



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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 № 8
по курсу «Разработка мобильных приложений»
«SSH-сервер»

Студент группы ИУ9-72Б Шемякин В.А.

Преподаватель Посевин Д. П.

Москва 2025

1 Задание

Реализовать приложение для подключения в ssh-серверу, которое позволяет работать с консолью на сервере

2 Результаты

Исходный код программы представлен в листинге 1.

```
1 import 'dart:convert';
2 import 'package:flutter/material.dart';
3 import 'package:dartssh2/dartssh2.dart';
4
5 void main() {
6   runApp(const MyApp());
7 }
8
9 class MyApp extends StatelessWidget {
10   const MyApp({super.key});
11   @override
12   Widget build(BuildContext context) {
13     return MaterialApp(
14       title: 'SSH Console',
15       debugShowCheckedModeBanner: false,
16       home: const Home(),
17       theme: ThemeData(useMaterial3: true),
18     );
19   }
20 }
21
22 class Home extends StatefulWidget {
23   const Home({super.key});
24   @override
25   State<Home> createState() => _HomeState();
26 }
27
28 class _HomeState extends State<Home> {
29   final hostC = TextEditingController(text: 'localhost');
30   final portC = TextEditingController(text: '22');
31   final userC = TextEditingController();
32   final passC = TextEditingController();
33   final cmdC = TextEditingController();
34 }
```

```

35 SSHClient? client;
36 bool connecting = false;
37 bool connected = false;
38 String log = '';
39
40 Future<void> connect() async {
41     if (connecting || connected) return;
42     setState(() => connecting = true);
43     try {
44         final socket = await SSHSocket.connect(hostC.text.trim(), int
45             .tryParse(portC.text.trim()) ?? 22);
46         final c = SSHClient(socket, username: userC.text.trim(),
47             onPasswordRequest: () => passC.text);
48         await c.ping();
49         setState(() {
50             client = c;
51             connected = true;
52             log = 'CONNECTED ${hostC.text}:${portC.text}\n$log';
53         });
54     } catch (e) {
55         setState(() {
56             log = 'ERROR: $e\n$log';
57         });
58     } finally {
59         setState(() => connecting = false);
60     }
61 }
62
63 Future<void> disconnect() async {
64     if (client == null) return;
65     try {
66         client!.close();
67         await client!.done;
68     } catch (_) {}
69     setState(() {
70         connected = false;
71         client = null;
72         log = 'DISCONNECTED\n$log';
73     });
74 }
75
76 Future<void> runCommand() async {
77     final c = client;
78     final cmd = cmdC.text.trim();
79     if (c == null || cmd.isEmpty) return;

```

```

78     setState(() {
79         log = '\$ $cmd\n$log';
80         cmdC.clear();
81     });
82     try {
83         final out = await c.run(cmd);
84         final text = utf8.decode(out, allowMalformed: true);
85         setState(() {
86             log = '$text\n$log';
87         });
88     } catch (e) {
89         setState(() {
90             log = 'ERROR: $e\n$log';
91         });
92     }
93 }
94
95 @override
96 void dispose() {
97     hostC.dispose();
98     portC.dispose();
99     userC.dispose();
100    passC.dispose();
101    cmdC.dispose();
102    super.dispose();
103 }
104
105
106 @override
107 Widget build(BuildContext context) {
108     final canSend = connected && !connecting;
109     return Scaffold(
110         appBar: AppBar(
111             title: const Text('SSH Console'),
112             actions: [
113                 TextButton(
114                     onPressed: canSend ? disconnect : connect,
115                     child: Text(
116                         connected ? 'Disconnect' : (connecting ? 'Connecting
... ' : 'Connect'),
117                 ),
118             ],
119         ),
120     ),
121     body: SafeArea(

```

```

122         child: Padding(
123           padding: const EdgeInsets.all(12),
124           child: Column(
125             children: [
126               Row(
127                 children: [
128                   Expanded(
129                     flex: 3,
130                     child: TextField(
131                       controller: hostC,
132                       decoration: const InputDecoration(labelText:
'Host '),
133                       enabled: !connected && !connecting,
134                     ),
135                   ),
136                   const SizedBox(width: 8),
137                   Expanded(
138                     child: TextField(
139                       controller: portC,
140                       decoration: const InputDecoration(labelText:
'Port '),
141                       keyboardType: TextInputType.number,
142                       enabled: !connected && !connecting,
143                     ),
144                   ),
145                 ],
146               ),
147               const SizedBox(height: 8),
148               Row(
149                 children: [
150                   Expanded(
151                     child: TextField(
152                       controller: userC,
153                       decoration: const InputDecoration(labelText:
'Username '),
154                       enabled: !connected && !connecting,
155                     ),
156                   ),
157                   const SizedBox(width: 8),
158                   Expanded(
159                     child: TextField(
160                       controller: passC,
161                       decoration: const InputDecoration(labelText:
'Password '),
162                       obscureText: true,

```

```

163         enabled: !connected && !connecting,
164     ),
165 ),
166 ],
167 ),
168 const SizedBox(height: 12),
169 Expanded(
170     child: Container(
171         width: double.infinity,
172         decoration: BoxDecoration(
173             border: Border.all(color: Theme.of(context).
dividerColor),
174             borderRadius: BorderRadius.circular(8),
175         ),
176         padding: const EdgeInsets.all(8),
177         child: SingleChildScrollView(
178             reverse: true,
179             child: SelectableText(
180                 log,
181                 style: const TextStyle(fontFamily: 'monospace',
fontSize: 13),
182             ),
183         ),
184     ),
185 ),
186 const SizedBox(height: 8),
187 Row(
188     children: [
189         Expanded(
190             child: TextField(
191                 controller: cmdC,
192                 decoration: const InputDecoration(hintText: '
Enter'),
193                 onSubmitted: (_) => runCommand(),
194                 enabled: canSend,
195             ),
196         ),
197         const SizedBox(width: 8),
198         FilledButton(
199             onPressed: canSend ? runCommand : null,
200             child: const Text('Send'),
201         ),
202     ],
203 ),
204 ],

```

```
205         ),
206         ),
207     ),
208 );
209 }
210 }
```

Результат запуска представлен на рисунке 1.

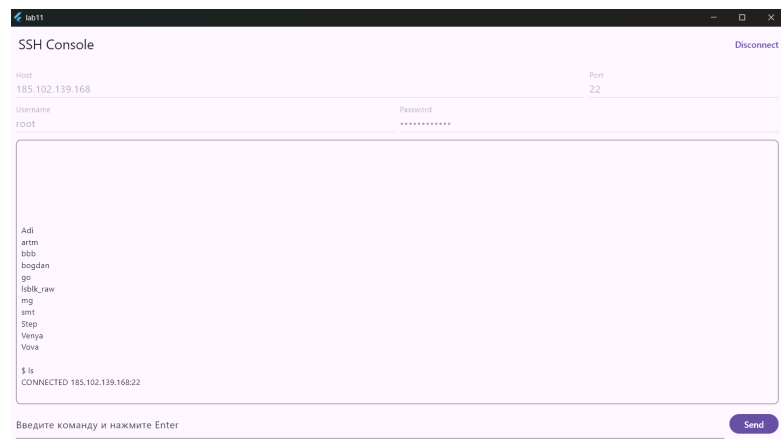


Рис. 1 — Результат