# Лабораторная работа No 4

## «Интеграция Яндекс.Карт»

## Задача

В функционал приложения выполненного на предыдущей лабораторной работе встроить два виджета. На первом виджете реализовать форму добавления объектов согласно варианта из таблицы ниже. Объекты могут храниться как локально, так и во внешней базе данных, по желанию, пример работы с MySQL был разобран на одной из лекций. Во втором виджете должна отображаться Яндекс.Карта с расположенными на ней метками объектов, по клику на метку объекта должен открываться виджет с подробной информацией об объекте. Реализовать отображение объектов недвижимости на карте. Поля для хранения информации об объектах: географические координаты, адрес, метро, количество комнат, этаж, стоимость, фотография, телефон хозяина.

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

import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:ftpconnect/ftpConnect.dart';
import 'package:path_provider/path_provider.dart';
import 'package:yandex_mapkit/yandex_mapkit.dart';
import 'package:yandex_mapkit_example/examples/placemark_map_object_page.dart';

enum Sky { form, ftpClient, mapForm, yandexMap, item }

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

class Place {
  String latitude = "";
  String longtitude = "";
  String address = "";
  String metro = "";
  String club = "";
  String telephone = "";
  String foto = "";
}

List<Place> places = <Place>[];

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

```dart
Map<Sky, Color> skyColors = <Sky, Color>{
  Sky.form: const Color(0xff40826d),
  Sky.ftpClient: const Color(0xff191970),
  Sky.mapForm: const Color(0xeee41670),
  Sky.yandexMap: const Color(0xaaaaa670),
  Sky.item: const Color(0xbbbbbbbb)
};

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return const CupertinoApp(
      title: 'Flutter Demo',
      theme: CupertinoThemeData(brightness: Brightness.light),
      home: SegmentedControlApp(),
    );
  }
}

class SegmentedControlApp extends StatelessWidget {
  const SegmentedControlApp();

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

class SegmentedControlExample extends StatefulWidget {
  const SegmentedControlExample();

  @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) {
              if (value != null) {
                setState(() {
                  _selectedSegment = value;
                });
              }
            },
            children: const <Sky, Widget>{
              Sky.form: Padding(
                padding: EdgeInsets.symmetric(horizontal: 10),
                child: Text(
                  'Form',
                  style: TextStyle(color: CupertinoColors.white),
                ),
              ),
              Sky.ftpClient: Padding(
                padding: EdgeInsets.symmetric(horizontal: 10),
                child: Text(
                  'FTP',
                  style: TextStyle(color: CupertinoColors.white),
                ),
              ),
              Sky.mapForm: Padding(
                padding: EdgeInsets.symmetric(horizontal: 10),
                child: Text(
                  'MF',
                  style: TextStyle(color: CupertinoColors.white),
                ),
              ),
              Sky.yandexMap: Padding(
                padding: EdgeInsets.symmetric(horizontal: 10),
                child: Text(
                  'yM',
                  style: TextStyle(color: CupertinoColors.white),
                ),
              ),
              Sky.item: Padding(
                padding: EdgeInsets.symmetric(horizontal: 10),
                child: Text(
                  'it',
                  style: TextStyle(color: CupertinoColors.white),
                ),
              ),
            },
          ),
        ),
        child: Center(
          child: _selectedSegment.name == "form"
              ? MyHomePage()
              : (_selectedSegment.name == "ftpClient"
                  ? FtpPage()
                  : (_selectedSegment.name == "mapForm"
                      ? MapForm()
                      : (_selectedSegment.name == "item" ? ItemPage() : YM()))))
```

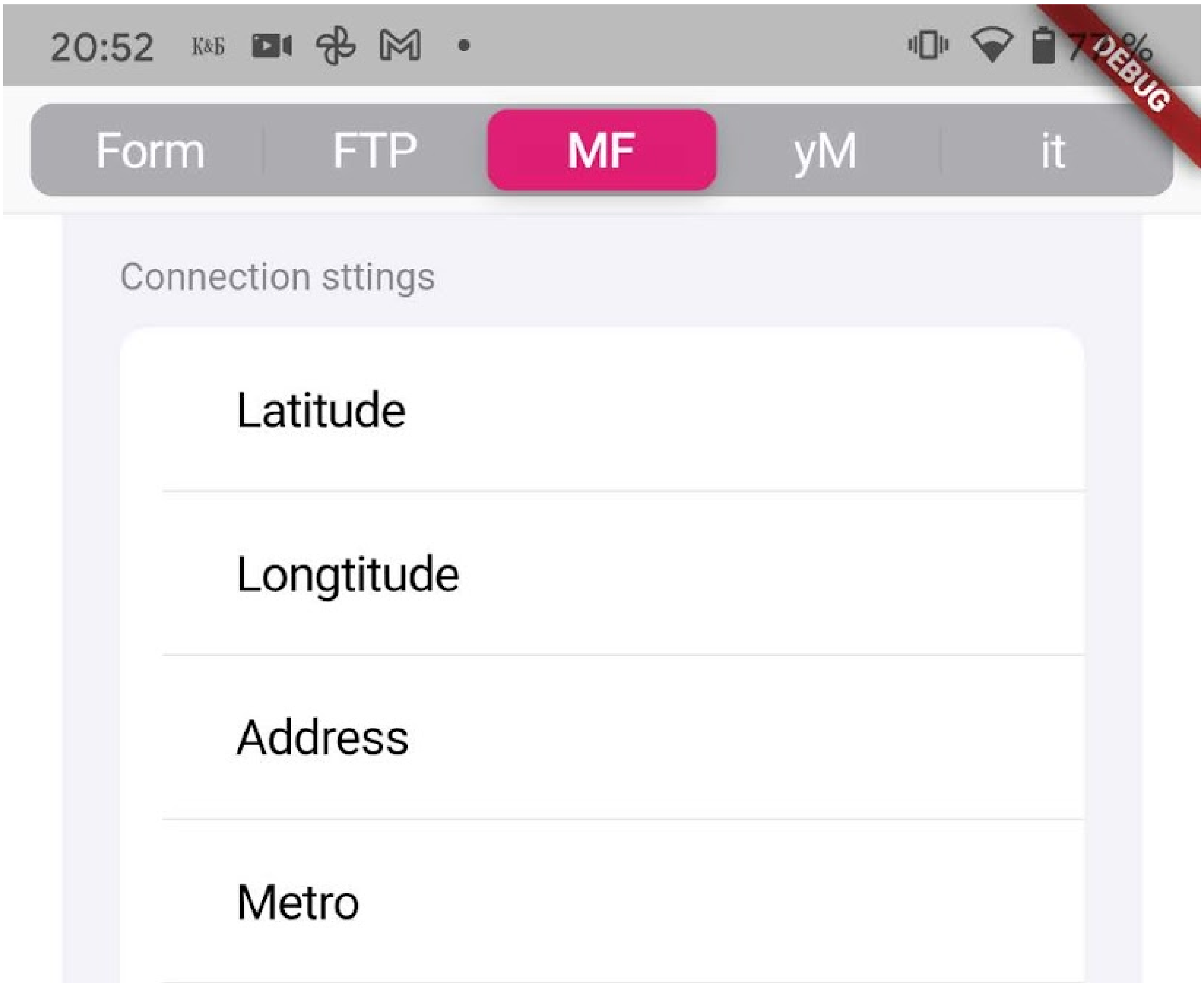
```
                    : (_selectedSegment.name == 'Item' ? ItemPage() : YM())));
      ));
  }
}

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 = "";
                    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.txt'));

      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(
  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),
              ),
            ),
          )),
))
```

```
          ],
        ),
      ),
    ),
  );
  }
}

class MapForm extends StatefulWidget {
  @override
  _MapFormState createState() => _MapFormState();
}

class _MapFormState extends State<MapForm> {
  TextEditingController controllerLatitude = TextEditingController();
  TextEditingController controllerLongtitude = TextEditingController();
  TextEditingController controllerAddress = TextEditingController();
  TextEditingController controllerMetro = TextEditingController();
  TextEditingController controllerClub = TextEditingController();
  TextEditingController controllerTelephone = TextEditingController();
  TextEditingController controllerFotography = 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('Latitude'),
                    placeholder: "",
                    controller: controllerLatitude,
                  )),
                  CupertinoFormRow(
                      child: CupertinoTextFormFieldRow(
                    prefix: const Text('Longtitude'),
                    placeholder: "",
                    controller: controllerLongtitude,
                  )),
                  CupertinoFormRow(
                      child: CupertinoTextFormFieldRow(
                    prefix: const Text('Address'),
                    placeholder: "",
                    controller: controllerAddress,
                  )),
                CupertinoFormRow(
                    child: CupertinoTextFormFieldRow(
                  prefix: const Text('Metro'),
                  placeholder: "",
                  controller: controllerMetro,
                )),
                CupertinoFormRow(
                    child: CupertinoTextFormFieldRow(
                  prefix: const Text('Club'),
                  placeholder: "",
                  controller: controllerClub,
                )),
                CupertinoFormRow(
                    child: CupertinoTextFormFieldRow(
                  prefix: const Text('Telephone'),
                  placeholder: "",
                  controller: controllerTelephone,
                )),
                CupertinoFormRow(
                    child: CupertinoTextFormFieldRow(
                  prefix: const Text('Address'),
                  placeholder: "",
                  controller: controllerFotography,
                )),
              ],
            ),
            CupertinoButton.filled(
              onPressed: () => {
                setState(() {
                  Place pl = Place();
                  pl.longtitude = controllerLongtitude.text;
                  pl.latitude = controllerLatitude.text;
                  pl.address = controllerAddress.text;
                  pl.club = controllerClub.text;
                  pl.foto = controllerFotography.text;
                  pl.metro = controllerMetro.text;
                  pl.telephone = controllerTelephone.text;
                  places.add(pl);
                  controllerLongtitude.text = "";
                  controllerLatitude.text = "";
                  controllerAddress.text = "";
                  controllerClub.text = "";
                  controllerFotography.text = "";
                  controllerMetro.text = "";
                  controllerTelephone.text = "";
                })
              },
              child: const Icon(CupertinoIcons.add),
            ),
          ],
        ),
      ),
    ),
  );
}
```

```
,

class YM extends StatefulWidget {
  @override
  _YandexMaptate createState() => _YandexMaptate();
}

class _YandexMaptate extends State<YM> {
  void getPlaceMarks() {
    print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
    print(places.length.toString());
    for (var i = 0; i < places.length; i++) {
      print(places[i].address);
      var placemarkMapObject = PlacemarkMapObject(
          mapId: MapObjectId(places[i].telephone),
          point: Point(latitude: double.parse(places[i].latitude), longitude: doubl
          onTap: (PlacemarkMapObject self, Point point) => {
            debugPrint("sjopa"),
            latitude = places[i].latitude,
            longtitude = places[i].longtitude,
            address = places[i].address,
            metro = places[i].metro,
            club = places[i].club,
            telephone = places[i].telephone,
            foto = places[i].foto,
            setState(() {})
          },
          opacity: 0.7,
          direction: 90,
          isDraggable: true,
          icon: PlacemarkIcon.single(PlacemarkIconStyle(
              image: BitmapDescriptor.fromAssetImage('lib/assets/place.png'),
              rotationType: RotationType.rotate)));
      mapObjects.add(placemarkMapObject);
      print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
      print(mapObjects[i].mapId);
    }
  }
  void getPlace() {
    Place pl1 = Place();
    pl1.latitude= "55.671407";
    pl1.longtitude= "37.877958";
    pl1.address = "Московская область, Люберцы, 115-й квартал, улица Авиаторов, 8, 140011"
    pl1.metro = "Котельники";
    pl1.club = "None";
    pl1.telephone = "89269098365";
    pl1.foto = "https://core-pht-proxy.maps.yandex.ru/v1/photos/download?photo_id=V
    places.add(pl1);
    Place pl2 = Place();
    pl2.latitude= "55.765932";
    pl2.longtitude= "37.684555";
    pl2.address = "2-я Бауманская улица, 5с1, Москва, 105005";
    pl2.metro = "Бауманская";
    pl2.club = "None";
    pl2.telephone = "84992636391";
    pl2.foto = "https://avatars.mds.yandex.net/get-altay/1608507/2a00000168d1eeba9f
    [illegible]
    places.add(pl2);
  }

  // PlacemarkMapObject(mapId: MapObjectId('placemark_3'), point: Point(latitude: 5
  YandexMap map = YandexMap();
  final List<MapObject> mapObjects = [];

  @override
  Widget build(BuildContext context) {

    getPlace();
    getPlaceMarks();
    return CupertinoPageScaffold(
      child: Center(
        // child: Text("HUI", style: TextStyle(color: Colors.black),),)
        child: Container(
          padding: const EdgeInsets.all(20.0),
          child: Column(
            children: <Widget>[
              Expanded(
                  child: Container(
                      padding: const EdgeInsets.all(8),
                      child: YandexMap(mapObjects: mapObjects))),
            ],
          ),
        ),
      ),
    );
  }
}

String latitude = "holder";
String longtitude = "holder";
String address = "holder";
String metro = "holder";
String club = "holder";
String telephone = "holder";
String foto = "holder";

class ItemPage extends StatelessWidget {
  ItemPage();

  @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: 20,
              ),
              Image.network(foto),
              const SizedBox(
                height: 20,
              ),
              Text(address),
              const SizedBox(
                height: 20,
              ),
              Text(metro),
              const SizedBox(
                height: 20,
              ),
              Text(club),
              const SizedBox(
                height: 20,
              ),
              Text(telephone),
            ],
          ),
        ),
      ),
    );
  }
}
```



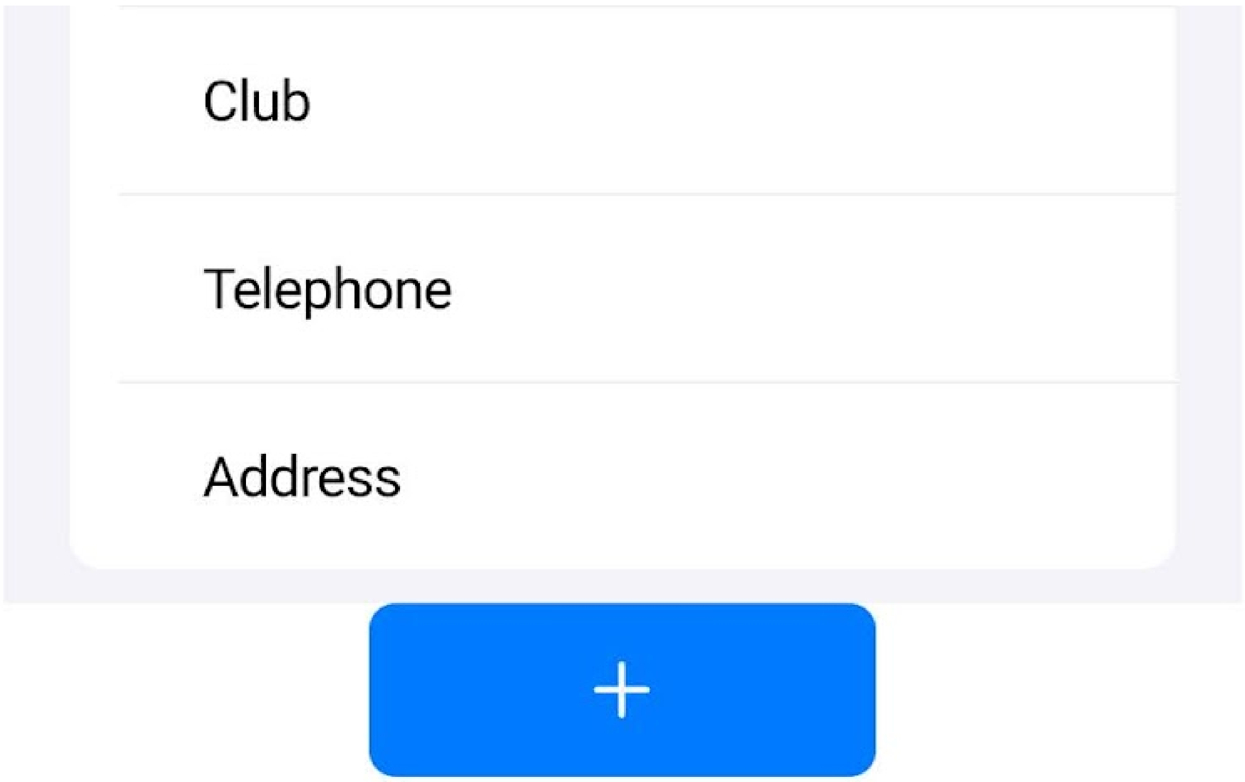
Connection sttings

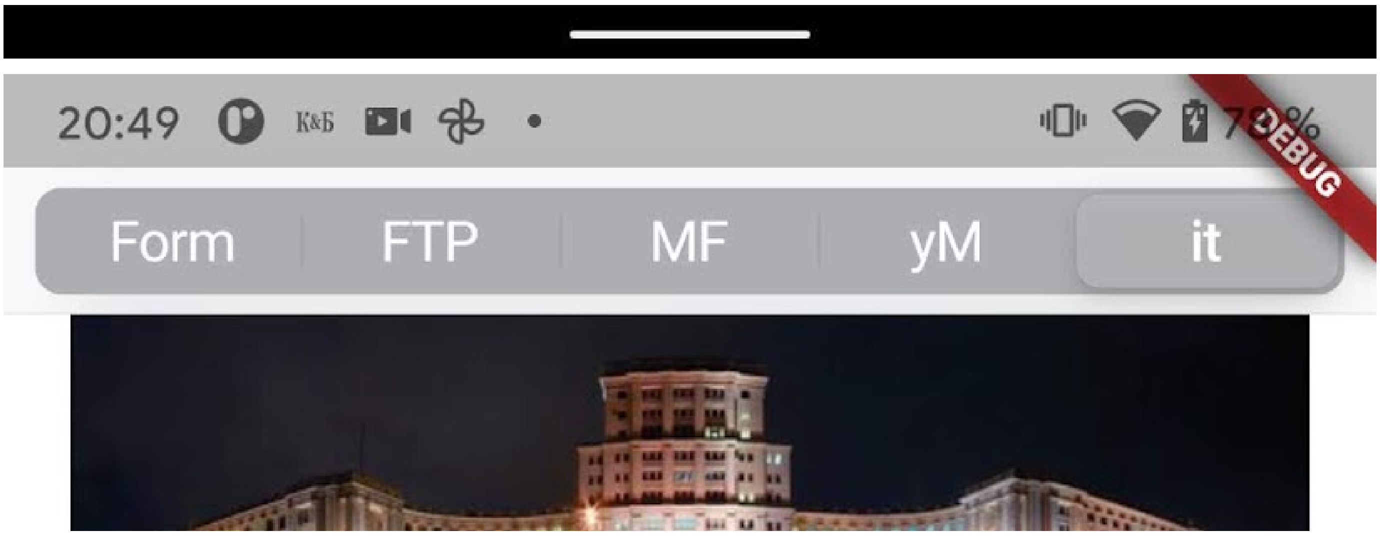
Latitude

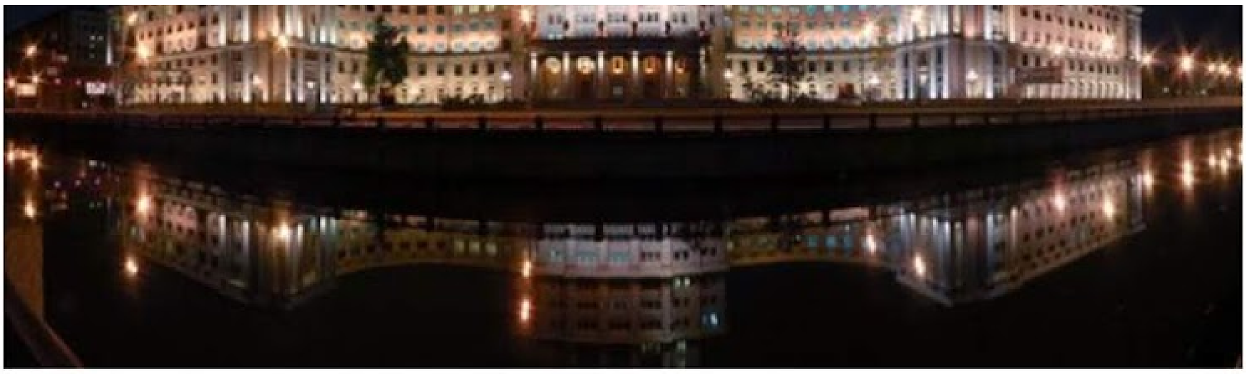
Longtitude

Address

Metro


Club
Telephone
Address
+


20:49
K&Б
DEBUG
Form
FTP
MF
yM
it

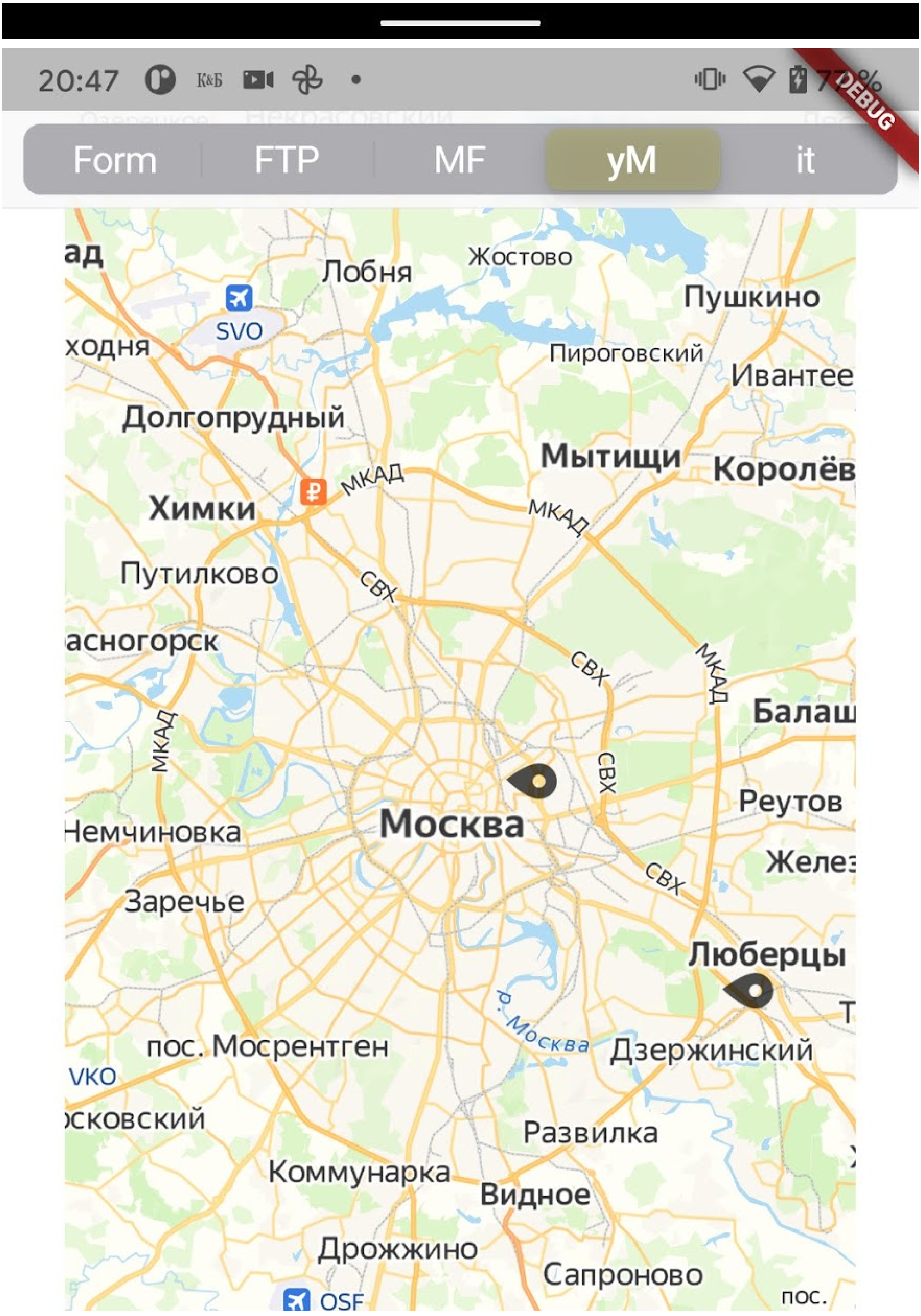


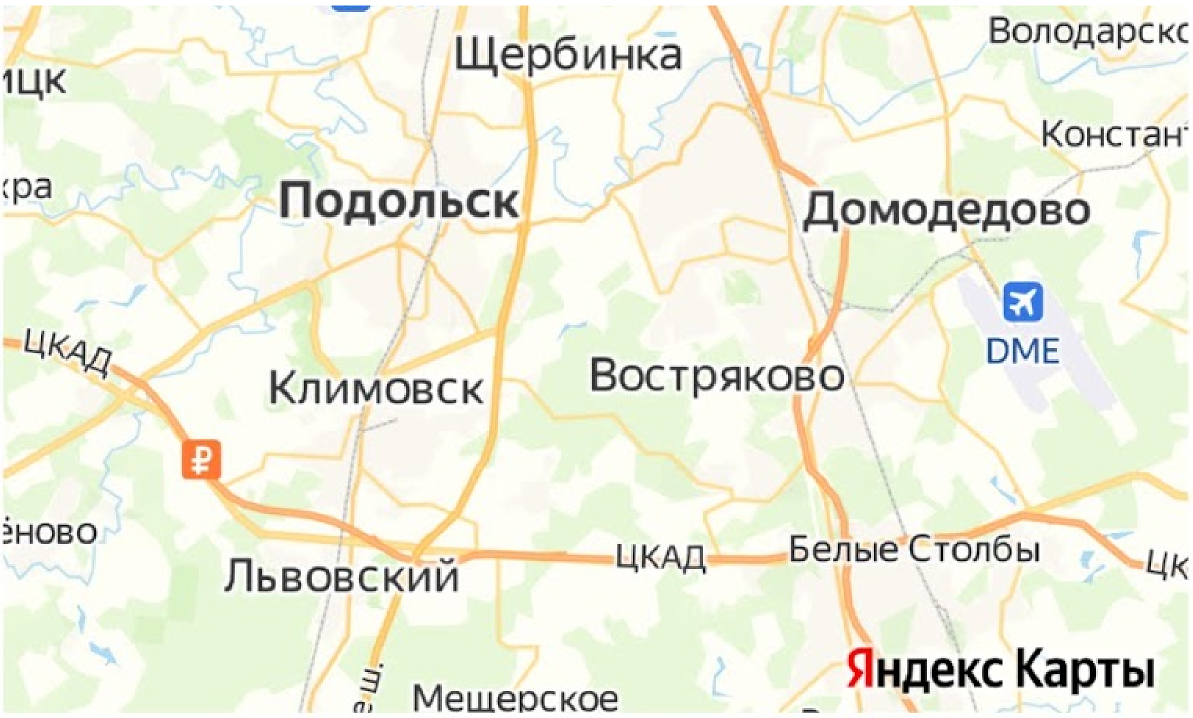
2-я Бауманская улица, 5с1, Москва, 105005

Бауманская

None

84992636391


20:47
DEBUG
Form
FTP
MF
yM
it
Лобня
Жостово
Пушкино
SVO
ходня
Пироговский
Ивантее
Долгопрудный
Мытищи
Королёв
МКАД
Химки
Путилково
СВХ
асногорск
Балаш
Немчиновка
Москва
Реутов
Желез
Заречье
Люберцы
р. Москва
пос. Мосрентген
Дзержинский
VKO
осковский
Развилка
Коммунарка
Видное
Дрожжино
Сапроново
OSF
пос.


Щербинка
Володарско
ицк
Констант
кра
Подольск
Домодедово
DME
ЦКАД
Климовск
Востряково
ёново
ЦКАД
Белые Столбы
ЦК
Львовский
Яндекс Карты
ш.
Мещерское