|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | |
| Институт информационных технологий | |
| Кафедра математического обеспечения и стандартизации информационных систем(МОСИТ)  **КУРСОВАЯ РАБОТА**  по дисциплине  «Функциональное и логическое программирование»  **Тема курсовой работы**: «Анализ предметной области и создание мобильного приложения согласно теме варианта» | |
| **Студент** группы ИКБО-07-18 | Зейналов Магеррам Гилал Оглы |
|  |  |
|  | (подпись студента) |
|  |  |
| **Руководитель курсовой работы** | ст. преподаватель Миронов А.Н. |
|  |  |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись руководителя) |
|  |  |
| Работа представлена к защите | «» 2020 г |
|  |  |
| Допущен к защите | «» 2020 г. |

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Утверждаю

Заведующий кафедрой МОСИТ

Головин С.А.

« » 2020 г.

## ЗАДАНИЕ

**на выполнение курсовой работы по дисциплине**

«Функциональное и логическое программирование»

|  |  |
| --- | --- |
| Студент Зейналов Магеррам Гилал Оглы | Группа ИКБО-07-18 |

**Тема работы:** Анализ предметной области и создание мобильного решения согласно теме варианта.

**Исходные данные:** Вариант № 20. Предметная область – Общение людей

## Перечень вопросов, подлежащих разработке, и обязательного графического материала:

1. Анализ предметной области для нахождения в ней проблем и «болей рынка»
2. Изучения проблематики предметной области на конкретном бизнес процессе в данной предметной области
3. Подготовка минимального функционала по проанализированному бизнес процессу;
4. Разработка схем IDEF0 и DFD для анализа внедрения приложения в работающий бизнес процесс и нагрузки системы данными.
5. Разработка и проектирование архитектуры создаваемой информационной системы.
6. Создание портретов пользователей разрабатываемого приложения
7. Создание информационной схемы мобильного приложения
8. Создание интерактивного макета разрабатываемого мобильного приложения
9. Разработка мобильного приложения;
10. Разработка или подключение остальных элементов разрабатываемой информационной системы.

**Срок представления к защите курсовой работы:** до « » 2020 г.

## Задание на курсовую работу выдал (Миронов А. Н.)

« » 2020 г.

## Задание на курсовую работу получил (Зейналов М. Г.)

# Содержание

[Содержание 4](#_Toc59643488)

[Введение 5](#_Toc59643489)

[Основная часть 6](#_Toc59643490)

[Раздел 1. Теоретическое введение 6](#_Toc59643491)

[Раздел 2. Проектная часть 7](#_Toc59643492)

[Раздел 3. Экспериментальная часть 16](#_Toc59643493)

[Заключение 19](#_Toc59643494)

[Список используемых источников 20](#_Toc59643495)

# ВВедение

В современном обществе важно иметь способ быстрой коммуникации. Данным способом является – мессенджеры. Актуальность данной темы подтверждается последними событиями в мире, где нам нужно было передавать друг другу информацию на расстоянии.

В ходе выполнения курсовой работы была рассмотрена предметная область «общение людей» и были выдвинуты главные функции системы.

В основной части курсовой работы будет представлена краткая характеристика предметной области, возможные сценарии и пути разработки. Осуществляется постановка задачи, определение минимального функционала приложения. Проектирование разрабатываемого приложения и экспериментальная часть с разработкой приложения.

# основная часть

## Раздел 1. Теоретическое введение

Общение всегда играло важную роль в жизни человека. Человек – существо социальное, и нам всегда было необходимо общаться с друг другом. С течением времени, многие процессы в жизни менялись, но потребность в общении никуда не исчезла, менялись только способы.

Благодаря интернету и доступности мобильных устройств человечество перешло в новый этап общения. Начали развиваться мессенджеры, социальные сети, видеохостинги и иные способы мобильной передачи информации. Биты информации летят по проводам, передавая данные в самые разные уголки мира за считанные миллисекунды.

В качестве реализации социальных сетей и мессенджеров используется клиент-серверная архитектура. Ее основной смысл состоит в разделении ролей на пользователей и серверов, которые будут обрабатывать запросы от пользователей. Преимуществом такого подхода является то, что клиент не знает реализацию функций приложения, тем самым защищая сервис от некоторых видов атак. Также разгружается устройство клиента отсутствием необходимости хранить реализации функций и поддерживать последнюю версию кода с данными приложениями.

Основной функционал таких приложений – отправка и прием сообщений. С эволюцией мессенджеров добавились такие функции, как голосовые сообщения, расшифровка голосовых сообщений, истории и каналы.

## Раздел 2. Проектная часть

#### **2.1 Постановка задачи**

Задачей моей курсовой работы является реализовать альтернативный способ общения людей со списком контактов и сообщений.

#### **2.2 Проектное решение**

В качестве архитектуры был взят клиент-серверный подход с реализацией REST API.

Реализации клиентской части реализовано с помощью языка программирования Java с Android SDK [3].

Реализация серверной части реализовано с помощью языка программирования Java и Spring Boot [1]. В качестве БД решено использовать MySQL.

Функционал разрабатываемого приложения:

1. Вход;
2. Регистрация;
3. Добавление контактов;
4. Отправка сообщений;
5. Просмотр сообщений;
6. Просмотр аккаунтов.

Среди возможных доработок системы возможны:

1. Улучшение безопасности программы;
2. Возможность отправлять фотографии и видео;
3. Добавление голосовых сообщений;
4. Написание отдельных сервисов.

Рассмотрим реализацию серверной части приложения. Как было ранее сказано, серверная часть реализована с помощью Spring Boot.

Наш backend состоял из 3 контроллеров:

1. UserController
2. MessageController
3. ContactController

**UserController**

В этом контроллере были реализованы вход и регистрация. При регистрации мы обращались к БД и проверяли, существует ли пользователь с таким ником в системе, если ответ приходил утвердительный, то мы выкидывали исключение, в котором указывали, что пользователь с таким ником уже есть. Пароли шифруются с помощью алгоритма BCrypt.

Вход же реализован в виде симуляции, просто проверяя есть ли такой пользователь с паролем в БД и возвращая в качестве ответа сущность с true/false. Такой подход было решено использовать из-за трудоемкости настройки безопасности в серверной части. Конечно, можно было использовать JWT авторизацию с refresh token, которая бы решила проблему повторного входа при запуске приложения, но это слишком трудоемко для 3 контроллеров.

**MessageController**

Основными функциями этого контроллера являются получение и сохранение сообщений. Получение сообщений реализовано довольно просто: сервер обращается к базе данных и возвращает список сообщений для конкретного пользователя. Сохранение тоже сильно не отличается сложностью реализации. Сервер получает данные от клиента, упаковывает в объект и сохраняет в БД.

**ContactController**

Реализовано аналогично MessageController. В привязке аккаунта добавлена проверка на существование такого пользователя.

**Хранение данных**

Для хранения данных используются принциаы работы ORM. Сервер не работает напрямую через SQL скрипты, а использует прослойку в виде Spring Data + Hibernate. Создаются классы-домены, которые помечаются специальными аннотациями, чтобы Spring мог понять, какие классы использовать для создания таблиц. Далее создаются интерфейсы-репозитории, в которых с помощью кода прописываются запросы. Данные интерфейсы отвечают за манипуляцию с БД.

Весь код проекта будет доступен в GitHub [5].

На рис.1 представлена архитектура создаваемой информационной системы

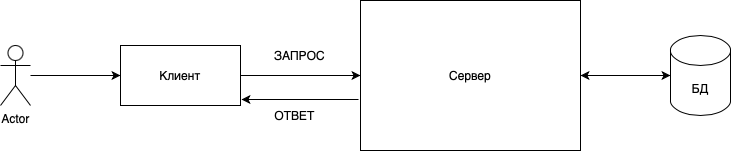


Рисунок 1 – Архитектура создаваемой информационной системы

На рис.2 представлена IDEF0 – диаграмма внедрения приложения в работающий бизнес-процесс

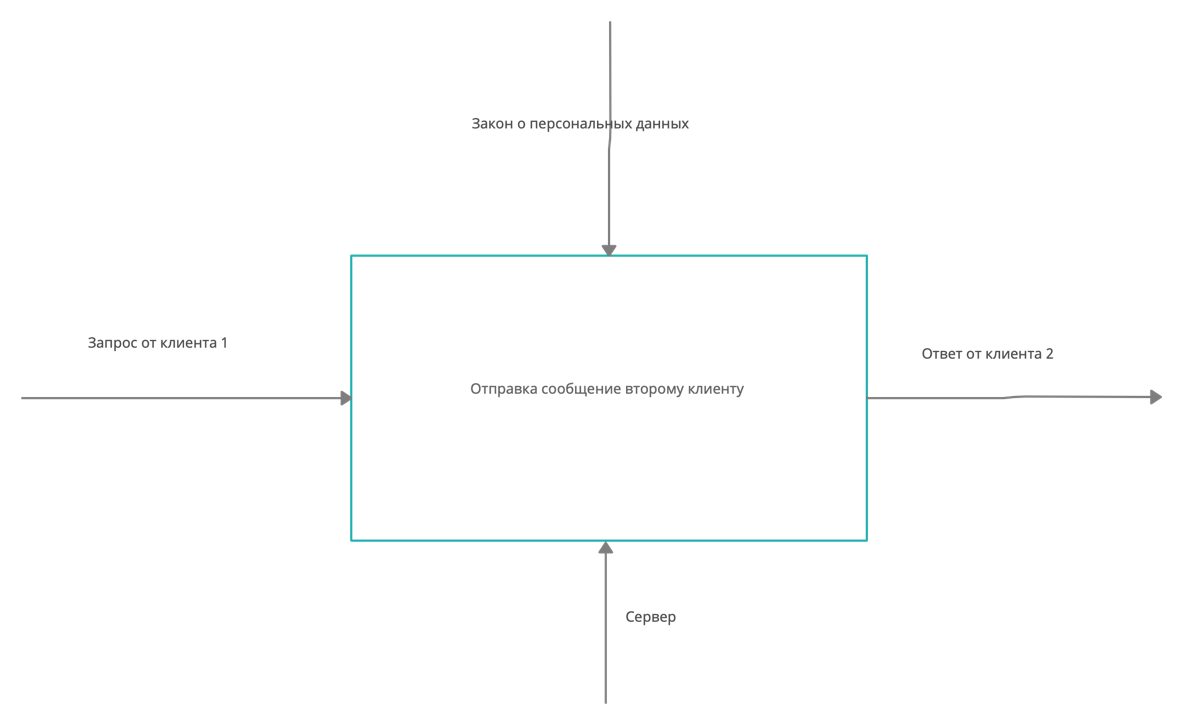


Рисунок 2.1 – Бизнес-процесс в нотации IDEF0

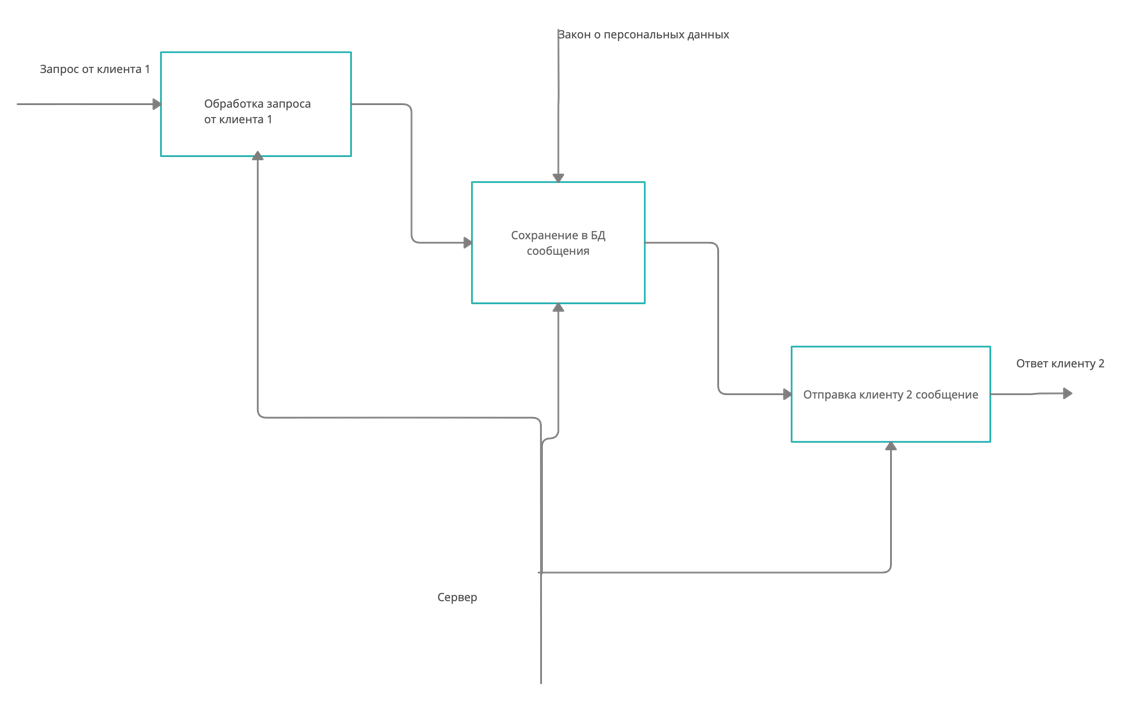


Рисунок 2.2 – Бизнес-процесс в нотации IDEF0

На рис.3 представлена DFD-диаграмма потоков данных

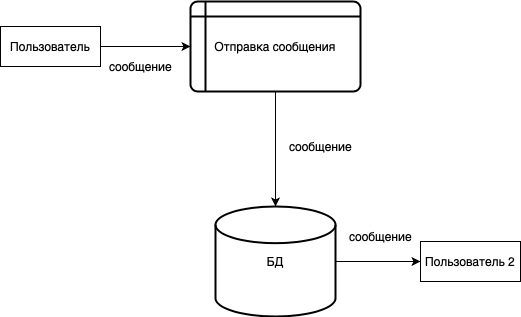


Рисунок 3 – Движение данных в DFD-диаграмме

На рис.4 представлена информационная схема мобильного приложения.

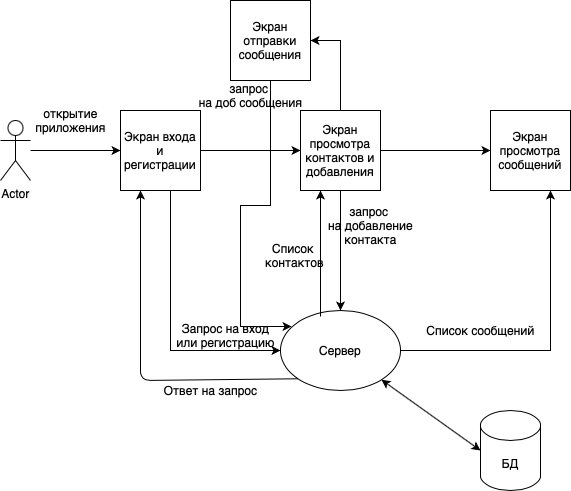


Рисунок 4 – Информационная схема мобильного приложения

В качестве основы была взята Figma, но в последствии остальные части приложения были уже задизайнены в редакторе дизайна Android Studio.

Все разработанные макеты представлены на рис.5 – 8.

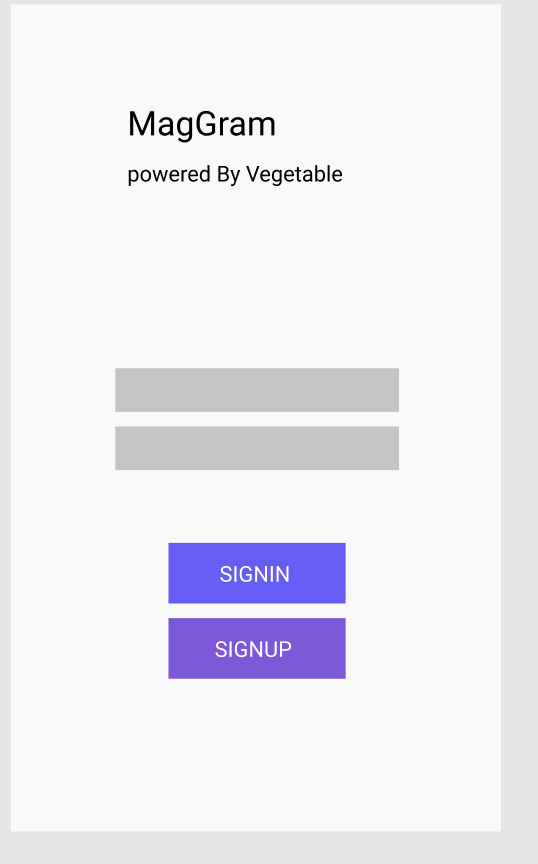


Рисунок 5 – Стартовый экран в Figma

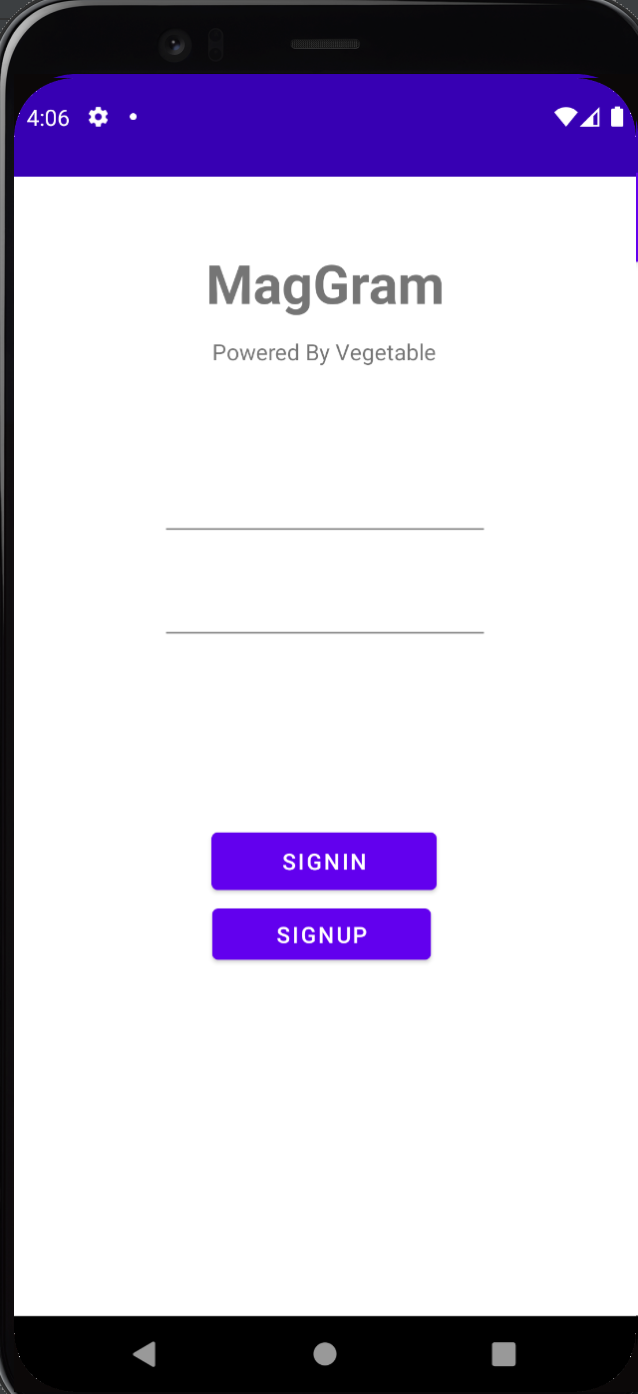


Рисунок 6 – Экран с входом

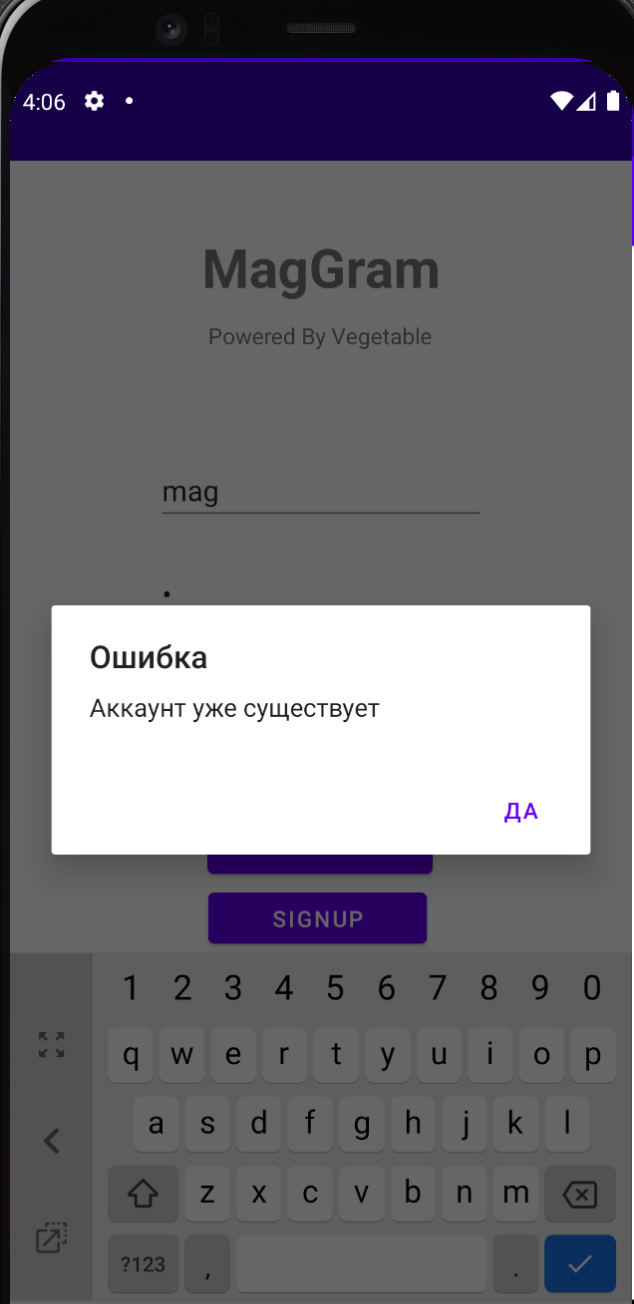


Рисунок 7 – Экран с ошибкой

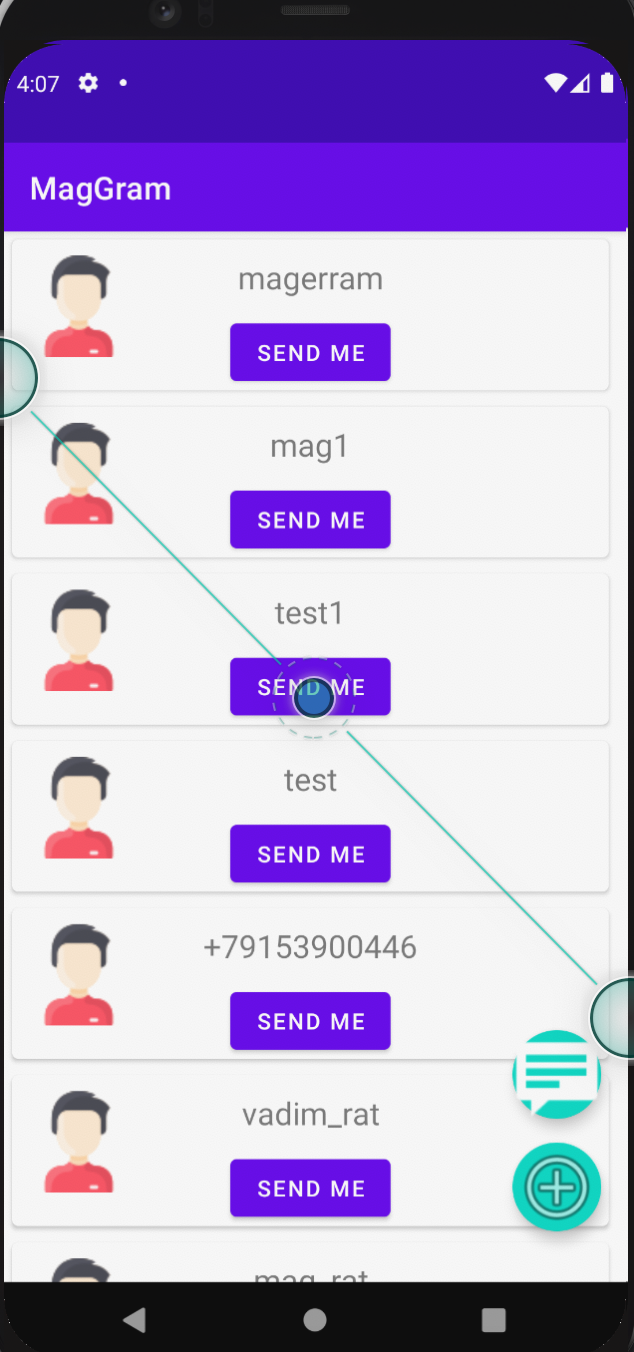


Рисунок 8 – Экран с контактами

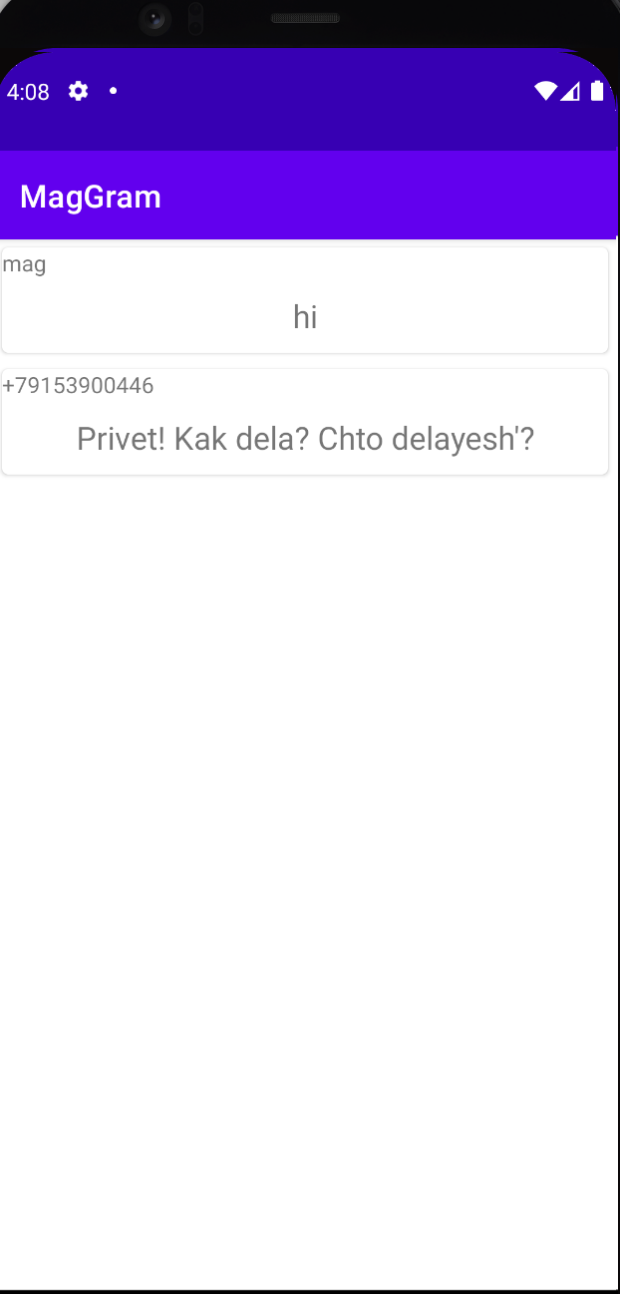


Рисунок 9 – Экран со списком сообщений

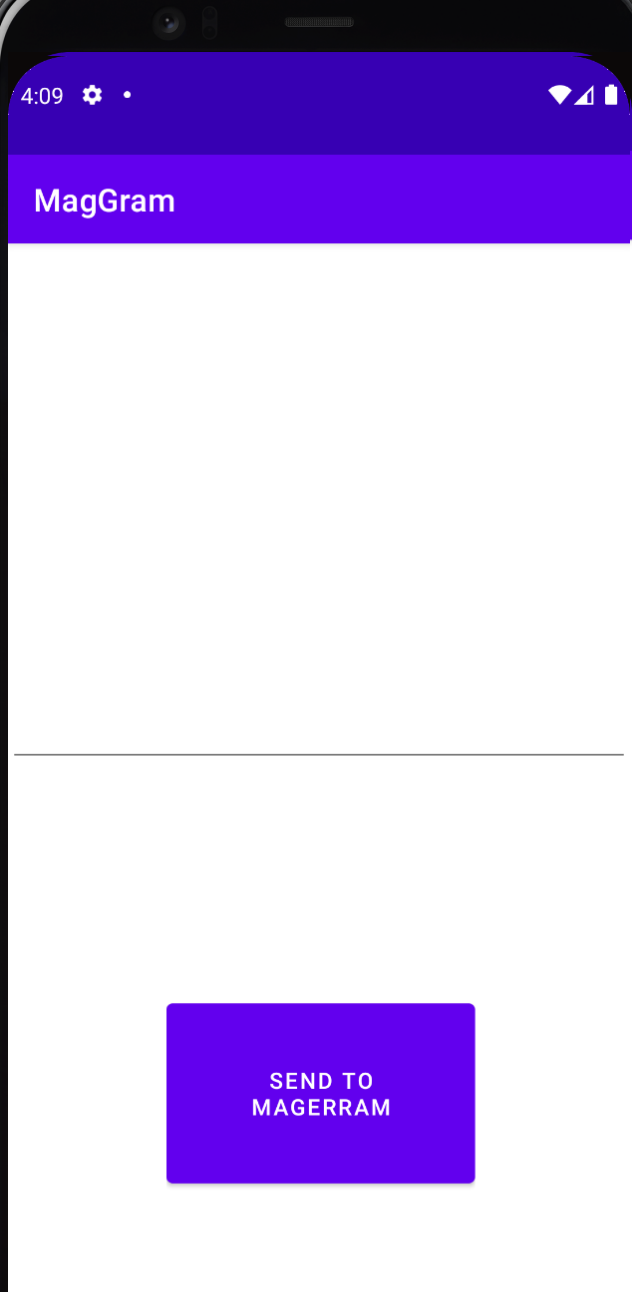


Рисунок 10 – Экран отправкой сообщения

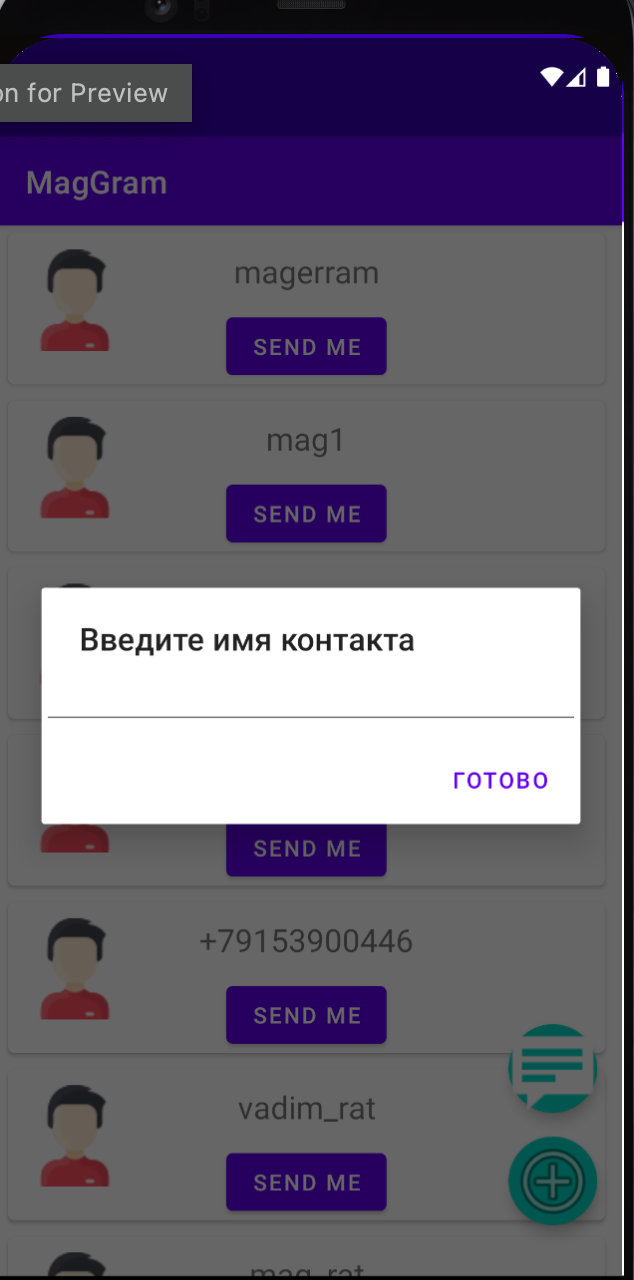


Рисунок 11 – Экран с добавлением контактов

## Раздел 3. Экспериментальная часть

#### **3.1 Тестирование бэкенда**

Бэкенд был протестирован с помощью приложения Postman. Ниже тестирование backend.

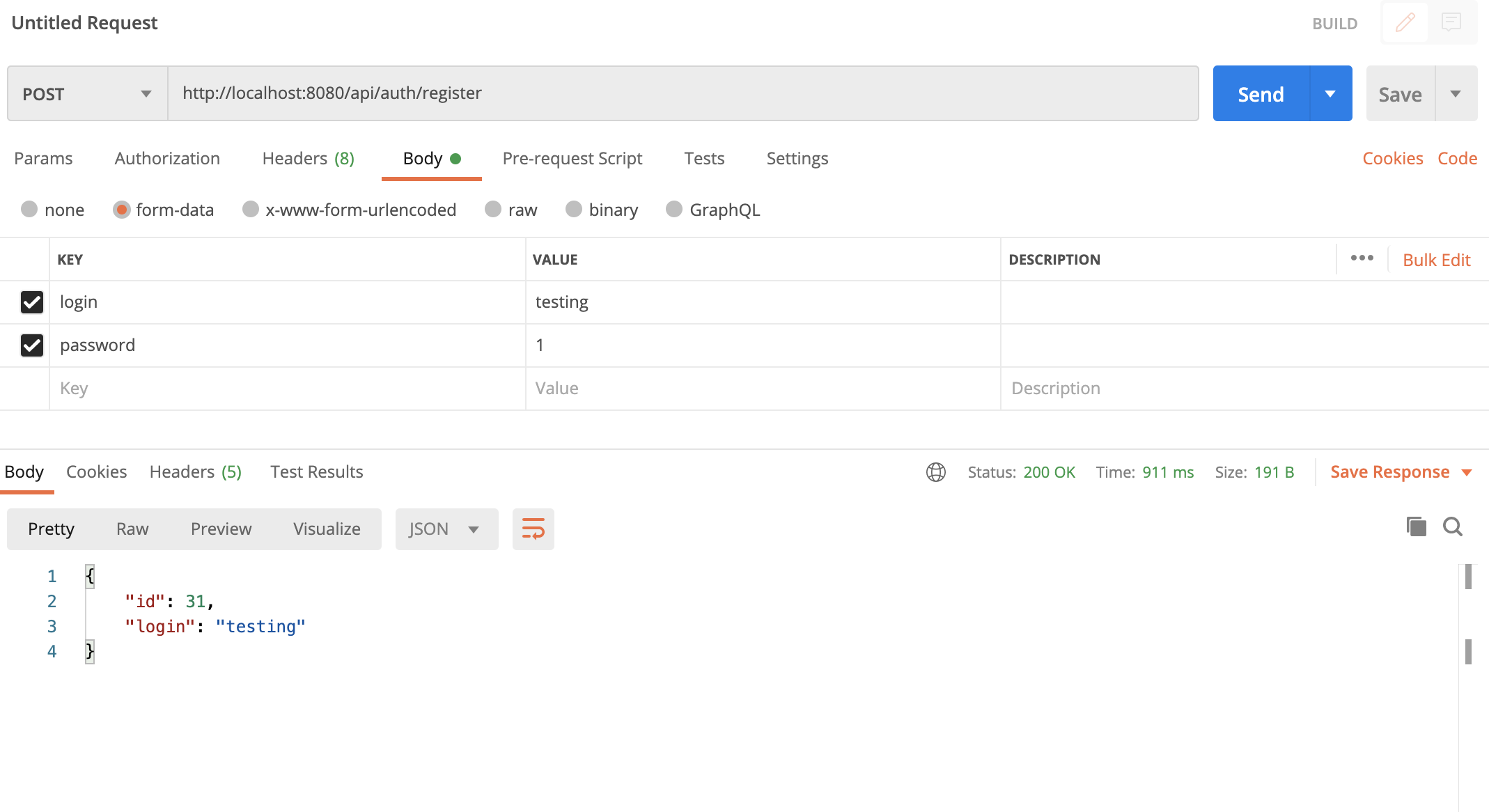


Рисунок 12. Тестирование регистрации

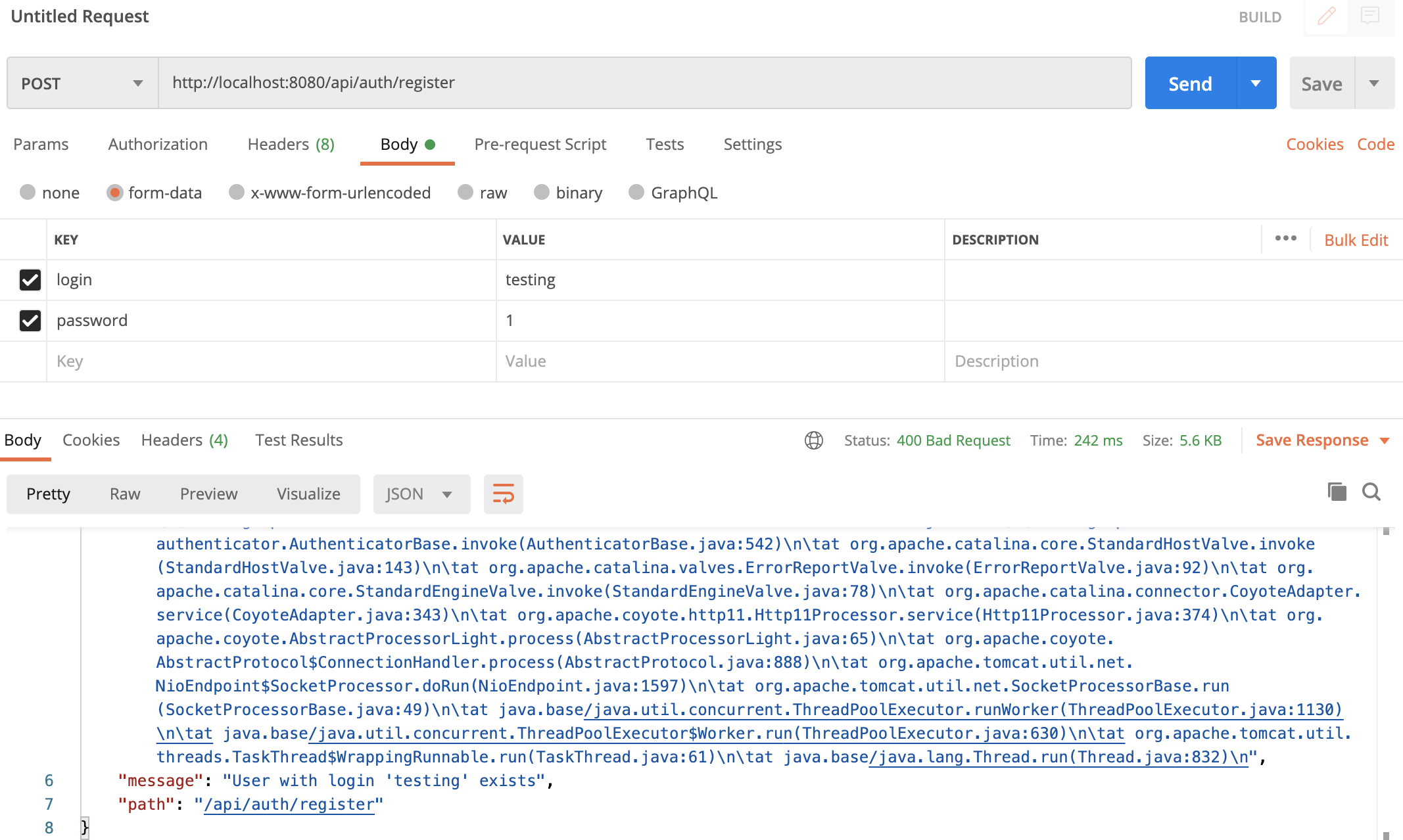


Рисунок 13. Тестирование повторной регистрации

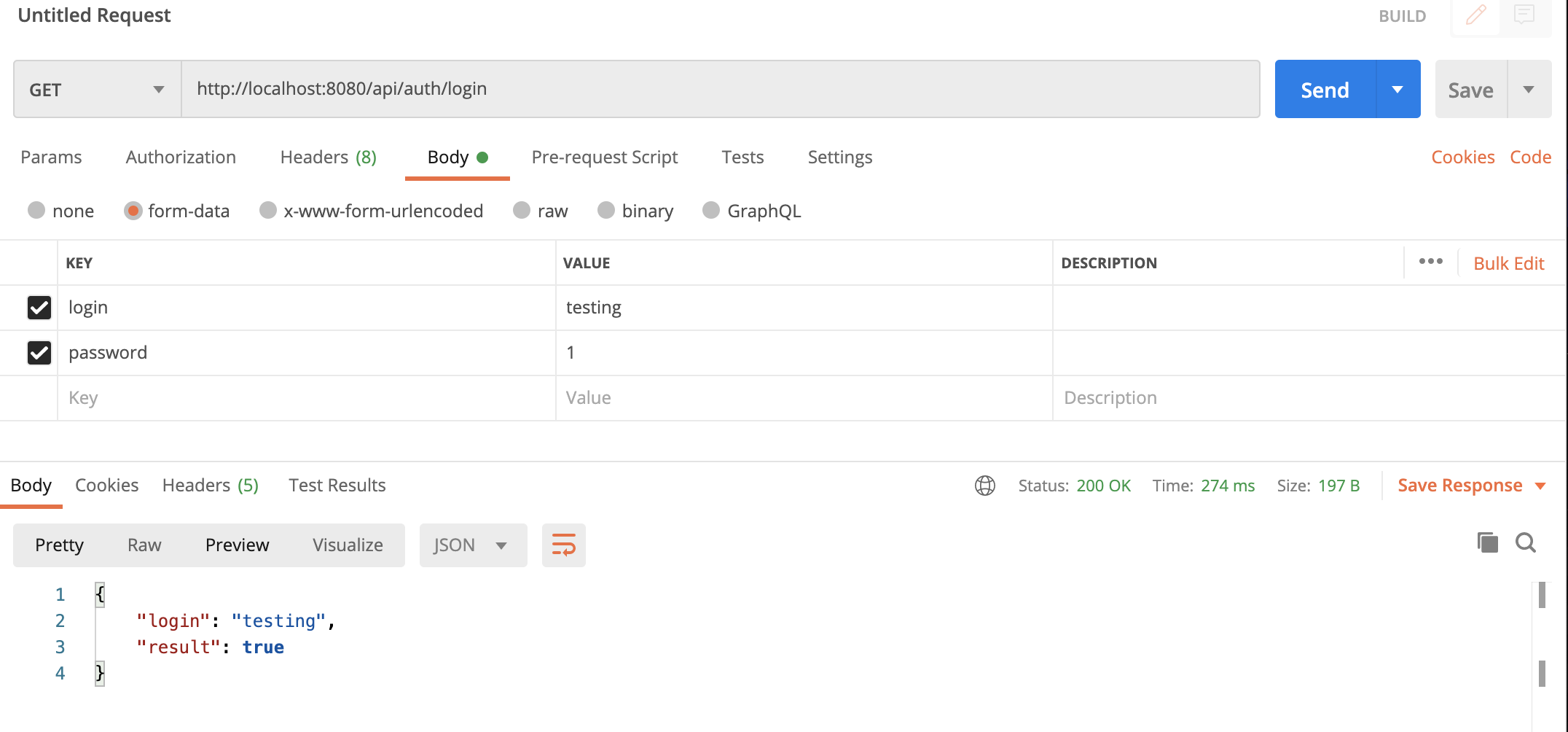


Рисунок 14. Тестирование входа

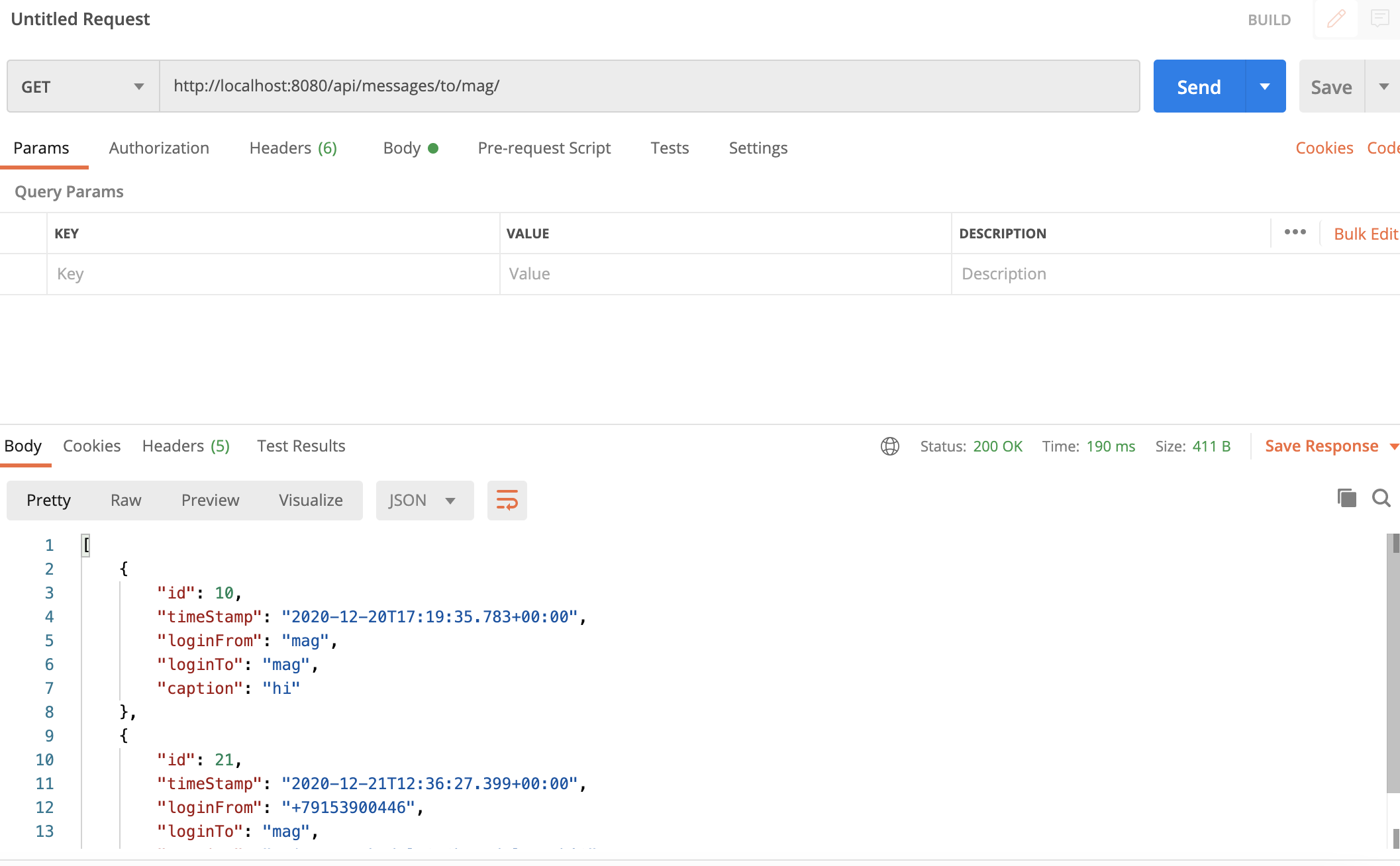


Рисунок 15. Тестирование сообщений

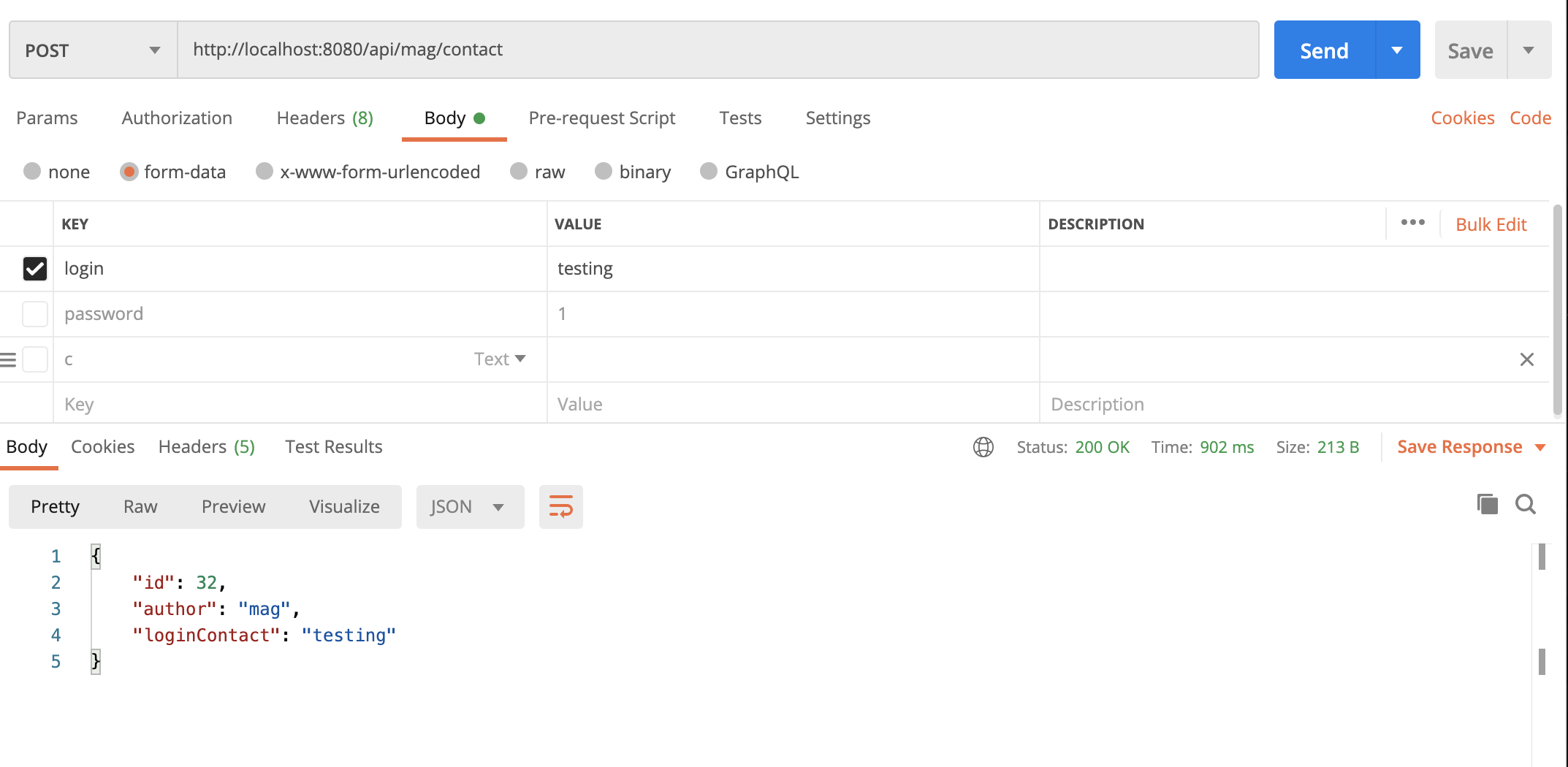


Рисунок 16. Тестирование добавления аккаунта.

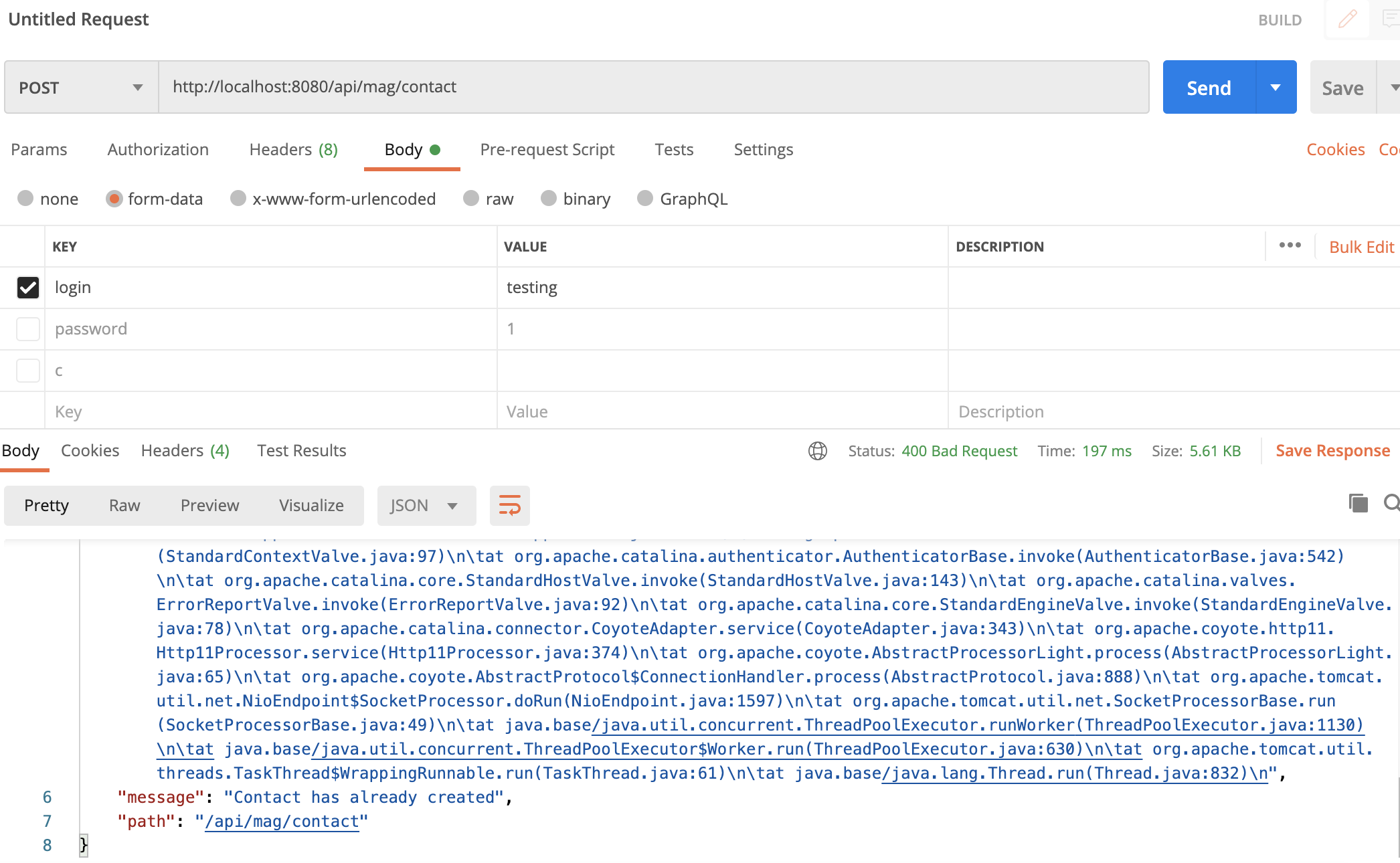


Рисунок 17. Тестирование повторного добавления аккаунта

#### **3.2 Управление экранов**

Для реализации управлении экранов использовали классы, отнаследованные от AppCompatActivity. В данных классах прописывается вся логика управления элементов на экране, ввод данных, проверка на корректность. Также в этих классах отправляются запросы на сервер.

#### **3.3 Связь данных**

Для передачи запроса использовалась библиотека RetroFit [2]. С помощью нее мы отправляли запросы на серверы и получали ответы сразу в объектах. Для этого нужно было настроить сериализацию в параметрах RetroFit.

В качестве библиотеки для десериализации и сериализации использовался GSON [4]. В классах-получателях для корректной работы десериализации настраивались имена полей с помощью аннотаций.

# заключение

В результате выполнения курсовой работы были изучены методы программирования на операционную систему Android. Подтянуты навыки работы с дизайном, рассмотрен такой инструмент как Figma. Были улучшены навыки написания серверной части приложения. В последствии, получены навыки связки клиента и сервера с помощью библиотеки Retrofit.

Также были рассмотрены и использованы такие методологии как DFD и IDEF0. C помощью них были построены диаграммы бизнес-процессов и потоков данных.

В заключении, можно заметить, что в результате работы были подняты общие навыки проектирования систем, где используются как клиент, так и сервер, повышена общая грамотность разработчика. Все заявление функции были реализованы в полной мере и протестированы.

# список используемых источников

1. Spring. Spring Boot [Электронный ресурс]. – Режим доступа: https://spring.io/projects/spring-boot, (свободный).
2. Retrofit [Электронный ресурс]. – Режим доступа: https://square.github.io/retrofit/, (свободный).
3. Android Studio. SDK [Электронный ресурс]. – Режим доступа: https://developer.android.com/studio, (свободный).
4. GitHub. GSON [Электронный ресурс]. – Режим доступа: [https://github.com/google/gson, (свободный)](https://github.com/google/gson,%20(свободный)).
5. GitHub. MagGram [Электронный ресурс]. – Режим доступа: <https://github.com/vgtstptlk/MagGram> , свободный.

**Приложение**

**ContactController.java**

package com.vgtstptlk.gramrestapi.controllers;  
  
import com.vgtstptlk.gramrestapi.domains.Contact;  
import com.vgtstptlk.gramrestapi.exception.ContactExistException;  
import com.vgtstptlk.gramrestapi.exception.UserNotFoundException;  
import com.vgtstptlk.gramrestapi.repository.ContactRepository;  
import com.vgtstptlk.gramrestapi.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/")  
public class ContactController {  
 private final ContactRepository contactRepository;  
 private final UserRepository userRepository;  
  
 @GetMapping("{author}/contact")  
 List<Contact> getContactsByAuthor(@PathVariable String author){  
 return contactRepository.findAllByAuthor(author);  
 }  
  
 @PostMapping("{author}/contact")  
 ResponseEntity<?> addContactToUser(@PathVariable String author, @RequestParam String login){  
  
 userRepository.findByLogin(login).orElseThrow(  
 () -> new UserNotFoundException(login)  
 );  
  
 if (contactRepository.findByAuthorAndLoginContact(author, login).isPresent()){  
 throw new ContactExistException();  
 }  
  
 Contact contact = new Contact(author, login);  
 contactRepository.save(contact);  
  
 return ResponseEntity.*ok*(contact);  
 }  
  
 @Autowired  
 public ContactController(ContactRepository contactRepository, UserRepository userRepository) {  
 this.contactRepository = contactRepository;  
 this.userRepository = userRepository;  
 }  
}

**MessageController.java**

package com.vgtstptlk.gramrestapi.controllers;  
  
import com.vgtstptlk.gramrestapi.domains.Message;  
import com.vgtstptlk.gramrestapi.repository.MessageRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/messages")  
public class MessageController {  
 private final MessageRepository messageRepository;  
  
 @GetMapping  
 List<Message> getMessages(){  
 return messageRepository.findAll();  
 }  
  
 @GetMapping("/from/{loginFrom}")  
 List<Message> getMessageByLoginFrom(@PathVariable String loginFrom){  
 return messageRepository.findAllByLoginFrom(loginFrom);  
 }  
  
 @PostMapping  
 ResponseEntity<?> createMessage(@RequestParam String loginFrom, @RequestParam String loginTo, @RequestParam String caption){  
 Message message = new Message(loginFrom, loginTo, caption);  
 messageRepository.save(message);  
 return ResponseEntity.*ok*(message);  
 }  
  
 @GetMapping("/to/{loginTo}")  
 List<Message> getMessagesByLoginTo(@PathVariable String loginTo){  
 return messageRepository.findAllByLoginTo(loginTo);  
 }  
  
 @Autowired  
 public MessageController(MessageRepository messageRepository) {  
 this.messageRepository = messageRepository;  
 }  
}

**UserController.java**

import org.springframework.http.ResponseEntity;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/api/")  
public class UserController {  
 private final BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();  
 private final UserRepository userRepository;  
  
 @PostMapping("auth/register")  
 ResponseEntity<?> createUser(@RequestParam String login, @RequestParam String password){  
 User tempUser = new User(login, bCryptPasswordEncoder.encode(password));  
 if (userRepository.findByLogin(login).isPresent()){  
 throw new UserExistException(login);  
 }  
 userRepository.save(tempUser);  
 return ResponseEntity.*ok*(tempUser);  
 }  
  
 @GetMapping("auth/login")  
 Auth emulateAuth(@RequestParam String login, @RequestParam String password){  
 Optional<User> optionalUser = userRepository.findByLogin(login);  
 if (optionalUser.isEmpty()){  
 return new Auth(login, false);  
 }  
 User user = optionalUser.get();  
 return bCryptPasswordEncoder.matches(password, user.getPassword()) ? new Auth(login, true) : new Auth(login, false);  
 }  
  
 @Autowired  
 public UserController(UserRepository userRepository) {  
 this.userRepository = userRepository;  
 }  
}

**ChatActivity.java**

public class ChatActivity extends AppCompatActivity {  
  
 private String login;  
 private RecyclerView recyclerView;  
 private RecyclerView.Adapter adapter;  
 private List<Message> messageList = new ArrayList<>();  
 private RecyclerView.LayoutManager layoutManager;  
 private final String URL = "http://10.0.2.2:8080";  
 private final Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl(URL)  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
  
 ServiceApi serviceApi = retrofit.create(ServiceApi.class);  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 login = (String) getIntent().getExtras().get("login");  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_chat*);  
 recyclerView = (RecyclerView) findViewById(R.id.*rec\_chat*);  
 layoutManager = new LinearLayoutManager(this);  
 recyclerView.setLayoutManager(layoutManager);  
  
 Call<List<Message>> listCall = serviceApi.getMessagesByLoginTo(login);  
 listCall.enqueue(new Callback<List<Message>>() {  
 @Override  
 public void onResponse(Call<List<Message>> call, Response<List<Message>> response) {  
 messageList = response.body();  
 adapter = new ChatAdapter(ChatActivity.this, messageList);  
 recyclerView.setAdapter(adapter);  
  
 }  
  
 @Override  
 public void onFailure(Call<List<Message>> call, Throwable t) {  
  
 }  
 });  
 }  
  
 private void showDialog(String title, String message){  
 AlertDialog.Builder a\_builder = new AlertDialog.Builder(ChatActivity.this);  
 a\_builder.setTitle(title);  
 a\_builder.setMessage(message)  
 .setCancelable(false)  
 .setPositiveButton("Да", (dialog, which) -> dialog.cancel());  
 a\_builder.show();  
 }  
  
}

**ChatAdapter.java**

public class ChatAdapter extends RecyclerView.Adapter<ChatAdapter.ViewHolder>{  
 private List<Message> messageList;  
 private Context context;  
 private final String URL = "http://10.0.2.2:8080";  
 private final Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl(URL)  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
  
 ServiceApi serviceApi = retrofit.create(ServiceApi.class);  
  
 public static class ViewHolder extends RecyclerView.ViewHolder{  
 CardView cardView;  
 TextView loginView;  
 TextView messageView;  
  
 public ViewHolder(@NonNull View itemView) {  
 super(itemView);  
 cardView = itemView.findViewById(R.id.*cv2*);  
 loginView = itemView.findViewById(R.id.*textViewLoginMessage*);  
 messageView = itemView.findViewById(R.id.*textViewMessage*);  
 }  
 }  
  
 public ChatAdapter(Context context, List<Message> messageList){  
 this.context = context;  
 this.messageList = messageList;  
  
 }  
  
 @NonNull  
 @Override  
 public ChatAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 LayoutInflater inflater = LayoutInflater.*from*(context);  
 View view = inflater.inflate(R.layout.*message*, parent, false);  
 ChatAdapter.ViewHolder viewHolder = new ChatAdapter.ViewHolder(view);  
 return viewHolder;  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ChatAdapter.ViewHolder holder, int position) {  
 holder.loginView.setText(messageList.get(position).getLoginFrom());  
 holder.messageView.setText(messageList.get(position).getText());  
 }  
  
  
  
 @Override  
 public int getItemCount() {  
 return messageList.size();  
 }  
}

**ContactActivity.java**

package com.example.maggram;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.cardview.widget.CardView;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.annotation.SuppressLint;  
import android.app.AlertDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.EditText;  
  
import com.example.maggram.Domains.Contact;  
import com.example.maggram.Domains.User;  
import com.example.maggram.api.ServiceApi;  
  
import java.io.StringReader;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
import retrofit2.Retrofit;  
import retrofit2.converter.gson.GsonConverterFactory;  
  
public class ContactActivity extends AppCompatActivity {  
  
 private String login;  
 private RecyclerView recyclerView;  
 private RecyclerView.Adapter adapter;  
 private List<Contact> contactList;  
 private RecyclerView.LayoutManager layoutManager;  
 private final String URL = "http://10.0.2.2:8080";  
 private final Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl(URL)  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
  
 ServiceApi serviceApi = retrofit.create(ServiceApi.class);  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_contact*);  
 recyclerView = (RecyclerView) findViewById(R.id.*rec\_cont*);  
 layoutManager = new LinearLayoutManager(this);  
 recyclerView.setLayoutManager(layoutManager);  
 Context context = this;  
  
 Bundle args = getIntent().getExtras();  
 login = (String) args.get("login");  
  
 Call<List<Contact>> listCall = serviceApi.getContacts(login);  
 listCall.enqueue(new Callback<List<Contact>>() {  
 @Override  
 public void onResponse(Call<List<Contact>> call, Response<List<Contact>> response) {  
 contactList = response.body();  
  
 if (contactList != null && contactList.size() == 0){  
 contactList = new ArrayList<>();  
 contactList.add(new Contact(1L, login));  
 }  
  
 adapter = new ContactAdapter(context, contactList, login);  
 recyclerView.setAdapter(adapter);  
 }  
  
 @Override  
 public void onFailure(Call<List<Contact>> call, Throwable t) {  
 showDialog("Ошибка", "Что-то пошло не так");  
 }  
 });  
  
 }  
  
 public void onClickMessage(View view){  
 Intent intent = new Intent(this, ChatActivity.class);  
 intent.putExtra("login", login);  
 startActivity(intent);  
 }  
  
 public void onClickContactButton(View view){  
 AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);  
 View layout = getLayoutInflater().inflate(R.layout.*contact\_add*, null);  
  
 alertDialog.setTitle("Введите имя контакта")  
 .setView(layout)  
 .setPositiveButton("Готово", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Map<String, String> fields = new HashMap<>();  
 EditText edt = (EditText) layout.findViewById(R.id.*edtContAdd*);  
 fields.put("login", edt.getText().toString());  
 Call<Contact> contactCall = serviceApi.addContact(login, fields);  
 contactCall.enqueue(new Callback<Contact>() {  
 @Override  
 public void onResponse(Call<Contact> call, Response<Contact> response) {  
 if (response.code() == 400){  
 showDialog("Ошибка","Аккаунт либо уже добавлен, либо не существует");  
 }  
 else {  
 showDialog("Успешно", "Контакт добавлен");  
 }  
 }  
  
 @Override  
 public void onFailure(Call<Contact> call, Throwable t) {  
 showDialog("Ошибка", "Что-то пошло не так");  
 }  
 });  
 }  
 });  
 alertDialog.create();  
 alertDialog.show();  
 }  
  
 private void showDialog(String title, String message){  
 AlertDialog.Builder a\_builder = new AlertDialog.Builder(ContactActivity.this);  
 a\_builder.setTitle(title);  
 a\_builder.setMessage(message)  
 .setCancelable(false)  
 .setPositiveButton("Да", (dialog, which) -> dialog.cancel());  
 a\_builder.show();  
 }  
}

**ContactAdapter.java**

public class ContactAdapter extends RecyclerView.Adapter<ContactAdapter.ViewHolder> {  
  
 private String loginFrom;  
 private List<Contact> contactList;  
 private Context context;  
 private final String URL = "http://10.0.2.2:8080";  
 private final Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl(URL)  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
  
 ServiceApi serviceApi = retrofit.create(ServiceApi.class);  
  
 public static class ViewHolder extends RecyclerView.ViewHolder{  
 CardView cardView;  
 TextView login;  
 ImageView imageView;  
 Button button;  
 public ViewHolder(@NonNull View itemView) {  
 super(itemView);  
 cardView = itemView.findViewById(R.id.*cv*);  
 login = itemView.findViewById(R.id.*textView3*);  
 imageView = itemView.findViewById(R.id.*imageView2*);  
 button = itemView.findViewById(R.id.*btnMessage*);  
 }  
 }  
  
 public ContactAdapter(Context context, List<Contact> contactList, String loginFrom){  
 this.context = context;  
 this.contactList = contactList;  
 this.loginFrom = loginFrom;  
  
 }  
  
 @NonNull  
 @Override  
 public ContactAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 LayoutInflater inflater = LayoutInflater.*from*(context);  
 View view = inflater.inflate(R.layout.*contact*, parent, false);  
 ViewHolder viewHolder = new ViewHolder(view);  
 return viewHolder;  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ContactAdapter.ViewHolder holder, int position) {  
 String login;  
 login = contactList.get(position).getLogin();  
 holder.login.setText(login);  
 holder.imageView.setImageResource(R.drawable.*man*);  
 holder.button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(ContactAdapter.this.context, SendActivity.class);  
 intent.putExtra("loginTo", login);  
 intent.putExtra("loginFrom", loginFrom);  
 holder.button.getContext().startActivity(intent);  
 }  
 });  
 }  
  
  
  
 @Override  
 public int getItemCount() {  
 return contactList.size();  
 }  
}

**MainActivity.java**

public class MainActivity extends AppCompatActivity {  
  
  
 EditText editTextEmail, editTextPassword;  
 String email, password;  
 private final String URL = "http://10.0.2.2:8080";  
 private final Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl(URL)  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
  
 ServiceApi serviceApi = retrofit.create(ServiceApi.class);  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 getSupportActionBar().hide();  
 }  
  
 public void onClickSignIn(View view){  
 editTextEmail = (EditText) findViewById(R.id.*editTextTextEmailAddress*);  
 editTextPassword = (EditText) findViewById(R.id.*editTextTextPassword*);  
  
 email = editTextEmail.getText().toString();  
 password = editTextPassword.getText().toString();  
  
 Call<Auth> authCall = serviceApi.auth(email, password);  
 Intent intent = new Intent(this, ContactActivity.class);  
 authCall.enqueue(new Callback<Auth>() {  
 @Override  
 public void onResponse(Call<Auth> call, Response<Auth> response) {  
 int status = response.code();  
 Auth auth = response.body();  
 if (auth.getResult()){  
 intent.putExtra("login", email);  
 startActivity(intent);  
  
 }  
 else {  
 showDialog("ЛЕЕЕЕЕЕ НЭ ПРОЙДЕШЬ", "ТЫ ВВЕЛ НЕВЕРНЫЙ ПАРОЛЬ"+response.body().getResult());  
 }  
 }  
  
 @Override  
 public void onFailure(Call<Auth> call, Throwable t) {  
 showDialog("Ошибка",t.getMessage());  
 }  
 });  
  
  
 }  
  
 public void onClickSignUp(View view) throws IOException {  
 editTextEmail = (EditText) findViewById(R.id.*editTextTextEmailAddress*);  
 editTextPassword = (EditText) findViewById(R.id.*editTextTextPassword*);  
  
 email = editTextEmail.getText().toString();  
 password = editTextPassword.getText().toString();  
  
 Map<String, String> fields = new HashMap<>();  
 fields.put("login",email);  
 fields.put("password",password);  
  
 Call<User> userCall = serviceApi.createUser(fields);  
 userCall.enqueue(new Callback<User>() {  
 @Override  
 public void onResponse(Call<User> call, Response<User> response) {  
 int status = response.code();  
 User user = response.body();  
 if (status == 400){  
 showDialog("Ошибка", "Аккаунт уже существует");  
 } else {  
 showDialog("Успешно","Аккаунт зарегистрирован");  
 }  
 }  
  
 @Override  
 public void onFailure(Call<User> call, Throwable t) {  
 showDialog("Ошибка", "Возможно проблема с соедиенением");  
 }  
 });  
  
 }  
  
 private void showDialog(String title, String message){  
 AlertDialog.Builder a\_builder = new AlertDialog.Builder(MainActivity.this);  
 a\_builder.setTitle(title);  
 a\_builder.setMessage(message)  
 .setCancelable(false)  
 .setPositiveButton("Да", (dialog, which) -> dialog.cancel());  
 a\_builder.show();  
 }  
  
  
  
}

**SendActivity.java**

import androidx.appcompat.app.AppCompatActivity;  
  
import android.annotation.SuppressLint;  
import android.app.AlertDialog;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
  
import com.example.maggram.Domains.Message;  
import com.example.maggram.api.ServiceApi;  
  
import java.util.HashMap;  
import java.util.Map;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
import retrofit2.Retrofit;  
import retrofit2.converter.gson.GsonConverterFactory;  
  
public class SendActivity extends AppCompatActivity {  
  
 private String loginTo, loginFrom;  
 private final String URL = "http://10.0.2.2:8080";  
 private final Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl(URL)  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
  
 ServiceApi serviceApi = retrofit.create(ServiceApi.class);  
  
 @SuppressLint("SetTextI18n")  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_send*);  
 loginTo = (String) getIntent().getExtras().get("loginTo");  
 loginFrom = (String) getIntent().getExtras().get("loginFrom");  
 Button button = (Button) findViewById(R.id.*btnSend*);  
 button.setText(button.getText() + " " + loginTo);  
 }  
  
 public void onClick(View view){  
 EditText editText = (EditText) findViewById(R.id.*editTextTextMultiLine2*);  
 Map<String, String> fields = new HashMap<>();  
 fields.put("loginFrom", loginFrom);  
 fields.put("loginTo", loginTo);  
 fields.put("caption", editText.getText().toString());  
 Call<Message> messageCall = serviceApi.createMessage(fields);  
 messageCall.enqueue(new Callback<Message>() {  
 @Override  
 public void onResponse(Call<Message> call, Response<Message> response) {  
 showDialog("Info", "Сообщение отправлено");  
 finish();  
 }  
  
 @Override  
 public void onFailure(Call<Message> call, Throwable t) {  
 showDialog("Ошибка", "Произошла ошибка");  
 }  
 });  
 }  
  
 private void showDialog(String title, String message){  
 AlertDialog.Builder a\_builder = new AlertDialog.Builder(SendActivity.this);  
 a\_builder.setTitle(title);  
 a\_builder.setMessage(message)  
 .setCancelable(false)  
 .setPositiveButton("Да", (dialog, which) -> dialog.cancel());  
 a\_builder.show();  
 }  
}