Лабораторная работа № 12

Реализация задач лекции №12 по вариантам

Вариант 1

Реализовать мобильное приложение выполняющее функцию ftp-клиента. В приложении должен быть следующий функционал:

1. Виджет с параметрами ftp-подключения, реализованной 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.
2. На отдельном виджете 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.
3. Интерфейс переключения между виджетами хранения данных подключения и интерфейса работы с сервером должен быть взят из лекции 9, номер примера 24.

```
import 'dart:async';
import 'dart:io';

import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:ftpconnect/ftpConnect.dart';
import 'package:path_provider/path_provider.dart';

enum Sky { form, ftpClient }

String ip = "ENTER IP";
String login = "ENTER LOGIN";
String password = "ENTER PASSWORD";

// String ip = "students.yss.su";
// String login = "ftpiu8";
// String password = "3Ru7yOTA";

Map<Sky, Color> skyColors = <Sky, Color>{
  Sky.form: const Color(0xff40826d),
  Sky.ftpClient: const Color(0xff191970),
};
```

```
};

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return const CupertinoApp(
      localizationsDelegates: <LocalizationsDelegate<dynamic>>[
        GlobalMaterialLocalizations.delegate,
        GlobalWidgetsLocalizations.delegate,
        GlobalCupertinoLocalizations.delegate,
      ],
      title: 'Flutter Demo',
      theme: CupertinoThemeData(brightness: Brightness.light),
      home: SegmentedControlApp(),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return const CupertinoApp(
      theme: CupertinoThemeData(brightness: Brightness.light),
      home: SegmentedControlExample(),
    );
  }
}

class SegmentedControlExample extends StatefulWidget {
  const SegmentedControlExample({super.key});

  @override
  State<SegmentedControlExample> createState() =>
      _SegmentedControlExampleState();
}

class _SegmentedControlExampleState extends State<SegmentedControlExample> {
  Sky _selectedSegment = Sky.form;

  @override
  Widget build(BuildContext context) {
    return CupertinoPageScaffold(
        backgroundColor: skyColors[_selectedSegment],
        navigationBar: CupertinoNavigationBar(
          // This Cupertino segmented control has the enum "Sky" as the type.
          middle: CupertinoSlidingSegmentedControl<Sky>(
            backgroundColor: CupertinoColors.systemGrey2,
            thumbColor: skyColors[_selectedSegment]!,
            // This represents the currently selected segmented control.
            groupValue: _selectedSegment,
            // Callback that sets the selected segmented control.
            onValueChanged: (Sky? value) {
```

```
            onValueChanged: (Sky? value) {
              if (value != null) {
                setState(() {
                  _selectedSegment = value;
                });
              }
            },
            children: const <Sky, Widget>{
              Sky.form: Padding(
                padding: EdgeInsets.symmetric(horizontal: 20),
                child: Text(
                  'Form',
                  style: TextStyle(color: CupertinoColors.white),
                ),
              ),
              Sky.ftpClient: Padding(
                padding: EdgeInsets.symmetric(horizontal: 20),
                child: Text(
                  'FTP',
                  style: TextStyle(color: CupertinoColors.white),
                ),
              ),
            },
          ),
        ),
        child: Center(
          child: _selectedSegment.name == "form" ? MyHomePage() : FtpPage(),
        ));
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  TextEditingController controllerIp = TextEditingController();
  TextEditingController controllerPort = TextEditingController();
  TextEditingController controllerLogin = TextEditingController();
  TextEditingController controllerPassword = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return CupertinoPageScaffold(
      child: Center(
        // child: Text("HUI", style: TextStyle(color: Colors.black),),)
        child: Container(
          padding: EdgeInsets.all(20.0),
          child: Column(
            children: <Widget>[
              SizedBox(
                height: 50,
              ),
              CupertinoFormSection.insetGrouped(
                header: const Text('Connection sttings'),
```

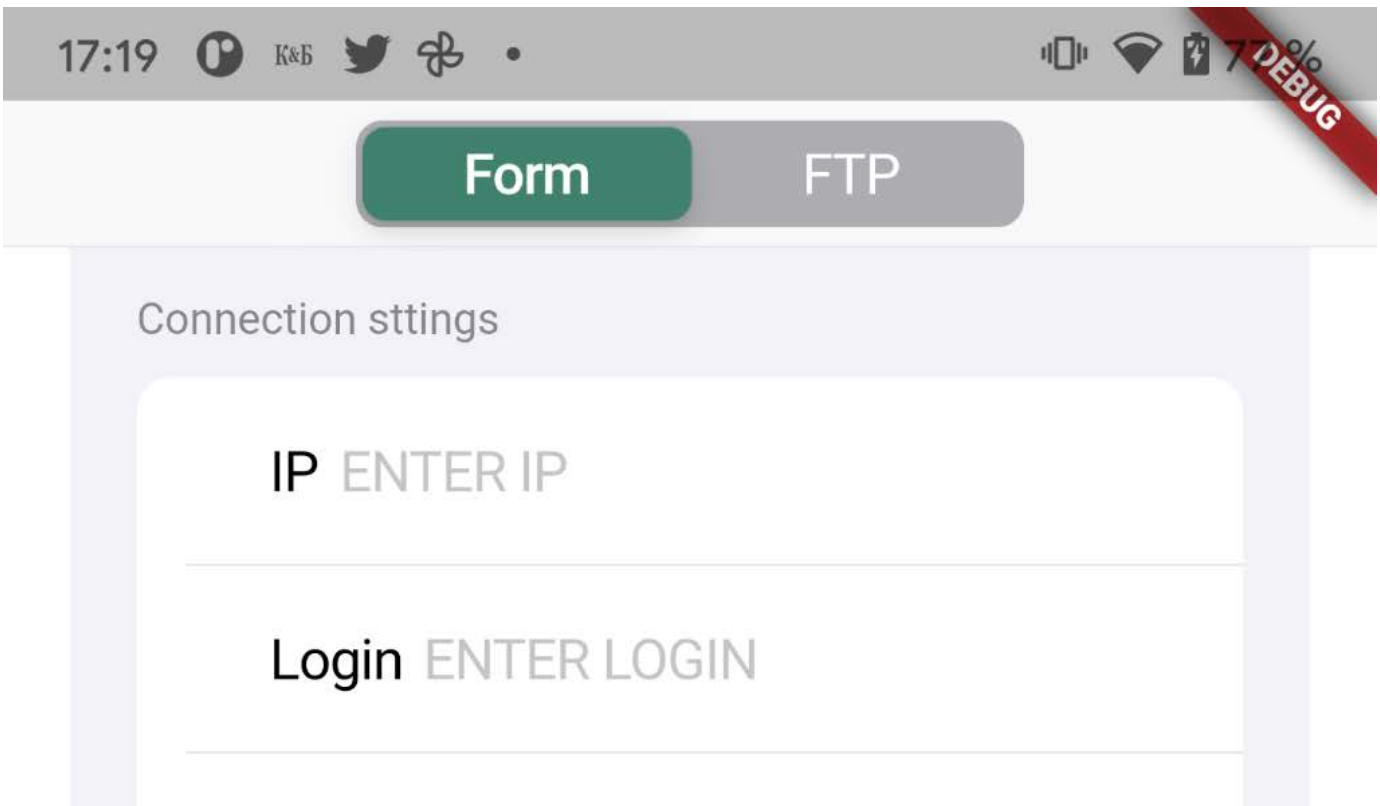
```
                children: [
                  CupertinoFormRow(
                      child: CupertinoTextFormFieldRow(
                    prefix: const Text('IP'),
                    placeholder: ip,
                    controller: controllerIp,
                  )),
                  CupertinoFormRow(
                      child: CupertinoTextFormFieldRow(
                    prefix: const Text('Login'),
                    placeholder: login,
                    controller: controllerLogin,
                  )),
                  CupertinoFormRow(
                      child: CupertinoTextFormFieldRow(
                    prefix: const Text('Password'),
                    placeholder: password,
                    controller: controllerPassword,
                  )),
                ],
              ),
              CupertinoButton.filled(
                onPressed: () => {
                  setState(() {
                    ip = controllerIp.text;
                    controllerIp.text = "";
                    debugPrint(ip);

                    login = controllerLogin.text;
                    controllerLogin.text = "";
                    debugPrint(login);

                    password = controllerPassword.text;
                    controllerPassword.text = "";
                    debugPrint(password);
                  })
                },
                child: const Icon(CupertinoIcons.add),
              ),
            ],
          ),
        ),
      ),
    );
  }
}

class FtpPage extends StatefulWidget {
  @override
  _FtpPageState createState() => _FtpPageState();
}

class _FtpPageState extends State<FtpPage> {
  TextEditingController controller = TextEditingController();
  final List<String> list = ["Hello", "lets go"];
```

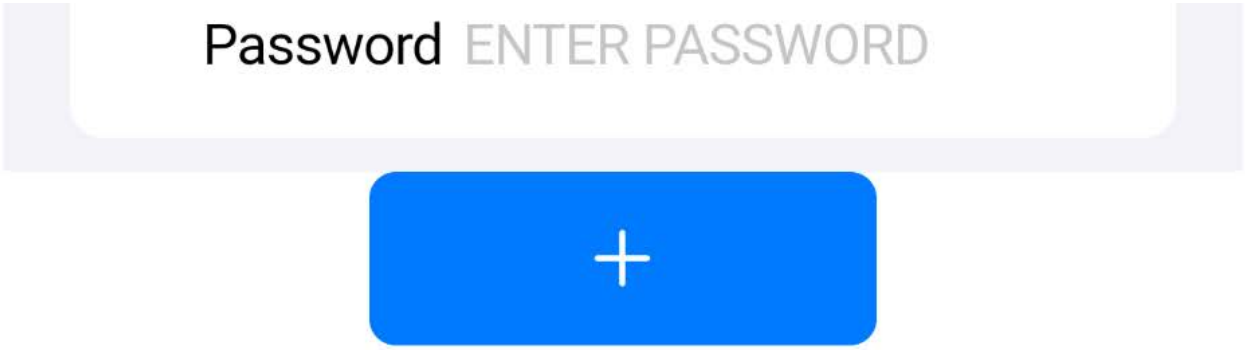
```
  FTPConnect ftpConnect = FTPConnect(ip, user: login, pass: password);

  void connect() async {
    print("Connecting...");
    await ftpConnect.connect();
  }

  void cdDir() async {
    try {
      String cd = await ftpConnect.currentDirectory();
      list.add("cd: $cd");
      print(list);
      controller.text = "";
      setState(() {});
    } catch (e) {
      print(e);
    }
  }

  void upload() async {
    try {
      String path = "/data/user/0/com.example.lab12/app_flutter/dir";
      Directory appDocDirectory = await getApplicationDocumentsDirectory();

      Directory('${appDocDirectory.path}/dir').create(recursive: true)
// The created directory is returned as a Future.
          .then((Directory directory) {
        print('Path of New Dir: ${directory.path}');
      });

      File('$path/${controller.text}').create(recursive: true);

      print("upload ${controller.text}");

      File fileToUpload = File("$path/${controller.text}");
      fileToUpload.writeAsString("ETO GOVNO RABOTAET?");
      String h = await fileToUpload.readAsString();
      print("UPLOAD FILE TEXT: $h");

      bool up = await ftpConnect.uploadFile(fileToUpload);
      list.add("${controller.text} upload: $up");
      print(list);
      controller.text = "";
      setState(() {});
    } catch (e) {
      print(e);
    }
  }

  void download() async {
    try {
      String path = "/data/user/0/com.example.lab12/app_flutter/dir";
      print("download ${controller.text}");

      bool dw = await ftpConnect.downloadFile(controller.text, File('$path/test2.tx
```

```
      File fileToUpload = File("$path/test2.txt");
      String h = await fileToUpload.readAsString();
      print("DOWNLOAD FILE TEXT: $h");

      list.add("${controller.text} download: $dw");
      print(list);
      controller.text = "";
      setState(() {});
    } catch (e) {
      //error
    }
  }

  void mkDir() async {
    try {
      print(controller.text);
      bool ch = await ftpConnect.makeDirectory(controller.text);
      list.add("Make dir is: $ch");
      print(list);
      controller.text = "";
      setState(() {});
    } catch (e) {
      print(e);
    }
  }

  void changeDir() async {
    try {
      print(controller.text);
      bool ch = await ftpConnect.changeDirectory(controller.text);
      list.add("Change dir is: $ch");
      print(list);
      controller.text = "";
      setState(() {});
    } catch (e) {
      print(e);
    }
  }

  @override
  Widget build(BuildContext context) {
    return CupertinoPageScaffold(
      child: Center(
        // child: Text("HUI", style: TextStyle(color: Colors.black),),)
        child: Container(
          padding: const EdgeInsets.all(20.0),
          child: Column(
            children: <Widget>[
              const SizedBox(
                height: 50,
              ),
              CupertinoFormSection.insetGrouped(
                header: const Text('Connection sttings'),
                children: [
```

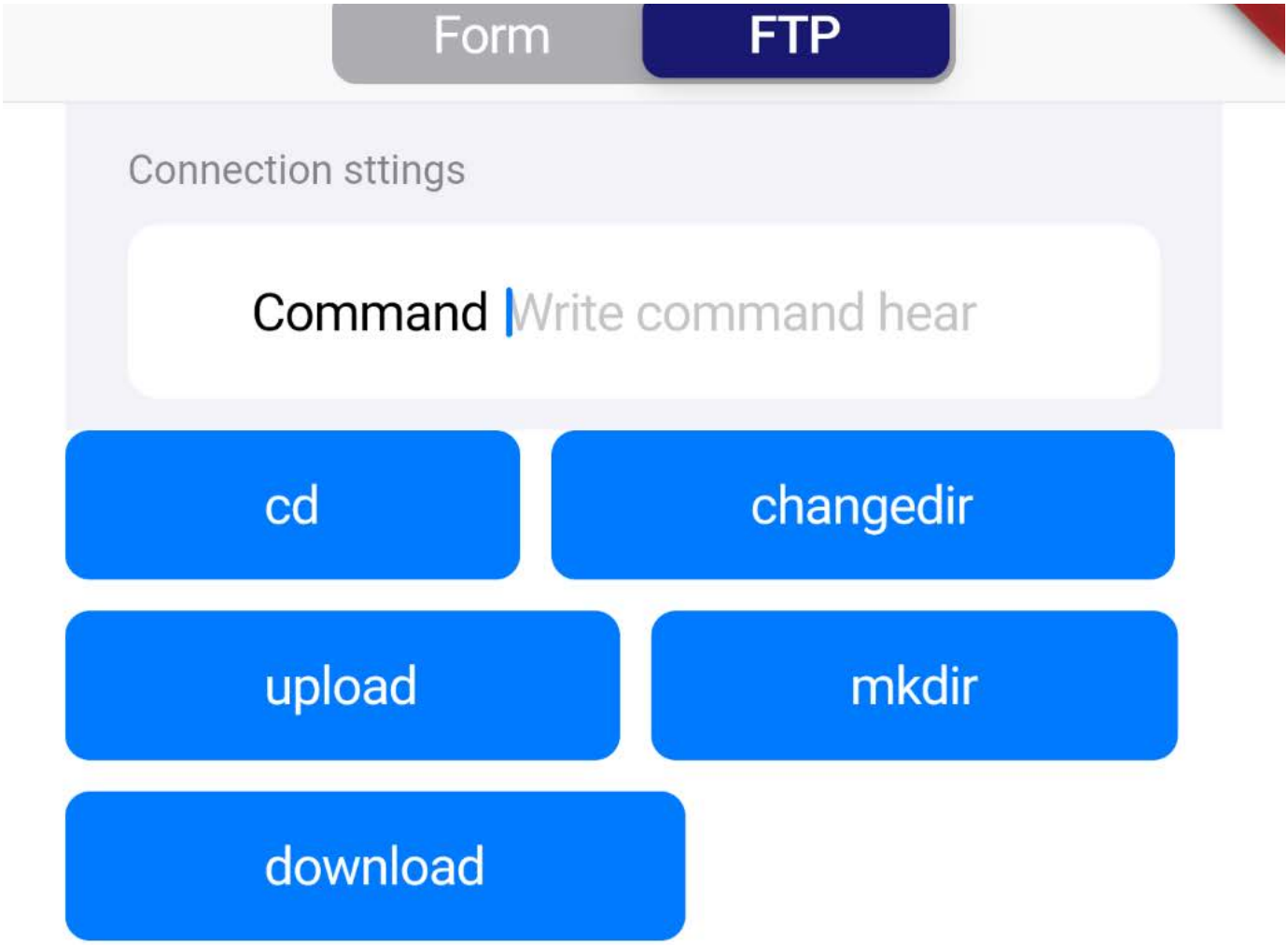
```
      CupertinoFormRow(
          child: CupertinoTextFormFieldRow(
        prefix: const Text('Command'),
        placeholder: "Write command hear",
        controller: controller,
      )),
    ],
  ),
  Row(
    children: [
      CupertinoButton.filled(
        onPressed: () => {
          if (list.length == 2)
            {
              connect(),
              Timer(Duration(seconds: 5), () => {cdDir()})
            }
          else
            {cdDir()}
        },
        child: const Text("cd"),
      ),
      const SizedBox(
        width: 10,
      ),
      CupertinoButton.filled(
        onPressed: () => {changeDir(), setState(() {})},
        child: const Text("changedir"),
      ),
      const SizedBox(
        width: 10,
      ),
    ],
  ),
  const SizedBox(
    height: 10,
  ),
  Row(
    children: [
      CupertinoButton.filled(
        onPressed: () => {upload(), setState(() {})},
        child: const Text("upload"),
      ),
      const SizedBox(
        width: 10,
      ),
      CupertinoButton.filled(
        onPressed: () => {mkDir(), setState(() {})},
        child: const Text("mkdir"),
      ),
    ],
  ),
  const SizedBox(
    height: 10,
  ),
  Row(
```

```
                Row(
                  children: [
                    CupertinoButton.filled(
                      onPressed: () => {download(), setState(() {})},
                      child: const Text("download"),
                    ),
                  ],
                ),
                const SizedBox(
                  height: 20,
                ),
                Expanded(
                    child: Column(
                  children: List.generate(
                      list.length,
                      (i) => Padding(
                            padding: const EdgeInsets.all(8.0),
                            child: Center(
                              child: Text(
                                list[i],
                                style: const TextStyle(color: Colors.black),
                              ),
                            ),
                          )),
                ))
              ],
            ),
          ),
        ),
      );
  }
}
```




Password
ENTER PASSWORD


17:18
DEBUG



Hello

lets go

cd: /

Make dir is: true

Change dir is: true

test.txt upload: true

test.txt download: true