Buffer’ın toString metodu ile bunu gerçekleştirdik. client.send ile client’e mesajı ilettik.

Server’ı başlatmak için terminale

```
node server.js
```

yazdık ve server’ı başlattık.

Client

```
import 'package:web_socket_channel/io.dart';
import 'package:web_socket_channel/web_socket_channel.dart';
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: "WebSocket",
      home: MyHomePage(
        title: "WebSocket App",
      ),
    );
  }
}
```

web_socket_channel/io.dart ve web_socket_channel/web_socket_channel.dart import edildi. main fonksiyonu içerisinde uygulama başlatıldı

MyApp sınıfı bir MaterialApp ile MyHomePage sınıfımızı oluşturdu.

```
class MyHomePage extends StatefulWidget {
  const MyHomePage({Key? key, required this.title,}) : super(key: key);
  final String title;

  @override
  _MyHomePageState createState() => _MyHomePageState();
}
```

MyHomePage sınıfı bir StatefulWidget. Bu sınıfın durumu _MyHomePageState sınıfında oluşturulur.

```
class _MyHomePageState extends State<MyHomePage> {
  final TextEditingController controller = TextEditingController();                 // text için kontrolcü
  WebSocketChannel channel = IOWebSocketChannel.connect('ws://192.168.18.6:1234');  // bağlanacağımız server kanalı
  /*
  final _channel = WebSocketChannel.connect(Uri.parse("http://10.0.2.2:1234"),);
  */

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [

            Form(
              child: TextFormField(
                controller: controller,
                decoration: const InputDecoration(
                  border: OutlineInputBorder(),
                  labelText: 'Mesajını gir'
                ),
              ),

            ),
            const SizedBox(height: 24),
            StreamBuilder(
              stream: channel.stream,
              builder: (context, snapshot) {
                return Text(snapshot.hasData ? '${snapshot.data}' : '');
              },
```

```
            )
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: sendMessage,
        tooltip: 'Send message',
        child: const Icon(Icons.send),
      ),
    );
  }
  void sendMessage() async {
    if (controller.text.isNotEmpty) {
      channel.sink.add(controller.text);
    }
  }
  @override
  void dispose() {
    channel.sink.close();
    super.dispose();
  }
}
```

Burada gördüğümüz controller ve channel deişkenleri önemli . Bunlar üzerinden işlemlerimizi yürüteceğiz. controller değişkeni textField alanına yazılan text'i kontrol edecek.

channel ise IOWebSocketChannel.connect ile oluşturduğumuz server'a bağlanacak.

Uygulama ekranımız Scaffold widget ile oluşturuldu. AppBar ile başlık verildi. Gövde kısmında ise Padding widget'tan yararlanıldı. Bu widget'ın içerisinde Column widget'ı ile bir dikey kolon oluşturulmuş oldu.

Column widget'ın ilk child'ı bir Form. Onun child'ı ise TextFormField. Bunlar ekranda gördüğümüz , en üstteki yazı yazabileceğimiz kutucuğu oluşturacaklar. controller parametresine kendi oluşturduğumuz controller'ı atadık.

Ardından diğer child'ımız bir SizedBox. Bununla kutucuk altına bir boşluk oluşturmuş olduk. Server'dan geri dönen bilgiler buradaki boşluğa gelecektir.

Son child'ımız ise bir StreamBuilder. Bu sınıf ile gelen veriyi (channel üzerinden gelen) ekrana yazdırırız. Burada builder parametresine context ve snaphot bilgisi verilmiş. snapshot üzerinden gelen veri (eğer varsa) ekrana yazdırırız.

Scaffold'ın floatingActionButton parametresine FloatingActionButton widget'ı yardımıyla yeni bir buton oluşturduk. Bu butona tıklandığında sendMessage metodu çalışacak.

sendMessage metodu controller üzerinden kutucuğu boş mu diye kontrol edecek. Eğer boş değilse server'a text bilgisi eklenecek.

Ve son olarak dispose metodu. Bu metod ile değişkenlere ayrılan belleği boşaltıcak ve channel'ı kapatabileceğiz.

Uygulama Ekran Görüntüleri

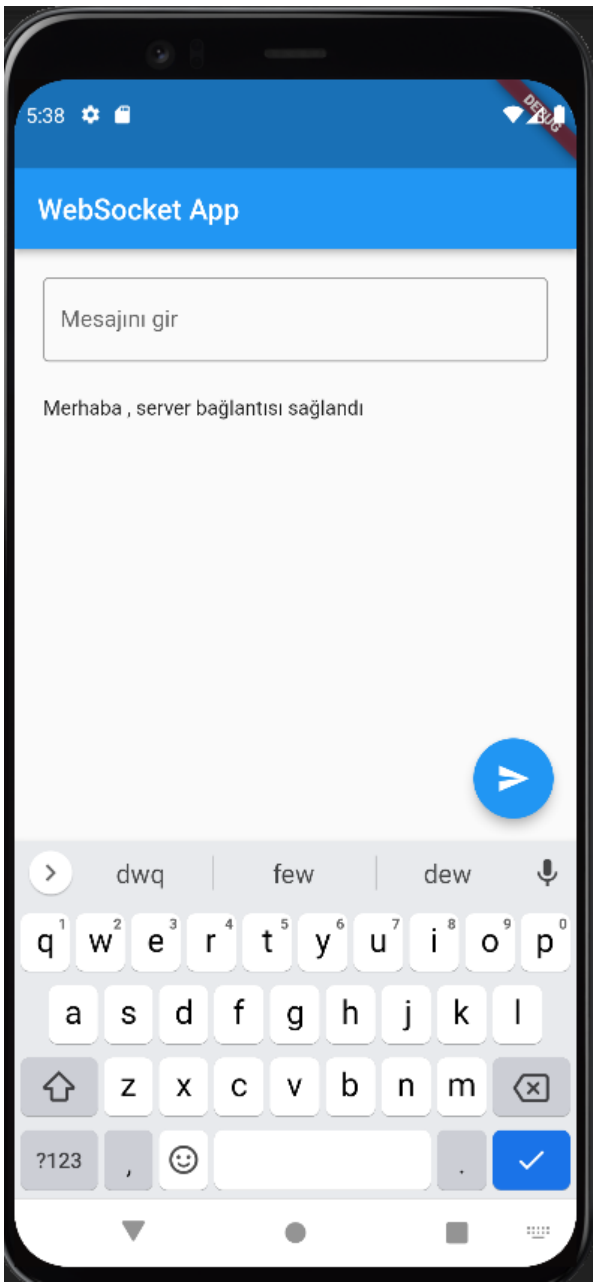
deneme1.dart | main.dart | pubspec.yaml

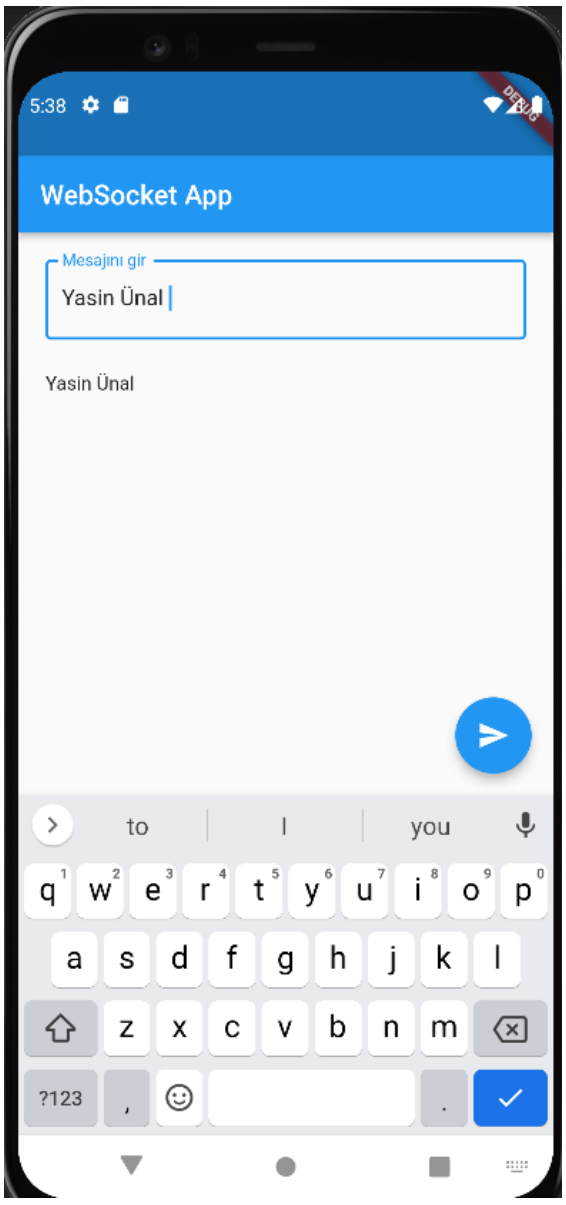
```dart
import 'package:web_socket_channel/io.dart';
import 'package:web_socket_channel/web_socket_channel.dart';
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: "WebSocket",
      home: MyHomePage(
        title: "WebSocket App",
      ),  // MyHomePage
    );  // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({Key? key, required this.title,}) : super(key: key);
  final String title;

  @override
  _MyHomePageState createState() => _MyHomePageState();
}
```

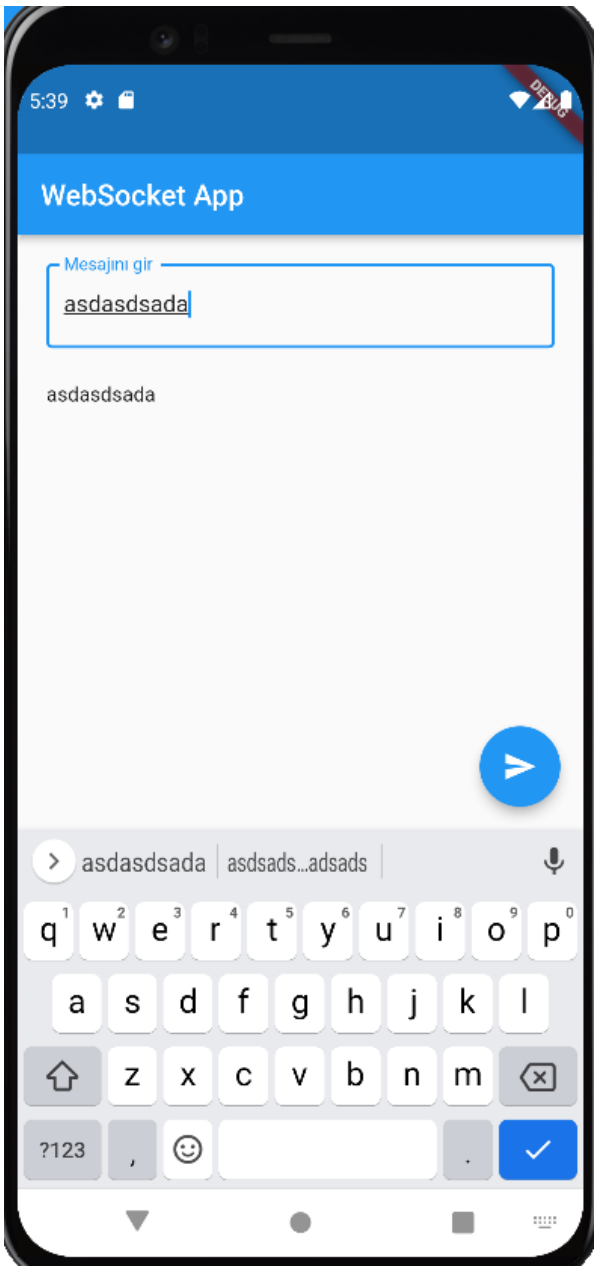
```dart
class _MyHomePageState extends State<MyHomePage> {
  final TextEditingController controller = TextEditingController();                // text için kontrolcü
  WebSocketChannel channel = IOWebSocketChannel.connect('ws://192.168.18.6:1234');  // bağlanacağımız server kanalı
  /*
  final _channel = WebSocketChannel.connect(Uri.parse("http://10.0.2.2:1234"),);
  */

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),  // AppBar
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [

            Form(
              child: TextFormField(
                controller: controller,
                decoration: const InputDecoration(
                  border: OutlineInputBorder(),
                  labelText: 'Mesajını gir'
                ),  // InputDecoration
              ),  // TextFormField
            ),  // Form
            const SizedBox(height: 24),
```

```
            StreamBuilder(
              stream: channel.stream,
              builder: (context, snapshot) {
                return Text(snapshot.hasData ? '${snapshot.data}' : '');
              },
            )  // StreamBuilder
          ],
        ),  // Column
      ),  // Padding
      floatingActionButton: FloatingActionButton(
        onPressed: sendMessage,
        tooltip: 'Send message',
        child: const Icon(Icons.send),
      ), // This trailing comma makes auto-formatting nicer for build methods.  // FloatingActionButton
    );  // Scaffold
  }

  void sendMessage() async {
    if (controller.text.isNotEmpty) {
      channel.sink.add(controller.text);
    }
  }

  @override
  void dispose() {
    channel.sink.close();
    super.dispose();
  }
```

Çalışırken Ekran Görüntüleri


5:38
WebSocket App
Mesajını gir
Merhaba , server bağlantısı sağlandı
dwq
few
dew


5:38
WebSocket App
Mesajını gir
Yasin Ünal
Yasin Ünal
to
I
you


5:39
WebSocket App
Mesajını gir
asdasdsada
asdasdsada
asdasdsada
asdsads...adsads