МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королёва»
(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра информационных систем и технологий

ОТЧЕТ

по лабораторной работе №4 по дисциплине

«Менеджмент разработки программного обеспечения»

Выполнили:

Гуреев М.А., гр. 6222-090401D

Елфимов А.Г., гр. 6222-090401D

Филатов В.В., гр. 6222-090401D

Проверил:

доцент каф. ИСТ, к.т.н. Крупец Н.Г.

Самара 2020

ЦЕЛЬ РАБОТЫ

Разработать мобильное приложение для диспетчера энергосбытовой компании.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ

Объектом автоматизации является процесс контрольного снятия показаний с приборов учёта электроэнергии. Он состоит в том, что энегосбытовая организация вынуждена проводить контрольные проверки показаний приборов учёта, для своевременно устранения неисправностей и недопущения получения некорректных показаний.

Мобильное приложение для диспетчера

К основным задачам диспетчера относятся контроль над работой и своевременное устранение проблем возникающих у контролёров энергосбыта во время снятия показаний. Необходимо обеспечить взаимодействие этих двух типов сотрудников. Обеспечим диспетчеру следующие возможности:

1. Информирование всех контролёров актуальной информацией по работе.
2. Добавление точки учёта в базу данных и назначение на неё контролёра.
3. Просмотр информации о профиле.
4. Просмотр состояния точек учёта.
5. Просмотр информационной рассылки для всех пользователей системы.

Далее представлены скриншоты программы. На рисунке 1 показано окно авторизации пользователя. Рисунок 2 – главное меню программы. На нём есть возможности добавить точку учёта (рисунок 3), создать сообщение в информационную рассылку (рисунок 4), просмотреть список ошибок на точках учёта (рисунок 5), просмотреть информационную рассылку (рисунок 6) и информацию о профиле (рисунок 7).

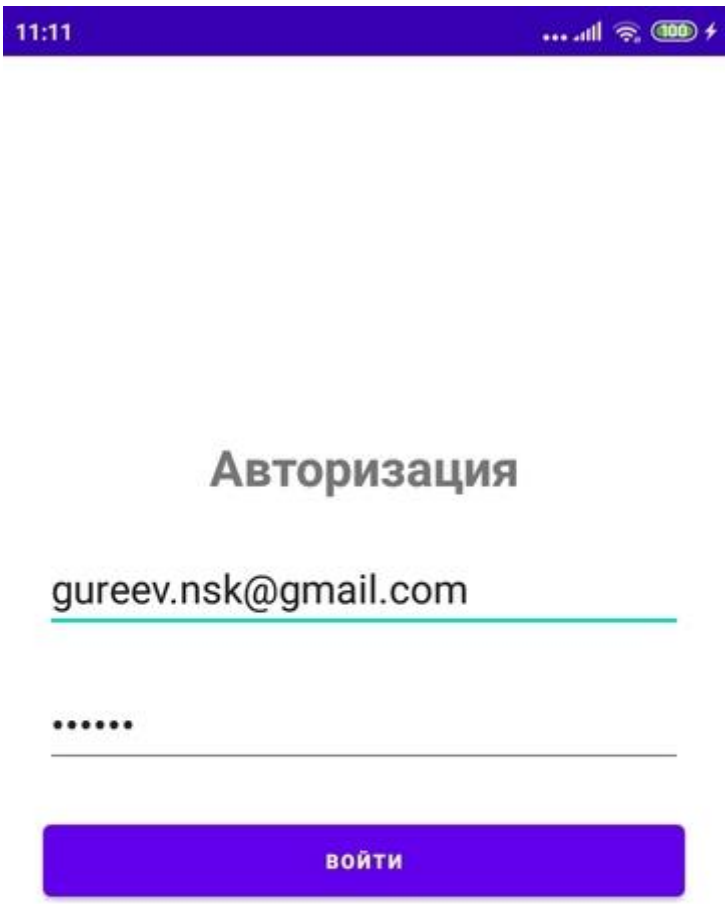

Рисунок 1 – Экран авторизации

Рисунок 2 – Экран главного меню

Рисунок 3 – Экран добавления точки учёта

Рисунок 4 – Экран добавления сообщения в информационную рассылку

Рисунок 5 – Экран просмотра списка ошибок на точках учёта

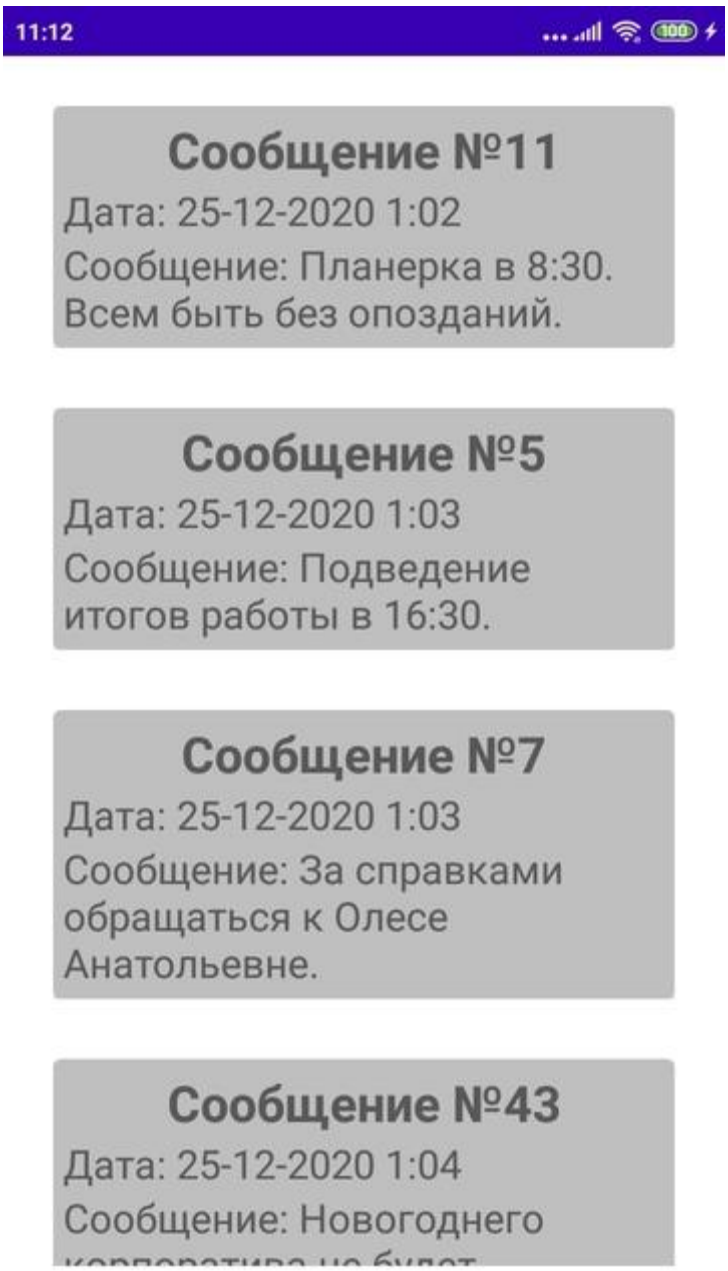

Рисунок 6 – Экран просмотра информационной рассылки

Рисунок 7 – Экран информации о пользователе

Приложение 1.

Код основных операций программы.

```
public class NetworkService {

    private Activity activity;
    private String login;

    private FirebaseAuth mAuth;
    private FirebaseUser currentUser;
    private FirebaseFirestore db;
    private StorageReference mStorageRef;

    private MutableLiveData<User> currentUserInfo = new MutableLiveData<>();
    private MutableLiveData<List<Message>> messagesList = new MutableLiveData<>();
    private MutableLiveData<List<User>> usersList = new MutableLiveData<>();
    private MutableLiveData<List<MeterPoint>> meterPointsList = new
MutableLiveData<>();

    private MutableLiveData<Boolean> isUploadPhoto = new MutableLiveData<>();


    public NetworkService(Activity activity) {
        initFB(activity);
    }

    private void initFB(Activity activity) {
        mAuth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();
        mStorageRef = FirebaseStorage.getInstance().getReference();
        this.activity = activity;
    }

    public void signIn(String email, String password) {
        currentUserInfo = new MutableLiveData<>();
        mAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(activity, task -> {
                    if (task.isSuccessful()) {
                        Log.d(TAG, "signInWithEmail:success");
                        currentUser = mAuth.getCurrentUser();
                        login = currentUser.getEmail();
                        Toast.makeText(activity, "Authentication success.",
                                Toast.LENGTH_SHORT).show();
                        getUsersInfo();
                    } else {
                        Log.w(TAG, "signInWithEmail:failure", task.getException());
                        Toast.makeText(activity, "Authentication failed.",
                                Toast.LENGTH_SHORT).show();
                    }
                });
    }


    private void getUsersInfo() {
        usersList.setValue(new ArrayList<>());
        db.collection("users")
                .get()
                .addOnCompleteListener(task -> {
                    User tempUser = new User();
                    tempUser.setPosition("no");
                    if (task.isSuccessful()) {
```

```
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        Log.d(TAG, document.getId() + " => " + document.getData());
                        addUserOfDocument(document);
                    }
                } else {
                    Log.w(TAG, "Error getting documents.", task.getException());
                    tempUser.setPosition("no");
                    currentUserInfo.setValue(tempUser);
                }
            });
}

private void addUserOfDocument(QueryDocumentSnapshot document) {
    User tempUser = new User();
    List<User> tempUserList;
    if (usersList.getValue() != null) {
        tempUserList = usersList.getValue();
    } else {
        tempUserList = new ArrayList<>();
    }

    tempUser.setId((String) document.getData().get("id"));
    tempUser.setName((String) document.getData().get("name"));
    tempUser.setLastName((String) document.getData().get("lastName"));
    tempUser.setFatherName((String) document.getData().get("fatherName"));
    tempUser.setBrigadeNum((String) document.getData().get("brigade_num"));
    tempUser.setPosition((String) document.getData().get("position"));
    tempUser.setDepartmentNum((String) document.getData().get("department_num"));
    tempUser.setPersonalNum((String) document.getData().get("personal_num"));

    if (document.getId().trim().equals(login)) {
        currentUserInfo.setValue(tempUser);
    }

    tempUserList.add(tempUser);
    usersList.setValue(tempUserList);
}

public void addMessage(Message message) {

    Map<String, Object> tempMessage = new HashMap<>();
    tempMessage.put("id", message.getId());
    tempMessage.put("date", message.getDate());
    tempMessage.put("title", message.getTitle());
    tempMessage.put("text", message.getText());

    db.collection("messages").document(message.getId())
            .set(message)
            .addOnSuccessListener(documentReference -> {
                Toast.makeText(activity, "Сообщение успешно добавлено.",
                        Toast.LENGTH_SHORT).show();
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Log.w(TAG, "Error adding document", e);
                    Toast.makeText(activity, "Сообщение не удалось добавить.",
                            Toast.LENGTH_SHORT).show();
                }
            });
}
```

```
public void getMessages() {
    messagesList.setValue(new ArrayList<>());
    db.collection("messages")
            .get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        Log.d(TAG, document.getId() + " => " + document.getData());
                        addMessageOfDocument(document);
                    }
                } else {
                    Log.w(TAG, "Error getting documents.", task.getException());
                }
            });
}

private void addMessageOfDocument(QueryDocumentSnapshot document) {
    Message tempMessage = new Message();
    List<Message> tempMessageList;

    if (messagesList.getValue() != null) {
        tempMessageList = messagesList.getValue();
    } else {
        tempMessageList = new ArrayList<>();
    }

    tempMessage.setDate((String) document.getData().get("date"));
    tempMessage.setTitle((String) document.getData().get("title"));
    tempMessage.setText((String) document.getData().get("text"));

    tempMessageList.add(tempMessage);
    messagesList.setValue(tempMessageList);
}

public void addMeterPoint(MeterPoint meterPoint) {

    Map<String, Object> tempMeterPoint = new HashMap<>();
    tempMeterPoint.put("id", meterPoint.getId());
    tempMeterPoint.put("pointNumber", meterPoint.getPointNumber());
    tempMeterPoint.put("userId", meterPoint.getUserId());
    tempMeterPoint.put("city", meterPoint.getCity());
    tempMeterPoint.put("district", meterPoint.getDistrict());
    tempMeterPoint.put("street", meterPoint.getStreet());
    tempMeterPoint.put("house", meterPoint.getHouse());
    tempMeterPoint.put("flat", meterPoint.getFlat());
    tempMeterPoint.put("description", meterPoint.getDescription());
    tempMeterPoint.put("deviceStatus", meterPoint.getDeviceStatus());
    tempMeterPoint.put("numMeter", meterPoint.getNumMeter());
    tempMeterPoint.put("isProblemPoint", meterPoint.isProblemPoint());
    tempMeterPoint.put("isReadingDone", meterPoint.isReadingDone());
    tempMeterPoint.put("problemDescription", meterPoint.getProblemDescription());
    tempMeterPoint.put("meterReading", meterPoint.getMeterReading());
    tempMeterPoint.put("imageUrl1", meterPoint.getImageUrl1());
    tempMeterPoint.put("imageUrl2", meterPoint.getImageUrl2());
    tempMeterPoint.put("imageUrl3", meterPoint.getImageUrl3());
    tempMeterPoint.put("date", meterPoint.getDateMetering());

    db.collection("meterPoints").document(meterPoint.getId())
            .set(tempMeterPoint)
            .addOnSuccessListener(documentReference -> {
            })
            .addOnFailureListener(e -> {
```

```
                    Log.w(TAG, "Error adding document", e);
                });
    }

    public void getMeterPoints() {
        meterPointsList.setValue(new ArrayList<>());
        db.collection("meterPoints")
                .get()
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            Log.d(TAG, document.getId() + " => " + document.getData());
                            addMeterPointsOfDocument(document);
                        }
                    } else {
                        Log.w(TAG, "Error getting documents.", task.getException());
                    }
                });
    }

    private void addMeterPointsOfDocument(QueryDocumentSnapshot document) {
        MeterPoint tempMeterPoint = new MeterPoint();
        List<MeterPoint> tempMeterPointsList;

        if (meterPointsList.getValue() != null) {
            tempMeterPointsList = meterPointsList.getValue();
        } else {
            tempMeterPointsList = new ArrayList<>();
        }

        tempMeterPoint.setId((String) document.getData().get("id"));
        tempMeterPoint.setPointNumber((String) document.getData().get("pointNumber"));
        tempMeterPoint.setUserId((String) document.getData().get("userId"));
        tempMeterPoint.setCity((String) document.getData().get("city"));
        tempMeterPoint.setDistrict((String) document.getData().get("district"));
        tempMeterPoint.setStreet((String) document.getData().get("street"));
        tempMeterPoint.setHouse((String) document.getData().get("house"));
        tempMeterPoint.setFlat((String) document.getData().get("flat"));
        tempMeterPoint.setDescription((String) document.getData().get("description"));
        tempMeterPoint.setDeviceStatus((String)
document.getData().get("deviceStatus"));
        tempMeterPoint.setNumMeter((String) document.getData().get("numMeter"));
        tempMeterPoint.setProblemPoint((boolean)
document.getData().get("isProblemPoint"));
        tempMeterPoint.setReadingDone((boolean)
document.getData().get("isReadingDone"));
        tempMeterPoint.setProblemDescription((String)
document.getData().get("problemDescription"));
        tempMeterPoint.setMeterReading((String)
document.getData().get("meterReading"));
        tempMeterPoint.setImageUrl1((String) document.getData().get("imageUrl1"));
        tempMeterPoint.setImageUrl2((String) document.getData().get("imageUrl2"));
        tempMeterPoint.setImageUrl3((String) document.getData().get("imageUrl3"));
        tempMeterPoint.setDateMetering((String)
document.getData().get("dateMetering"));

        tempMeterPointsList.add(tempMeterPoint);
        meterPointsList.setValue(tempMeterPointsList);
    }

    public void addMeterPointPhoto(Uri uri, String meterPointId, String imageName) {
        isUploadPhoto.setValue(false);
```

```
        StorageReference riversRef =
mStorageRef.child(String.format("images/%s/%s.jpg", meterPointId, imageName));

        riversRef.putFile(uri)
                .addOnSuccessListener(taskSnapshot -> {
                    //Uri downloadUrl = taskSnapshot.getUploadSessionUri();
                    isUploadPhoto.setValue(true);
                    Toast.makeText(activity, "Фото успешно добавлено.",
                            Toast.LENGTH_SHORT).show();
                })
                .addOnFailureListener(exception -> {
                    Log.w(TAG, "Error adding document", exception);
                    Toast.makeText(activity, "Не удалось добавить фото.",
                            Toast.LENGTH_SHORT).show();
                });
    }

    public MutableLiveData<User> getCurrentUserInfo() {
        return currentUserInfo;
    }

    public MutableLiveData<List<Message>> getMessagesList() {
        return messagesList;
    }

    public MutableLiveData<List<User>> getUsersList() {
        return usersList;
    }

    public MutableLiveData<List<MeterPoint>> getMeterPointsList() {
        return meterPointsList;
    }

    public MutableLiveData<Boolean> getIsUploadPhoto() {
        return isUploadPhoto;
    }
}
```