



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 № 4
по курсу «Разработка мобильных приложений»
«Приложение выполняющее функцию FTP-клиента,
SSH-клиента и SMTP-клиента»

Студентка группы ИУ9-72Б Самохвалова П. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Задание

Реализовать мобильное приложение выполняющее функцию FTP-клиента, SSH-клиента и SMTP-клиента. В приложении должен быть следующий функционал:

1. Экран с параметрами FTP-подключения, реализованный в виде формы: значения полей FTP-доступа должны храниться постоянно, другими словами после ввода данных при последующем запуске приложения ранее введенные значения полей должны сохраняться и загружаться в поля ввода, если пользователь хочет изменить параметры FTP-доступа, то он редактирует значения соответствующих полей и пересохраняет форму. На отдельном экране необходимо реализовать интерфейс выполнения FTP-команд: вывода содержимого директории, перехода в другую директорию, создания директории, загрузки файла, скачивание файла, результат ответа от FTP-сервера должен записываться в виджет вывода данных под формой ввода команд.

2. Экран с параметрами SSH-подключения, реализованный в виде формы: значения полей SSH-доступа должны храниться постоянно, другими словами после ввода данных при последующем запуске приложения ранее введенные значения полей должны сохраняться и загружаться в поля ввода, если пользователь хочет изменить параметры SSH-доступа, то он редактирует значения соответствующих полей и пересохраняет форму. На отдельном экране необходимо реализовать интерфейс выполнения консольных команд выполняемых на SSH-сервере. Результат ответа от SSH-сервера должен записываться в виджет вывода данных под формой ввода команд.

3. Экран с параметрами SMTP-подключения, реализованный в виде формы, значения полей SMTP-аутентификации должны храниться постоянно, другими словами после ввода данных при последующем запуске приложения ранее введенные значения полей должны сохраняться и загружаться в поля ввода, если пользователь хочет изменить параметры smtp-аутентификации, то он редактирует значения соответствующих полей и пересохраняет форму. На отдельном экране необходимо реализовать форму отправки письма, при этом поля формы должны состоять из полей: адрес электронной почты, тема письма, сообще-

ние. Результат ответа от SMTP-сервера должен записываться в виджет вывода данных под формой ввода команд.

2 Практическая реализация

Исходный код программы представлен в листинге 1.

Листинг 1: FTP-клиент, SSH-клиент и SMTP-клиент

```
1 import 'package:flutter/material.dart';
2 import 'ftp_filelist_screen.dart';
3 import 'package:shared_preferences/shared_preferences.dart';
4 import 'dart:io';
5
6 class MyApp extends StatelessWidget {
7   @override
8   Widget build(BuildContext context) {
9     return MaterialApp(
10       home: FTPScreen(),
11     );
12   }
13 }
14
15 class FTPScreen extends StatefulWidget {
16   @override
17   _FTPScreenState createState() => _FTPScreenState();
18 }
19
20 class _FTPScreenState extends State<FTPScreen> {
21   late TextEditingController _ftpHostController = TextEditingController();
22   late TextEditingController _ftpUsernameController =
23     TextEditingController();
23   late TextEditingController _ftpPasswordController =
24     TextEditingController();
24   // late TextEditingController _ftpPortController =
25     TextEditingController();
25
26   late SharedPreferences _prefs;
27
28   @override
29   void initState() {
30     super.initState();
31     _init();
32   }
```

```

33
34 void _init() async {
35     _prefs = await SharedPreferences.getInstance();
36     _initTextControllers();
37 }
38
39 void _initTextControllers() {
40     _ftpHostController.text = _prefs.getString('ftpHost') ?? '';
41     _ftpUsernameController.text = _prefs.getString('ftpUsername') ?? '';
42     _ftpPasswordController.text = _prefs.getString('ftpPassword') ?? '';
43     // _ftpPortController.text = _prefs.getString('ftpPort') ?? '';
44 }
45
46 void _saveFtpSettings() {
47     _prefs.setString('ftpHost', _ftpHostController.text);
48     _prefs.setString('ftpUsername', _ftpUsernameController.text);
49     _prefs.setString('ftpPassword', _ftpPasswordController.text);
50     // _prefs.setString('ftpPort', _ftpPortController.text);
51     ScaffoldMessenger.of(context).showSnackBar(
52         SnackBar(content: Text('FTP settings saved')),
53     );
54 }
55
56 void _navigateToFTPCommandsScreen() {
57     String host = _ftpHostController.text;
58     String username = _ftpUsernameController.text;
59     String password = _ftpPasswordController.text;
60     // String port = _ftpPortController.text;
61
62     Navigator.push(
63         context,
64         MaterialPageRoute(
65             builder: (context) => FileListScreen(
66                 server: host,
67                 login: username,
68                 password: password,
69                 // port: port,
70             ),
71         ),
72     );
73 }
74
75 @override
76 Widget build(BuildContext context) {
77     return Scaffold(
78         appBar: AppBar(

```

```

79         title: Text('FTP Data Entry Form'),
80     ),
81     body: Padding(
82         padding: EdgeInsets.all(16.0),
83         child: Column(
84             mainAxisAlignment: MainAxisAlignment.center,
85             children: <Widget>[
86                 TextField(
87                     controller: _ftpHostController,
88                     decoration: InputDecoration(labelText: 'FTP Server'),
89                 ),
90                 TextField(
91                     controller: _ftpUsernameController,
92                     decoration: InputDecoration(labelText: 'FTP Login'),
93                 ),
94                 TextField(
95                     controller: _ftpPasswordController,
96                     decoration: InputDecoration(labelText: 'FTP Password'),
97                 ),
98                 // TextField(
99                 //     controller: _ftpPortController,
100                 //     decoration: InputDecoration(labelText: 'FTP Port'),
101                 // ),
102                 SizedBox(height: 10),
103                 ElevatedButton(
104                     onPressed: _saveFtpSettings,
105                     child: Text('Save'),
106                 ),
107                 SizedBox(height: 10),
108                 ElevatedButton(
109                     onPressed: _navigateToFTPCommandsScreen,
110                     child: Text('Connect to FTP server'),
111                 ),
112             ],
113         ),
114     ),
115 );
116 }
117 }
118
119 import 'dart:io';
120
121 import 'package:flutter/material.dart';
122 import 'package:ftpconnect/ftpconnect.dart';
123 import 'package:file_picker/file_picker.dart';
124

```

```

125 class FileListScreen extends StatefulWidget {
126     final String server;
127     final String login;
128     final String password;
129
130     FileListScreen(
131         {required this.server, required this.login, required this.password
132         });
133
134     @override
135     _FileListScreenState createState() =>
136         _FileListScreenState(server, login, password);
137 }
138
139 class FTPManager {
140     late String _server;
141     late String _login;
142     late String _password;
143     late FTPConnect _ftpConnect;
144
145     List _fileList = [];
146     String _currentDir = "";
147     // List<String> _currentPath = [];
148
149     List get fileList => _fileList;
150     // late List<String> consoleLogs = [];
151
152     int maxLogCount = 4;
153     late List<String> consoleLogs = List<String>.generate(
154         maxLogCount, (index) => "");
155
156     FTPManager(String server, String login, String password) {
157         _server = server;
158         _login = login;
159         _password = password;
160         _ftpConnect =
161             FTPConnect(_server, user: _login, pass: _password, showLog: true
162             );
163     }
164
165     Function()? _onUpdate;
166
167     void _log(String message) {
168         final DateTime now = DateTime.now();
169         final String formattedTime = "[${now}] ";
170         final formattedMessage = formattedTime + "> " + message;

```

```

169     consoleLogs.insert(0, formattedMessage);
170     if (consoleLogs.length > maxLogCount) {
171         consoleLogs.removeLast();
172     }
173     _onUpdate!();
174 }
175
176 void setOnUpdateCallback(Function() onUpdate) {
177     _onUpdate = onUpdate;
178 }
179
180 Future<void> connect() async {
181     try {
182         await _ftpConnect.connect();
183         _log("Connected to " + _server);
184     } catch (error) {
185         _log("FTP connection error: " + error.toString());
186     }
187 }
188
189 Future<void> disconnect() async {
190     await _ftpConnect.disconnect();
191     _fileList.clear();
192     _currentDir = '';
193     _log("Disconnected from " + _server);
194 }
195
196 Future<void> refreshFTPData(String currentDir) async {
197     // _currentPath.add(currentDir);
198     print("Refreshing");
199     // print(_currentPath.join("/"));
200     await _ftpConnect.changeDirectory(currentDir);
201     List dirList = await _ftpConnect.listDirectoryContent();
202     _fileList = dirList;
203     _currentDir = currentDir;
204     // for (var file in dirList) {
205     //     print('File name: ${file.name}' + 'File type ${file.type}');
206     // }
207     if (currentDir == "") {
208         _log("Refreshing in current directory");
209     } else {
210         _log("Refreshing in directory: " + currentDir);
211     }
212 }
213 }
214

```

```

215 void _openFileExplorer() async {
216     FilePickerResult? result = await FilePicker.platform.pickFiles();
217
218     if (result != null) {
219         File file = File(result.files.single.path!);
220         await uploadFileToFTP(file);
221     } else {
222
223     }
224 }
225
226 Future<void> uploadFileToFTP(File fileName) async {
227     try {
228         await _ftpConnect.uploadFile(fileName);
229         _log("File ${fileName} has been successfully uploaded to FTP");
230         refreshFTPData('');
231     } catch (error) {
232         _log("FTP upload error: ${error.toString()}");
233     }
234 }
235
236 Future<void> deleteFileFromFTP(String fileName) async {
237     // implementation for deleting file from FTP
238     try {
239         await _ftpConnect.deleteFile(fileName);
240         _log("File: " + fileName + "deleted successfully");
241         refreshFTPData('');
242     } catch (error) {
243         _log("FTP deleting error: " + error.toString());
244     }
245 }
246
247 Future<void> downloadFileFromFTP(String fileName) async {
248
249     try {
250         File localFilePath = File("/home/user1/" + fileName);
251         _log("Local path: " + localFilePath.toString());
252         await _ftpConnect.downloadFile(fileName, localFilePath);
253         _log("File: " + fileName + " downloaded successfully");
254     } catch (error) {
255         _log("FTP download error: " + error.toString());
256     }
257 }
258
259 Future<void> createFTPDirectory(String currentDir) async {
260     // implementation for creating directory on FTP

```



```

261
262     try {
263         await _ftpConnect.makeDirectory(currentDir);
264         _log("New directory created successfully: " + currentDir);
265         refreshFTPData('');
266     } catch (error) {
267         _log("FTP creating directory error: " + error.toString());
268     }
269 }
270 }
271
272 class _FileListScreenState extends State<FileListScreen> {
273     late String _server;
274     late String _login;
275     late String _password;
276     late FTPManager _ftpManager;
277
278     _FileListScreenState(String server, String login, String password) {
279         _server = server;
280         _login = login;
281         _password = password;
282         _ftpManager = FTPManager(_server, _login, _password);
283         _ftpManager.setOnUpdateCallback(_onUpdate);
284     }
285
286     _onUpdate() {
287         setState(() {});
288     }
289
290     @override
291     void initState() {
292         super.initState();
293         _ftpManager.connect().then((_) {
294             _ftpManager.refreshFTPData('').then((_) {
295                 _onUpdate();
296             });
297         });
298     }
299
300     @override
301     Widget build(BuildContext context) {
302         return Scaffold(
303             appBar: AppBar(
304                 title: Text('File List Screen'),
305             ),
306             body: Column(

```

```

307     children: [
308         // Align(
309         //     alignment: Alignment.centerLeft,
310         //     child: Text('Current Path: ' + _ftpManager._currentDir),
311         // ),
312
313         Expanded(
314             child: ListView.builder(
315                 itemCount: _ftpManager.fileList.length,
316
317                 itemBuilder: (context, index) {
318                     final file = _ftpManager.fileList[index];
319                     final isDirectory = file.type.toString() == "
FTPEntryType.DIR";
320                     final icon =
321                         isDirectory ? Icons.folder : Icons.insert_drive_file
;
322                     final color = isDirectory ? Colors.orange : Colors.grey;
323
324                     return ListTile(
325                         leading: Icon(icon, color: color),
326                         title: Text(file.name),
327                         onTap: () async {
328                             final currentFile = _ftpManager.fileList[index].name
;
329                             // final currentPath = _ftpManager._currentPath.join
330                             ("/");
331                             // final path = currentPath.isNotEmpty ? "
$currentPath/$currentFile" : currentFile;
332                             await _ftpManager.refreshFTPData(currentFile);
333                             setState(
334                                 () {});
335                             },
336                             onLongPress: () {
337
338                                 showMenu(
339                                     context: context,
340                                     position: RelativeRect.fromLTRB(0, 0, 0, 0),
341                                     items: [
342                                         const PopupMenuItem(
343                                             value: "download",
344                                             child: Text("Download file"),
345                                         ),
346                                         const PopupMenuItem(
347                                             value: "delete",
348                                             child: Text("Delete file"),

```

```

348         ),
349     ],
350 ).then((value) {
351     if (value == "download") {
352         // if (_ftpManager.fileList[index].type == "
FTPEntryType.FILE") {
353             final currentFile = _ftpManager.fileList[index].
name;
354             _ftpManager.downloadFileFromFTP(currentFile);
355             // }
356         } else if (value == "delete") {
357
358             _ftpManager.deleteFileFromFTP(_ftpManager.
fileList[index].name);
359         }
360     });
361 },
362 );
363 },
364 ),
365 ),
366 Container(
367     height: MediaQuery.of(context).size.height /
368         6,
369     padding: EdgeInsets.all(8.0),
370     decoration: BoxDecoration(
371         border: Border.all(),
372     ),
373     child: ListView.builder(
374         itemCount: _ftpManager.consoleLogs.length,
375         itemBuilder: (context, index) {
376             return Text(_ftpManager.consoleLogs[index]);
377         },
378     ),
379 ),
380 Row(
381     mainAxisAlignment: MainAxisAlignment.start,
382     children: [
383         ElevatedButton(
384             onPressed: () async {
385                 await _ftpManager.disconnect();
386                 setState(() {});
387             },
388             child: Text('Disconnect'),
389         ),
390         SizedBox(width: 10),

```

```

391
392
393 ElevatedButton(
394     onPressed: () {
395         showDialog(
396             context: context,
397             builder: (BuildContext context) {
398                 String newDir = "";
399                 return AlertDialog(
400                     title: Text('Create a new folder '),
401                     content: TextField(
402                         onChanged: (value) {
403                             newDir = value;
404                         },
405                         decoration: InputDecoration(hintText: "Enter
the folder name"),
406                     ),
407                     actions: <Widget>[
408                         ElevatedButton(
409                             onPressed: () {
410                                 Navigator.of(context).pop();
411                             },
412                             child: Text('Cancel '),
413                         ),
414                         ElevatedButton(
415                             onPressed: () {
416                                 _ftpManager.createFTPDiretory(newDir);
417                                 Navigator.of(context).pop();
418                             },
419                             child: Text('Create '),
420                         ),
421                     ],
422                 );
423             },
424         );
425     },
426     child: Text('Create diretory '),
427 ),
428 SizedBox(width: 10),
429 ElevatedButton(
430     onPressed: () {
431         _ftpManager._openFileExplorer();
432     },
433     child: Text('Upload to FTP'),
434 ),
435 SizedBox(width: 10),

```

```

436
437         ],
438     ),
439     ],
440 ),
441 );
442 }
443 }
444
445 import 'package:flutter/material.dart';
446 import 'ftp.dart';
447 import 'ssh.dart';
448 import 'smtp.dart';
449
450 void main() {
451     runApp(MyApp());
452 }
453
454 class MyApp extends StatelessWidget {
455     @override
456     Widget build(BuildContext context) {
457         return MaterialApp(
458             home: HomeScreen(),
459         );
460     }
461 }
462
463 class HomeScreen extends StatelessWidget {
464     @override
465     Widget build(BuildContext context) {
466         return Scaffold(
467             appBar: AppBar(
468                 title: Text('Main Screen'),
469             ),
470             body: Center(
471                 child: Column(
472                     mainAxisAlignment: MainAxisAlignment.center,
473                     children: <Widget>[
474                         ElevatedButton(
475                             onPressed: () {
476                                 Navigator.push(
477                                     context,
478                                     MaterialPageRoute(builder: (context) => FTPScreen()),
479                                 );
480                             },
481                             child: Text('FTP'),

```

```

482         ),
483         SizedBox(height: 10),
484         ElevatedButton(
485             onPressed: () {
486                 Navigator.push(
487                     context,
488                     MaterialPageRoute(builder: (context) => SMTPScreen()),
489                 );
490             },
491             child: Text('SMTP'),
492         ),
493         SizedBox(height: 10),
494         ElevatedButton(
495             onPressed: () {
496                 Navigator.push(
497                     context,
498                     MaterialPageRoute(builder: (context) => SSHScreen()),
499                 );
500             },
501             child: Text('SSH'),
502         ),
503     ],
504 ),
505 ),
506 );
507 }
508 }
509
510 import 'package:flutter/material.dart';
511 import 'package:shared_preferences/shared_preferences.dart';
512 import 'smtp_commands_screen.dart';
513 import 'dart:io';
514
515 class MyApp extends StatelessWidget {
516     @override
517     Widget build(BuildContext context) {
518         return MaterialApp(
519             home: SMTPScreen(),
520         );
521     }
522 }
523
524 class SMTPScreen extends StatefulWidget {
525     @override
526     _SMTPScreenState createState() => _SMTPScreenState();
527 }

```

```

528
529 class _SMTPScreenState extends State<SMTPScreen> {
530     late TextEditingController _smtpHostController = TextEditingController
531         ();
532     late TextEditingController _smtpUsernameController =
533         TextEditingController();
534     late TextEditingController _smtpKeyController = TextEditingController
535         ();
536     late TextEditingController _smtpPortController = TextEditingController
537         ();
538
539     late SharedPreferences _prefs;
540
541     @override
542     void initState() {
543         super.initState();
544         _init();
545     }
546
547     void _init() async {
548         _prefs = await SharedPreferences.getInstance();
549         _initTextControllers();
550     }
551
552     void _initTextControllers() {
553         _smtpHostController.text = _prefs.getString('smtpHost') ?? '';
554         _smtpUsernameController.text = _prefs.getString('smtpUsername') ??
555             '';
556         _smtpKeyController.text = _prefs.getString('smtpKey') ?? '';
557         _smtpPortController.text = _prefs.getString('smtpPort') ?? '';
558     }
559
560     void _saveSmtpSettings() {
561         _prefs.setString('smtpHost', _smtpHostController.text);
562         _prefs.setString('smtpUsername', _smtpUsernameController.text);
563         _prefs.setString('smtpKey', _smtpKeyController.text);
564         _prefs.setString('smtpPort', _smtpPortController.text);
565         ScaffoldMessenger.of(context).showSnackBar(
566             SnackBar(content: Text('SMTP settings saved')),
567         );
568     }
569
570     void _navigateToEmailComposeScreen() {
571         Smtpparams params = Smtpparams(
572             host: _smtpHostController.text,
573             username: _smtpUsernameController.text,

```

```

569         key: _smtpKeyController.text,
570         port: _smtpPortController.text,
571     );
572     Navigator.push(
573         context,
574         MaterialPageRoute(
575             builder: (context) => EmailComposeScreen(params: params),
576         ),
577     );
578 }
579
580 @override
581 Widget build(BuildContext context) {
582     return Scaffold(
583         appBar: AppBar(
584             title: Text('SMTP Data Entry Form'),
585         ),
586         body: Padding(
587             padding: EdgeInsets.all(16.0),
588             child: Column(
589                 mainAxisAlignment: MainAxisAlignment.center,
590                 children: <Widget>[
591                     TextField(
592                         controller: _smtpHostController,
593                         decoration: InputDecoration(labelText: 'SMTP Server'),
594                     ),
595                     TextField(
596                         controller: _smtpUsernameController,
597                         decoration: InputDecoration(labelText: 'SMTP Login'),
598                     ),
599                     TextField(
600                         controller: _smtpKeyController,
601                         decoration: InputDecoration(labelText: 'SMTP Password'),
602                     ),
603                     TextField(
604                         controller: _smtpPortController,
605                         decoration: InputDecoration(labelText: 'SMTP Port'),
606                     ),
607                     SizedBox(height: 10),
608                     ElevatedButton(
609                         onPressed: _saveSmtplibSettings,
610                         child: Text('Save'),
611                     ),
612                     SizedBox(height: 10),
613                     ElevatedButton(
614                         onPressed: _navigateToEmailComposeScreen,

```



```

615         child: Text('Create an email'),
616     ),
617 ],
618 ),
619 ),
620 );
621 }
622 }
623
624 import "package:flutter/material.dart";
625 import "package:mailer/mailer.dart";
626 import "package:mailer/smtp_server.dart";
627
628 class SmtParams {
629     final String host;
630     final String username;
631     final String key;
632     final String port;
633
634     SmtParams({
635         required this.host,
636         required this.username,
637         required this.key,
638         required this.port,
639     });
640 }
641
642 class EmailComposeScreen extends StatefulWidget {
643     final SmtParams params;
644     EmailComposeScreen({required this.params});
645
646     @override
647     _EmailComposeScreenState createState() => _EmailComposeScreenState();
648 }
649
650 class _EmailComposeScreenState extends State<EmailComposeScreen> {
651     final TextEditingController emailController = TextEditingController();
652     final TextEditingController subjectController = TextEditingController();
653     final TextEditingController messageController = TextEditingController();
654
655     String? emailResponse;
656
657     @override
658     Widget build(BuildContext context) {

```

```

659     return Scaffold(
660       appBar: AppBar(title: Text("ComposeEmail")),
661       body: Padding(
662         padding: EdgeInsets.all(16.0),
663         child: Column(
664           children: [
665             TextField(
666               controller: emailController,
667               decoration: InputDecoration(labelText: "EmailAddress"),
668             ),
669             TextField(
670               controller: subjectController,
671               decoration: InputDecoration(labelText: "Subject"),
672             ),
673             TextField(
674               controller: messageController,
675               decoration: InputDecoration(labelText: "Message"),
676             ),
677             ElevatedButton(
678               onPressed: () async {
679                 String email = emailController.text;
680                 String subject = subjectController.text;
681                 String message = messageController.text;
682                 String username = widget.params.username;
683                 String password = widget.params.key;
684                 final smtpServer = Smtplib.Smtplib(
685                   widget.params.host,
686                   port: int.parse(widget.params.port),
687                   ssl: true,
688                   username: username,
689                   password: password,
690                 );
691                 final emailMessage = Message()
692                   ..from = Address(username, "DanilaPalych")
693                   ..recipients.add(email)
694                   ..subject = subject
695                   ..text = message;
696
697                 try {
698                   final sendReport = await smtpServer
699                     .send(emailMessage);
700                   setState(() {
701                     emailResponse = "Message sent: " + sendReport.
702                       toString();
703                   });
704                 } catch (e) {

```

```

703         setState(() {
704             emailResponse = "Errorsendingmessage:$e";
705         });
706     }
707 },
708     child: Text("SendEmail"),
709 ),
710     SizedBox(height: 16.0),
711     Text(emailResponse ?? ""),
712 ],
713 ),
714 ),
715 );
716 }
717 }
718
719 import 'package:flutter/material.dart';
720 import 'package:shared_preferences/shared_preferences.dart';
721 import 'ssh_commands_screen.dart';
722 import 'dart:io';
723
724 class MyApp extends StatelessWidget {
725     @override
726     Widget build(BuildContext context) {
727         return MaterialApp(
728             home: SSHScreen(),
729         );
730     }
731 }
732
733 class SSHScreen extends StatefulWidget {
734     @override
735     _SSHScreenState createState() => _SSHScreenState();
736 }
737
738 class _SSHScreenState extends State<SSHScreen> {
739     late TextEditingController _sshHostController = TextEditingController();
740     late TextEditingController _sshUsernameController =
741         TextEditingController();
741     late TextEditingController _sshKeyController = TextEditingController();
742     late TextEditingController _sshPortController = TextEditingController();
743
744     late SharedPreferences _prefs;

```

```

745
746 @override
747 void initState() {
748     super.initState();
749     _init();
750 }
751
752 void _init() async {
753     _prefs = await SharedPreferences.getInstance();
754     _initTextControllers();
755 }
756
757 void _initTextControllers() {
758     _sshHostController.text = _prefs.getString('sshHost') ?? '';
759     _sshUsernameController.text = _prefs.getString('sshUsername') ?? '';
760     _sshKeyController.text = _prefs.getString('sshPassword') ?? '';
761     _sshPortController.text = _prefs.getString('sshPort') ?? '';
762 }
763
764 void _saveSshSettings() {
765     _prefs.setString('sshHost', _sshHostController.text);
766     _prefs.setString('sshUsername', _sshUsernameController.text);
767     _prefs.setString('sshPassword', _sshKeyController.text);
768     _prefs.setString('sshPort', _sshPortController.text);
769     ScaffoldMessenger.of(context).showSnackBar(
770         SnackBar(content: Text('SSH settings saved')),
771     );
772 }
773
774 void _navigateToSSHCommandsScreen() {
775     String host = _sshHostController.text;
776     String username = _sshUsernameController.text;
777     String password = _sshKeyController.text;
778     String port = _sshPortController.text;
779
780     Navigator.push(
781         context,
782         MaterialPageRoute(
783             builder: (context) => SSHCommandsScreen(
784                 host: host,
785                 username: username,
786                 password: password,
787                 port: port,
788             ),
789         ),
790     );

```

```

791 }
792
793 @override
794 Widget build(BuildContext context) {
795   return Scaffold(
796     appBar: AppBar(
797       title: Text('SSH Data Entry Form'),
798     ),
799     body: Padding(
800       padding: EdgeInsets.all(16.0),
801       child: Column(
802         mainAxisAlignment: MainAxisAlignment.center,
803         children: <Widget>[
804           TextField(
805             controller: _sshHostController,
806             decoration: InputDecoration(labelText: 'SSH Server'),
807           ),
808           TextField(
809             controller: _sshUsernameController,
810             decoration: InputDecoration(labelText: 'SSH Login'),
811           ),
812           TextField(
813             controller: _sshKeyController,
814             decoration: InputDecoration(labelText: 'SSH Password'),
815           ),
816           TextField(
817             controller: _sshPortController,
818             decoration: InputDecoration(labelText: 'SSH Port'),
819           ),
820           SizedBox(height: 10),
821           ElevatedButton(
822             onPressed: _saveSshSettings,
823             child: Text('Save'),
824           ),
825           SizedBox(height: 10),
826           ElevatedButton(
827             onPressed: _navigateToSSHCommandsScreen,
828             child: Text('Connect to SSH server'),
829           ),
830         ],
831       ),
832     ),
833   );
834 }
835 }
836

```

```

837 import 'dart:convert';
838 import 'dart:io';
839
840 import 'package:flutter/material.dart';
841 import 'package:dartssh2/dartssh2.dart';
842
843 class SSHCommandsScreen extends StatefulWidget {
844   final String host;
845   final String username;
846   final String password;
847   final String port;
848
849   SSHCommandsScreen({
850     required this.host,
851     required this.username,
852     required this.password,
853     required this.port,
854   });
855
856   @override
857   _SSHCommandsScreenState createState() => _SSHCommandsScreenState(host,
858     username, password, port);
859 }
860
861
862
863 class SSHManager {
864   late String _server;
865   late String _login;
866   late String _password;
867   late String _port;
868   late SSHClient client;
869
870   List _fileList = [];
871   late List<String> consoleLogs = [];
872   int maxLogCount = 100;
873
874   SSHManager(String server, String login, String password, String port)
875     {
876     _server = server;
877     _login = login;
878     _password = password;
879     _port = port;
880   }

```

```

881 void _log(String message) {
882     final DateTime now = DateTime.now();
883     final String formattedTime = "[${now}] ";
884     final formattedMessage = formattedTime + "> " + message;
885     consoleLogs.insert(0, formattedMessage);
886     if (consoleLogs.length > maxLogCount) {
887         consoleLogs.removeLast();
888     }
889 }
890
891 Future<void> connect() async {
892     try {
893         final socket = await SSHSocket.connect(_server, int.parse(_port));
894         client = SSHClient(
895             socket,
896             username: _login,
897             onPasswordRequest: () => _password,
898         );
899
900         _log("Connected to " + _server);
901
902         final response = await client.run('uname');
903         _log(utf8.decode(response));
904     } catch (error) {
905         _log("SSH connection error: " + error.toString());
906     }
907 }
908
909
910 Future<void> disconnect() async {
911     try {
912         client.close();
913         await client.done;
914         _log("Disconnected from " + _server);
915     } catch (error) {
916         _log("Error while disconnecting: " + error.toString());
917     }
918 }
919
920
921 Future<void> command(String cmd) async {
922     try {
923         final response = await client.run(cmd);
924         _log(utf8.decode(response));
925
926         _log("Command " + cmd + " executed successfully");

```

```

927     } catch (error) {
928         _log("Command execution error: " + error.toString());
929     }
930 }
931
932 }
933
934 class _SSHCommandsScreenState extends State<SSHCommandsScreen> {
935     late SSHManager _sshManager;
936
937     _SSHCommandsScreenState(String server, String login, String password,
938         String port) {
939         _sshManager = SSHManager(server, login, password, port);
940     }
941
942     @override
943     void initState() {
944         super.initState();
945         _sshManager.connect().then((_) {
946             setState(() {});
947         });
948     }
949
950     void _executeCommand(String command) {
951         _sshManager.command(command).then((_) {
952             setState(() {});
953         });
954     }
955
956     @override
957     Widget build(BuildContext context) {
958
959         String cmd = '';
960
961         return Scaffold(
962             appBar: AppBar(
963                 title: Text('SSH Commands Screen'),
964             ),
965             body: Column(
966                 children: [
967
968                 TextField(
969                     onChanged: (value) {
970                         cmd = value;
971                     },
972                     decoration: InputDecoration(

```



```

972         hintText: 'Enter command',
973     ),
974 ),
975
976 Row(
977     mainAxisAlignment: MainAxisAlignment.spaceEvenly,
978     children: [
979         ElevatedButton(
980             onPressed: () {
981                 _executeCommand(cmd);
982             },
983             child: Text('Execute'),
984         ),
985         SizedBox(width: 10),
986         ElevatedButton(
987             onPressed: () {
988                 _sshManager.disconnect();
989             },
990             child: Text('Disconnect'),
991         ),
992     ],
993 ),
994
995
996 Expanded(
997     child: Container(
998         padding: EdgeInsets.all(8.0),
999         // decoration: BoxDecoration(
1000         //     border: Border.all(),
1001         // ),
1002         child: ListView.builder(
1003             itemCount: _sshManager.consoleLogs.length,
1004             itemBuilder: (context, index) {
1005                 return Text(_sshManager.consoleLogs[index]);
1006             },
1007         ),
1008     ),
1009 ),
1010
1011 ],
1012 ),
1013 );
1014 }
1015 }

```

3 Результаты

Результаты работы программы представлены на рисунках 1 – 10.

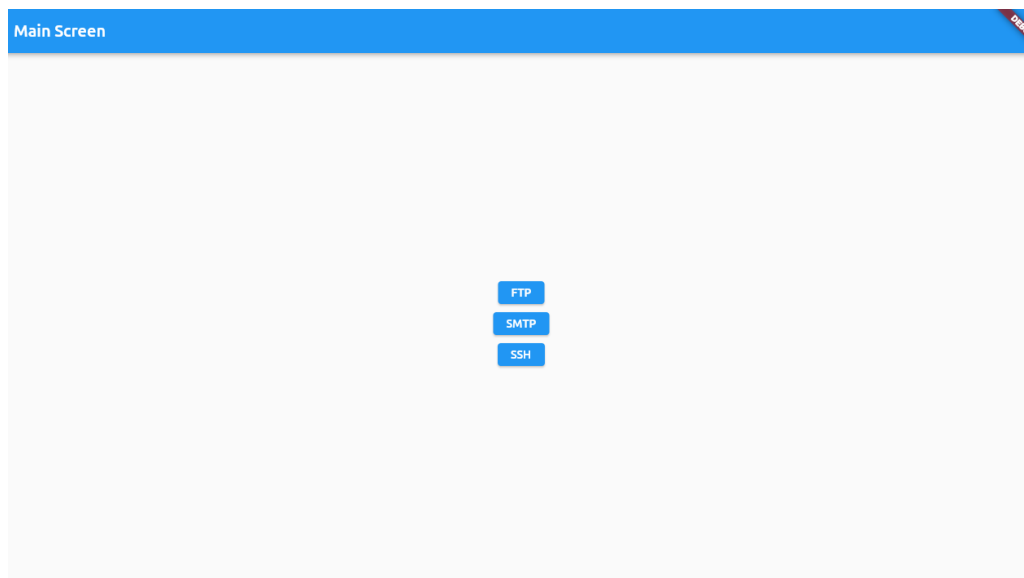


Рис. 1 — Результаты

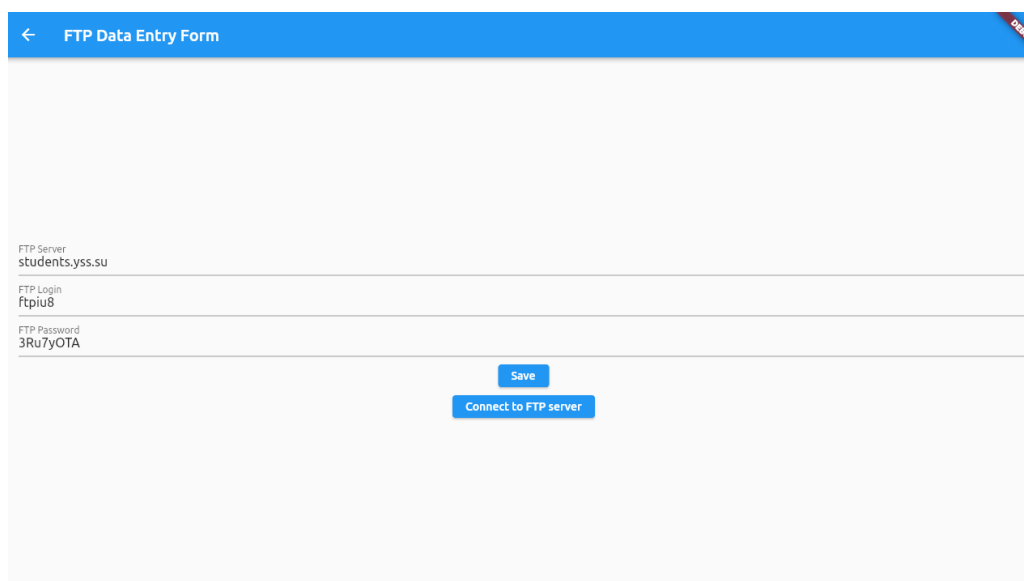


Рис. 2 — Результаты

4 Выводы

В результате выполнения лабораторной работы было реализовано мобильное приложение выполняющее функцию FTP-клиента, SSH-клиента и SMTP-клиента.

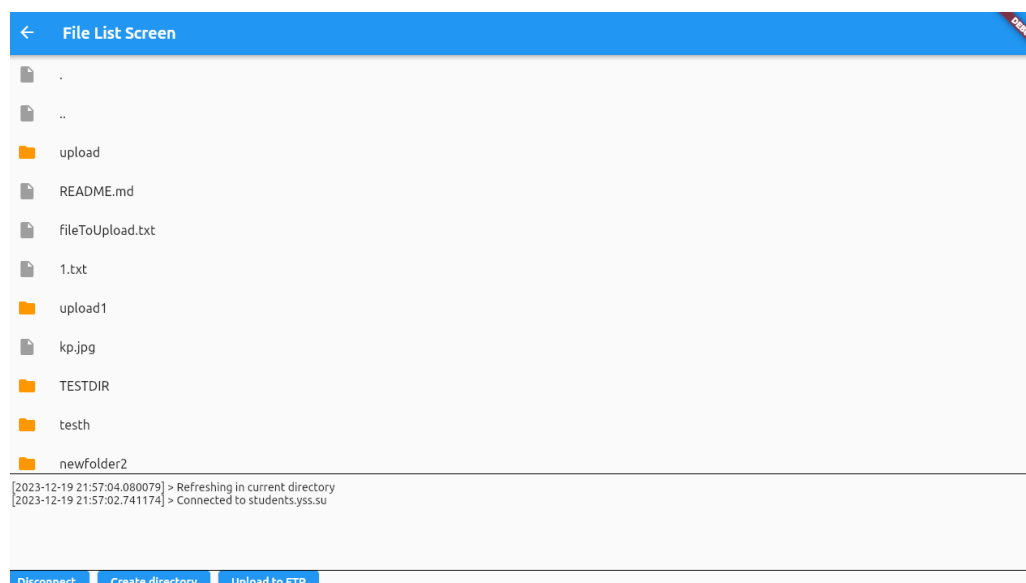


Рис. 3 — Результаты

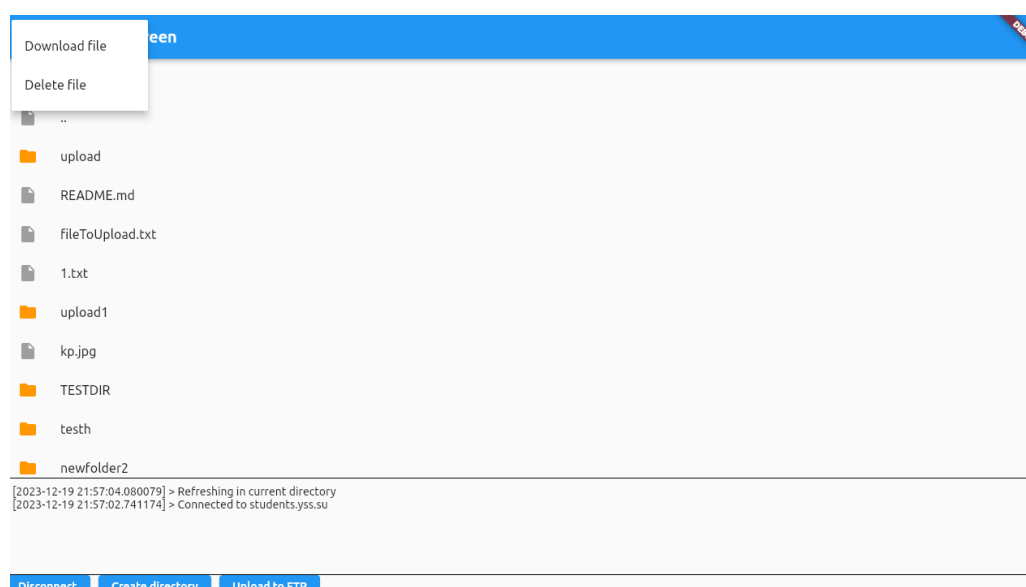


Рис. 4 — Результаты

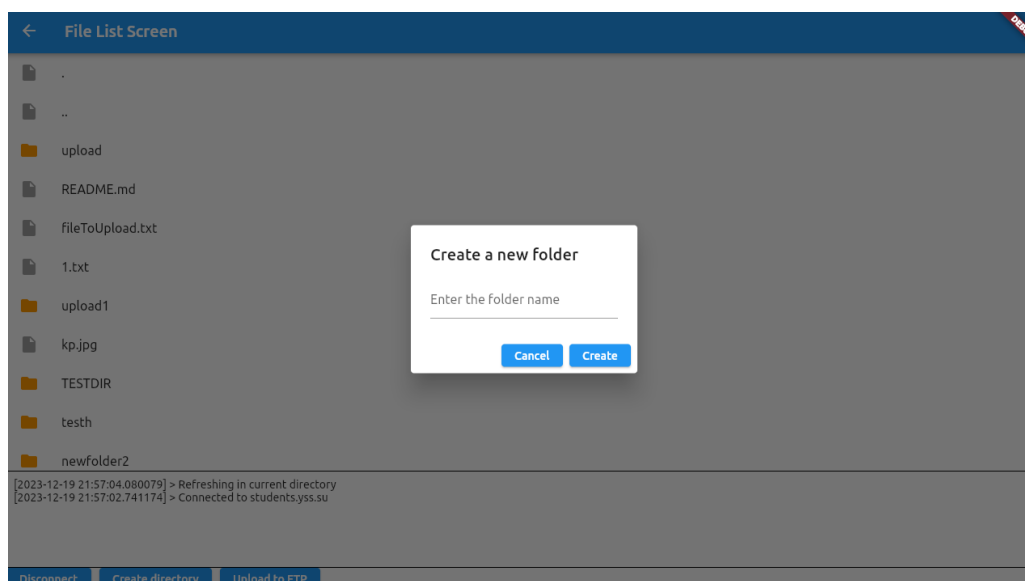
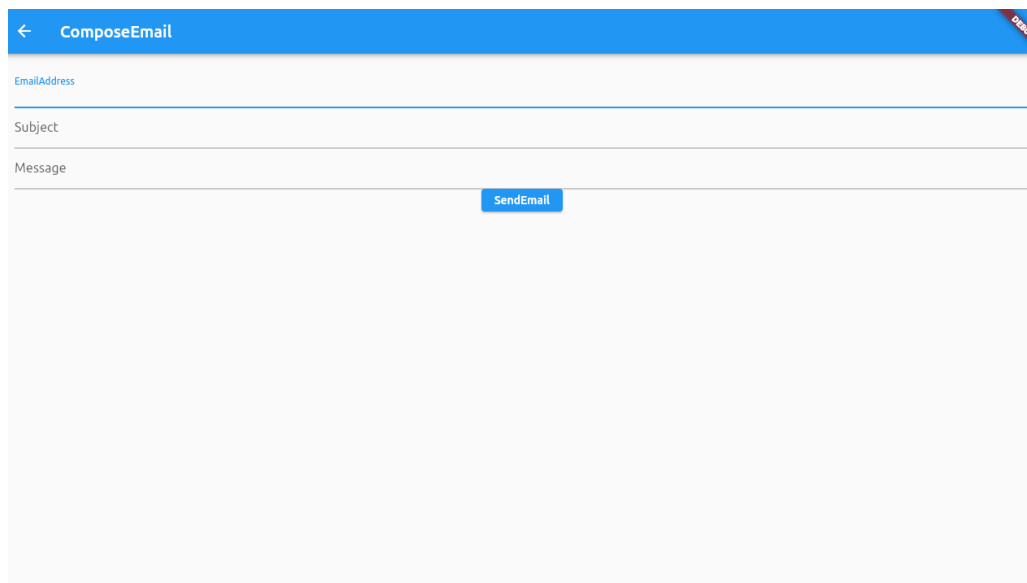


Рис. 5 — Результаты

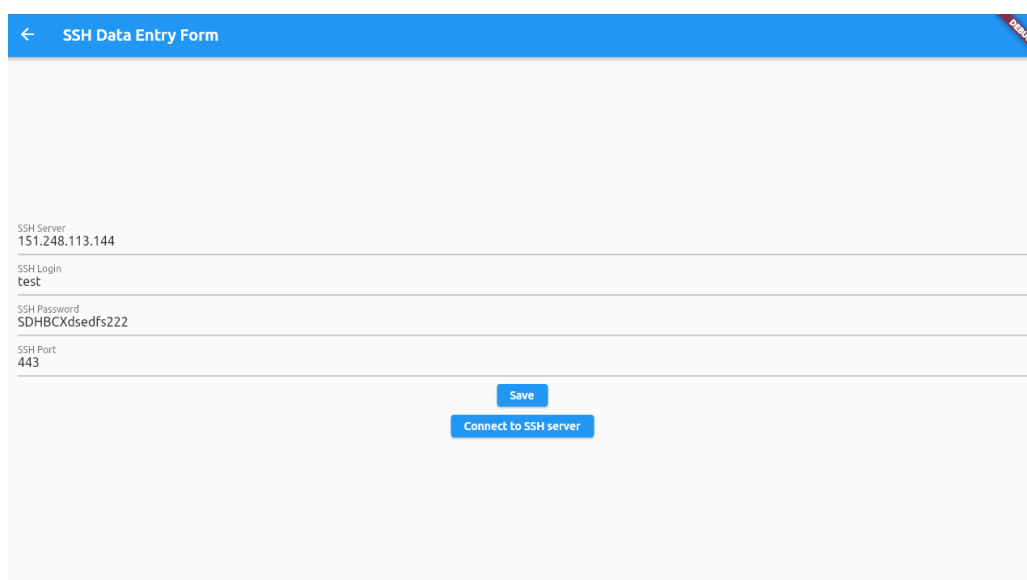
A screenshot of a web application interface titled "SMTP Data Entry Form". The form contains four input fields with the following labels and values: "SMTP Server" with "smtp.mail.ru", "SMTP Login" with "danila@bmstu.posevin.ru", "SMTP Password" with "W6bG7z6xc5Q4AsiBWMmk", and "SMTP Port" with "465". Below the input fields are two buttons: "Save" and "Create an email".

Рис. 6 — Результаты



The screenshot shows a mobile application interface for composing an email. At the top is a blue header bar with a back arrow and the text "ComposeEmail". Below the header are three input fields: "EmailAddress", "Subject", and "Message". A blue button labeled "SendEmail" is positioned in the center of the "Message" field. A red "demo" label is visible in the top right corner of the header bar.

Рис. 7 — Результаты



The screenshot shows a mobile application interface for entering SSH data. At the top is a blue header bar with a back arrow and the text "SSH Data Entry Form". Below the header are four input fields: "SSH Server" (containing "151.248.113.144"), "SSH Login" (containing "test"), "SSH Password" (containing "SDHBCXdsedfs222"), and "SSH Port" (containing "443"). Below the fields are two blue buttons: "Save" and "Connect to SSH server". A red "demo" label is visible in the top right corner of the header bar.

Рис. 8 — Результаты

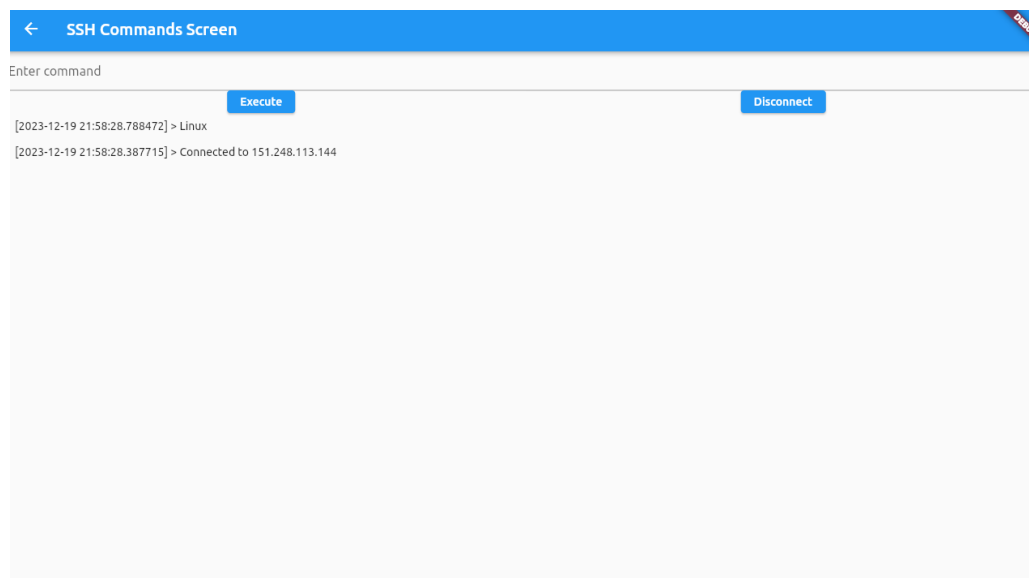


Рис. 9 — Результаты

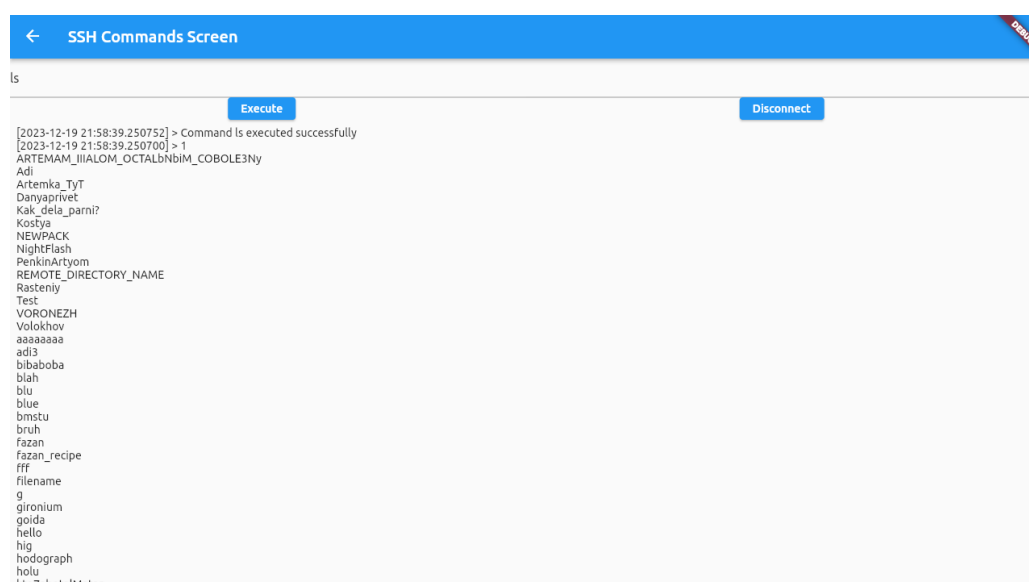


Рис. 10 — Результаты