|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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» |

**Отчет по лабораторной работе № 3**

**«Обмен данными с сервером на языке Dart»**

***по курсу***

***«Разработка мобильных приложений»***

Выполнил:

Студент группы ИУ9-72Б

Караник А.А.

Проверил:

Посевин Д.П.

1. *г.*

## Цель

Целью лабораторной работы является разработка клиент-серверного приложения, состоящего из сервера на языке Dart, который принимает и обрабатывает целочисленное значение через POST и GET запросы, а также мобильного приложения на Flutter, позволяющего взаимодействовать с сервером. Мобильное приложение должно поддерживать инкремент и декремент числового значения, отправлять данные на сервер, получать их с сервера, а также вводить произвольное значение для отправки через POST запрос.

## Практическая реализация

Код приложения:

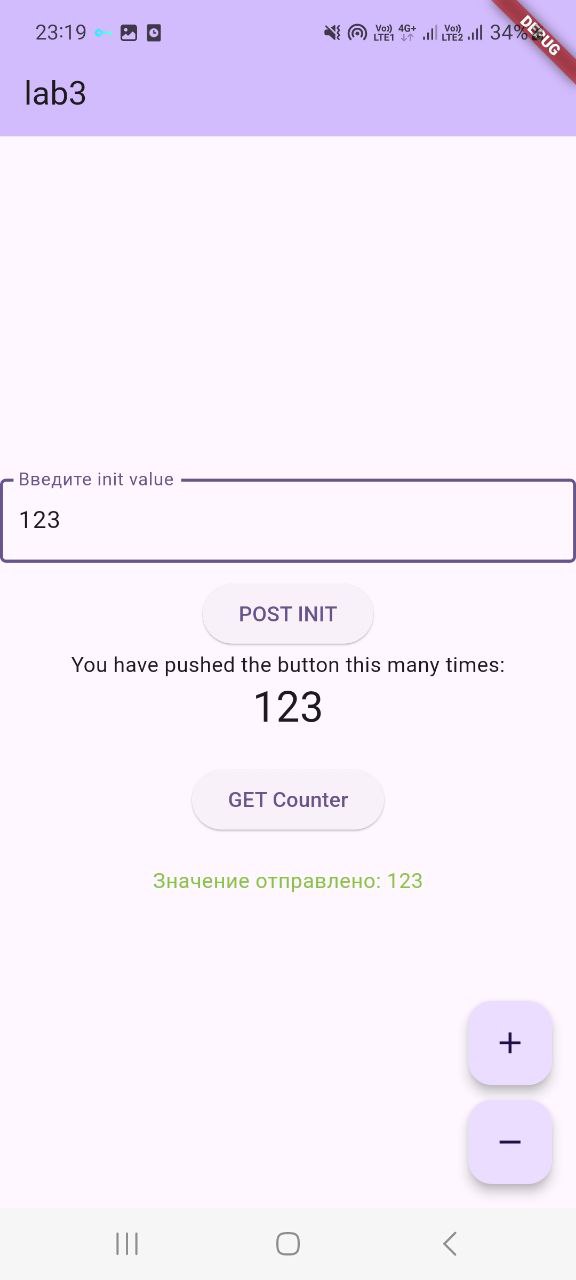
import 'package:flutter/material.dart';  
import 'package:http/http.dart' as http;  
  
void main() {  
 runApp(const MyApp());  
}  
  
class MyApp extends StatelessWidget {  
 const MyApp({super.key});  
  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 title: 'lab3',  
 theme: ThemeData(  
 colorScheme: ColorScheme.fromSeed(seedColor: Colors.*deepPurple*),  
 useMaterial3: true,  
 ),  
 home: const MyHomePage(title: 'lab3'),  
 );  
 }  
}  
  
class MyHomePage extends StatefulWidget {  
 const MyHomePage({super.key, required this.title});  
  
 final String title;  
  
 @override  
 State<MyHomePage> createState() => \_MyHomePageState();  
}  
  
class \_MyHomePageState extends State<MyHomePage> {  
 int \_counter = 0;  
 String \_serverResponse = '';  
 final TextEditingController \_textController = TextEditingController();  
  
 // Функция инкремента  
 void \_incrementCounter() {  
 setState(() {  
 \_counter++;  
 });  
 \_sendCounterToServer();  
 }  
  
 // Функция декремента  
 void \_decrementCounter() {  
 setState(() {  
 \_counter--;  
 });  
 \_sendCounterToServer();  
 }  
  
 // Отправка значения на сервер (POST)  
 Future<void> \_sendCounterToServer() async {  
 final url = Uri.*parse*('http://194.67.88.154:8100/$\_counter');  
 try {  
 final response = await http.post(url);  
  
 if (response.statusCode == 200) {  
 setState(() {  
 \_serverResponse = 'Значение отправлено: $\_counter';  
 });  
 } else {  
 setState(() {  
 \_serverResponse =  
 'Не удалось отправить значение. Сервер ответил кодом статуса: ${response.statusCode}';  
 });  
 }  
 } catch (e) {  
 setState(() {  
 \_serverResponse = 'Ошибка отправки значения: $e';  
 });  
 }  
 }  
  
 Future<void> \_sendValueToServer(int value) async {  
 final url = Uri.*parse*('http://194.67.88.154:8100/$value');  
 try {  
 final response = await http.post(url);  
  
 if (response.statusCode == 200) {  
 setState(() {  
 \_serverResponse = 'Значение отправлено: $value';  
 \_counter = value;  
  
 });  
 } else {  
 setState(() {  
 \_serverResponse =  
 'Не удалось отправить значение. Сервер ответил кодом статуса: ${response.statusCode}';  
 });  
 }  
 } catch (e) {  
 setState(() {  
 \_serverResponse = 'Ошибка отправки значения: $e';  
 });  
 }  
 }  
  
 // Получение значения с сервера (GET)  
 Future<void> \_getValueFromServer() async {  
 final url = Uri.*parse*('http://194.67.88.154:8100');  
 try {  
 final response = await http.get(url);  
  
 if (response.statusCode == 200) {  
 setState(() {  
 \_counter = int.*parse*(response.body);  
 \_serverResponse = 'Счетчик обновлен с сервера: $\_counter';  
 });  
 } else {  
 setState(() {  
 \_serverResponse =  
 'Не удалось получить значение. Сервер ответил кодом статуса: ${response.statusCode}';  
 });  
 }  
 } catch (e) {  
 setState(() {  
 \_serverResponse = 'Ошибка получения значения: $e';  
 });  
 }  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 backgroundColor: Theme.*of*(context).colorScheme.inversePrimary,  
 title: Text(widget.title),  
 ),  
 body: Center(  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: <Widget>[  
 const SizedBox(height: 20),  
 TextField(  
 controller: \_textController,  
 decoration: const InputDecoration(  
 labelText: 'Введите init value',  
 border: OutlineInputBorder(),  
 ),  
 ),  
 const SizedBox(height: 10),  
 ElevatedButton(  
 onPressed: () {  
 \_sendValueToServer(int.*parse*(\_textController.text));  
 },  
 child: const Text('POST INIT'),  
 ),  
 const Text(  
 'You have pushed the button this many times:',  
 ),  
 Text(  
 '$\_counter',  
 style: Theme.*of*(context).textTheme.headlineMedium,  
 ),  
 const SizedBox(height: 20),  
 ElevatedButton(  
 onPressed: \_getValueFromServer,  
 child: const Text('GET Counter'),  
 ),  
 const SizedBox(height: 20),  
 Text(  
 \_serverResponse,  
 style: const TextStyle(color: Colors.*lightGreen*),  
 ),  
 ],  
 ),  
 ),  
 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*),  
 ),  
 ],  
 ),  
 );  
 }  
}

Код сервера:

import 'dart:io';  
  
const String filePath = 'value.txt';  
  
void main() async {  
 final server = await HttpServer.*bind*(InternetAddress.*anyIPv4*, 8100);  
 print('Server running on http://${server.address.address}:${server.port}');  
  
 await for (HttpRequest request in server) {  
 final pathSegments = request.uri.pathSegments;  
  
 if (request.method == 'GET' && pathSegments.length == 0) {  
 await handleGetRequest(request);  
 } else if (request.method == 'POST' && pathSegments.length == 1) {  
 final valueString = pathSegments[0];  
  
 try {  
 final value = int.*parse*(valueString);  
 await handlePostRequest(request, value);  
 } catch (e) {  
 handleInvalidRequest(request, 'Invalid integer value: $valueString');  
 }  
 } else {  
 handleInvalidRequest(request, 'Invalid route or method');  
 }  
 }  
}  
  
Future<void> handleGetRequest(HttpRequest request) async {  
 try {  
 final file = File(filePath);  
  
 if (await file.exists()) {  
 String content = await file.readAsString();  
 request.response  
 ..statusCode = HttpStatus.*ok* ..write('$content');  
 } else {  
 request.response  
 ..statusCode = HttpStatus.*ok* ..write('No value stored yet.');  
 }  
 } catch (e) {  
 request.response  
 ..statusCode = HttpStatus.*internalServerError* ..write('Error reading file');  
 } finally {  
 await request.response.close();  
 }  
}  
  
Future<void> handlePostRequest(HttpRequest request, int value) async {  
 try {  
 final file = File(filePath);  
 await file.writeAsString(value.toString());  
  
 request.response  
 ..statusCode = HttpStatus.*ok* ..write('Value stored: $value');  
 } catch (e) {  
 request.response  
 ..statusCode = HttpStatus.*internalServerError* ..write('Error writing to file');  
 } finally {  
 await request.response.close();  
 }  
}  
  
void handleInvalidRequest(HttpRequest request, String message) {  
 request.response  
 ..statusCode = HttpStatus.*badRequest* ..write(message);  
 request.response.close();  
}

## Результаты

Введем в текстовое поле некоторое значение и нажмем на кнопку “POST INIT”



Посмотрим значение в файле на сервере и убедимся, что оно совпадает с отправленным.



Были проведены также проверки изменения счетчика и получения значения с сервера по нажатию на соответствующие кнопки.

## Вывод

В ходе выполнения лабораторной работы был реализован сервер на Dart, способный принимать через POST запрос и сохранять целочисленное значение в файл. Также было создано мобильное приложение на Flutter, включающее кнопки для инкремента и декремента числа, поле для ввода произвольного значения, а также соответствующие кнопки для отправки и получения данных с сервера. Работа продемонстрировала навыки создания клиент-серверных взаимодействий и использования сетевых запросов в приложении Flutter.