Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ: «Информатика и системы управления»
КАФЕДРА: «Теоретическая информатика и компьютерные технологии»

Лабораторная работа №6
«Реализация взаимодействия Flutter-приложения с сервером на WebSocket»

по курсу
«Разработка мобильных приложений»

Выполнил:
студент группы ИУ9-72Б
Караник А.А.

Проверено:
Посевин Д.П.

Москва, 2024

Цель работы

Создать сервер на WebSocket с использованием Dart и модифицировать Flutter-приложение для обмена данными с этим сервером в реальном времени. Реализовать функционал кликера с отправкой и получением данных через WebSocket.

Реализация

Исходный код flutter-приложения:

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'lab6',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'lab6'),
    );
  }
}

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

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  final _channel = WebSocketChannel.connect(Uri.parse('ws://194.67.88.154:8100'));
  final TextEditingController _textController = TextEditingController();
  int _counter = 0;
  String _serverResponse = '';

  @override
  void dispose() {
    _channel.sink.close();
    _textController.dispose();
    super.dispose();
  }

  void _sendCounter(String command) {
    if (command == 'GET') {
      _channel.sink.add('GET');
    } else if (command == 'POST') {
      _channel.sink.add('POST $_counter');
    }
  }

  void _incrementCounter() {
```

```
    setState(() {
      _counter++;
    });
    _sendCounter('POST');
  }

  void _decrementCounter() {
    setState(() {
      _counter--;
    });
    _sendCounter('POST');
  }

  void _sendValueToServer(int value) {
    setState(() {
      _counter = value;
    });
    _sendCounter('POST');
  }

  void _getValueFromServer() {
    _sendCounter('GET');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            TextField(
              controller: _textController,
              decoration: const InputDecoration(
                labelText: 'Введите начальное значение',
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: () {
                final value = int.parse(_textController.text);
                _sendValueToServer(value);
              },
              child: const Text('POST INIT'),
            ),
            const Text('You have pushed the button this many times:'),
            Text(
              '$_counter'
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: _getValueFromServer,
              child: const Text('GET Counter'),
            ),
            const SizedBox(height: 20),
            StreamBuilder(
              stream: _channel.stream,
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  _serverResponse = snapshot.data.toString();
```

```
                if (int.tryParse(_serverResponse) != null) {
                  _counter = int.parse(_serverResponse);
                }
              }
              return Text(
                _serverResponse,
                style: const TextStyle(color: Colors.green),
              );
            },
          ),
        ],
      ),
    ),
    floatingActionButton: Column(
      mainAxisAlignment: MainAxisAlignment.end,
      children: <Widget>[
        FloatingActionButton(
          onPressed: _incrementCounter,
          tooltip: 'Increment',
          child: const Icon(Icons.add),
        ),
        const SizedBox(height: 10),
        FloatingActionButton(
          onPressed: _decrementCounter,
          tooltip: 'Decrement',
          child: const Icon(Icons.remove),
        ),
      ],
    ),
  );
 }
}
```

Исходный код сервера:

```
import 'dart:io';

const String filePath = 'value.txt';

void main() async {
  final server = await HttpServer.bind(InternetAddress.anyIPv4, 8100);
  print('WebSocket server running on ws://${server.address.address}:${server.port}');

  await for (HttpRequest request in server) {
    if (WebSocketTransformer.isUpgradeRequest(request)) {
      final socket = await WebSocketTransformer.upgrade(request);
      handleWebSocket(socket);
    } else {
      request.response
        ..statusCode = HttpStatus.forbidden
        ..write('Only WebSocket connections are allowed')
        ..close();
    }
  }
}

void handleWebSocket(WebSocket socket) {
  print('Client connected');

  socket.listen((message) async {
    if (message == 'GET') {
      await handleGetRequest(socket);
    } else if (message.startsWith('POST')) {
```

```
      final valueString = message.split(' ')[1];
      try {
        final value = int.parse(valueString);
        await handlePostRequest(socket, value);
      } catch (e) {
        socket.add('Error: Invalid integer value');
      }
    } else {
      socket.add('Error: Invalid command');
    }
  }, onDone: () {
    print('Client disconnected');
  });
}

Future<void> handleGetRequest(WebSocket socket) async {
  try {
    final file = File(filePath);
    if (await file.exists()) {
      final content = await file.readAsString();
      socket.add('$content');
    } else {
      socket.add('No value stored yet.');
    }
  } catch (e) {
    socket.add('Error: Unable to read file');
  }
}

Future<void> handlePostRequest(WebSocket socket, int value) async {
  try {
    final file = File(filePath);
    await file.writeAsString(value.toString());
    socket.add('Stored value: $value');
  } catch (e) {
    socket.add('Error: Unable to write file');
  }
}
```

Результаты

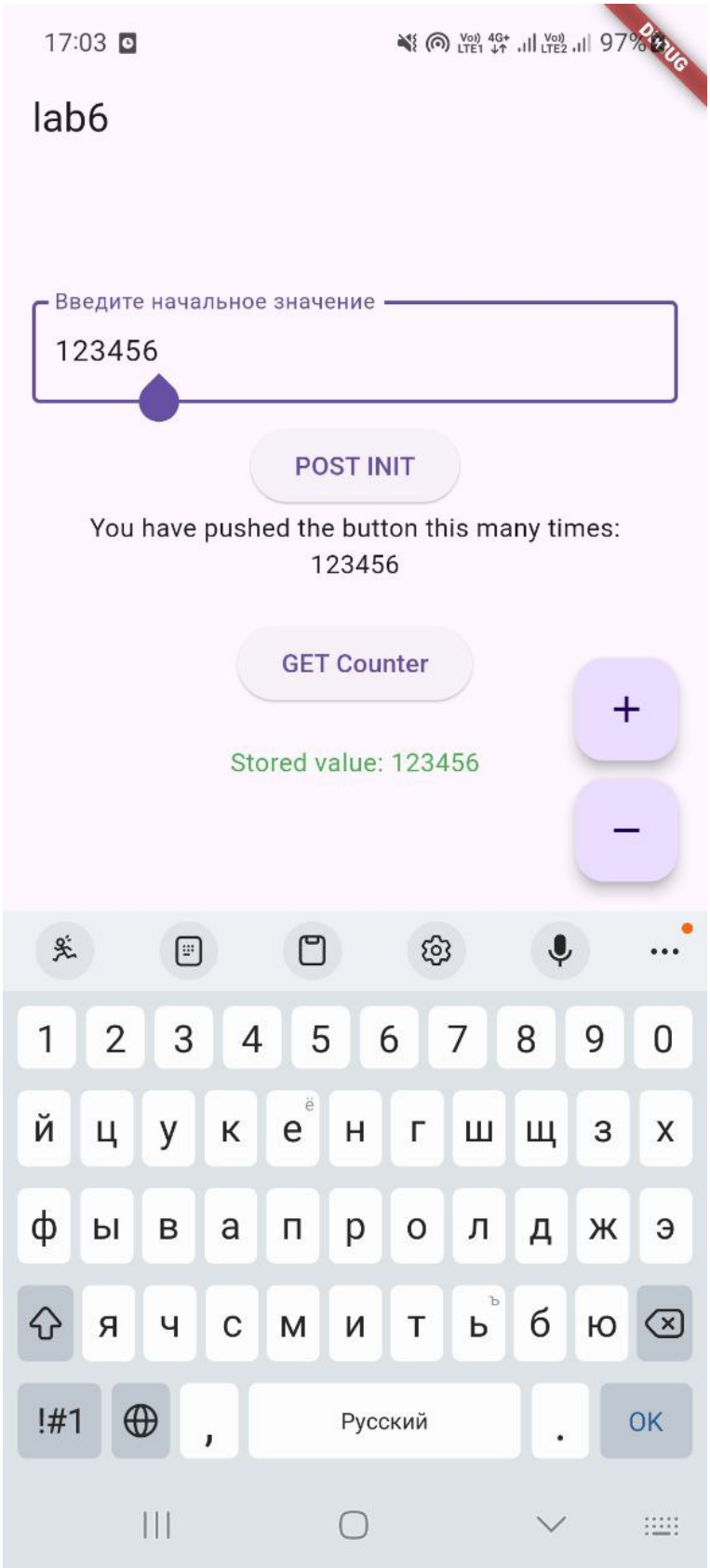

Рис. 1: результаты

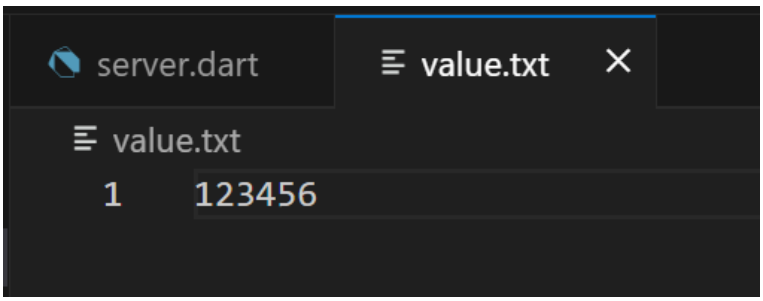

Рис. 2: результаты

Вывод

В ходе работы успешно реализовано взаимодействие клиента и сервера через WebSocket, что обеспечило обмен данными в реальном времени. Применение WebSocket позволило сократить задержки и упростить обмен данными между Flutter-приложением и сервером, что делает архитектуру приложения более эффективной для задач, требующих частых обновлений.