Mobile programming
Project report
github link: Dauren788/Messenger

Team members:
22MD0205 Seitay Yernar
22MD0114 Abdikadyr Dauren

Used technologies:
Android (Java), Golang - backend API, MSSQL – database.

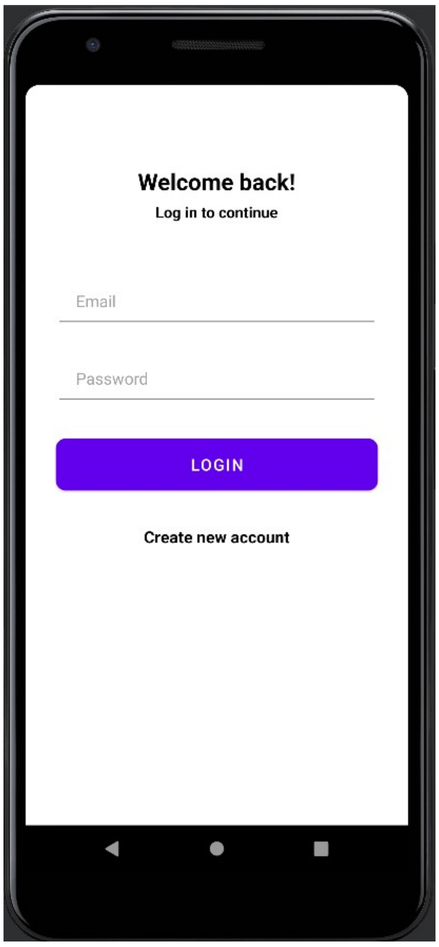
Functionalities (current moment):
- Auth
- Websocket(chat)

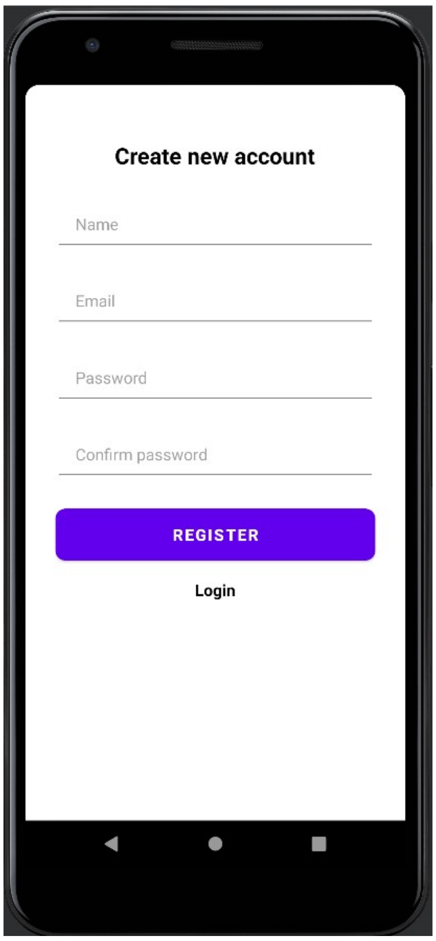
Pages: Login, Registration, Profile, Chats, Chat conversation

Used android components: Fragment, RecyclerView, , Menu-Item and etc.

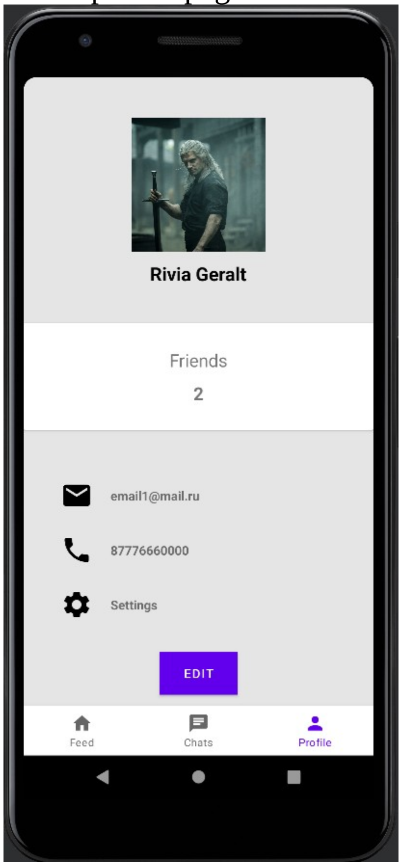
Implementation:
Android

- User auth pages:





- User profile page:


Rivia Geralt

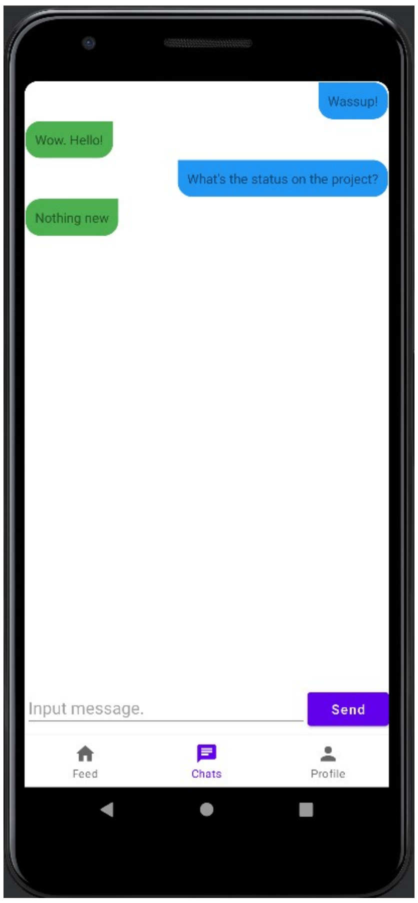
Friends
2

email1@mail.ru

87776660000

Settings

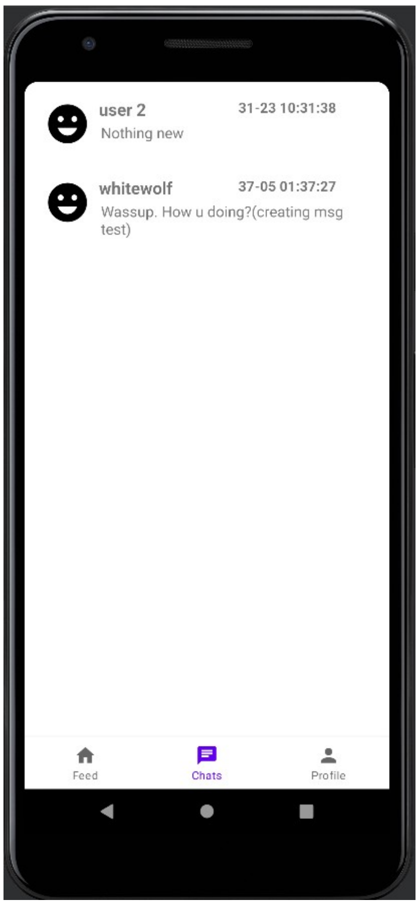
- Chats:


Wassup!

Wow. Hello!

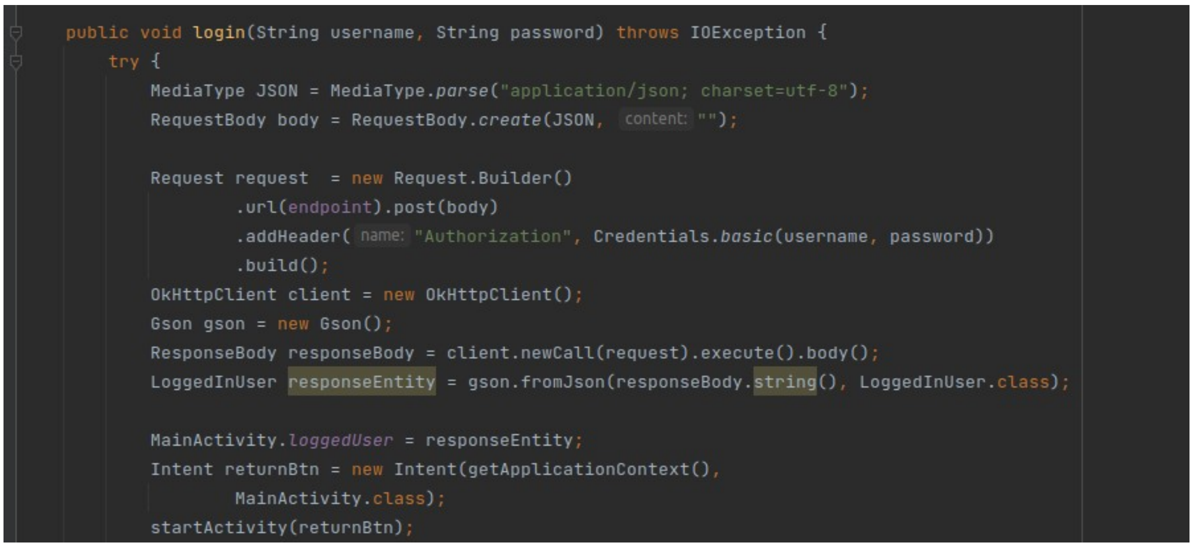
What's the status on the project?

Nothing new


user 2 31-23 10:31:38
Nothing new

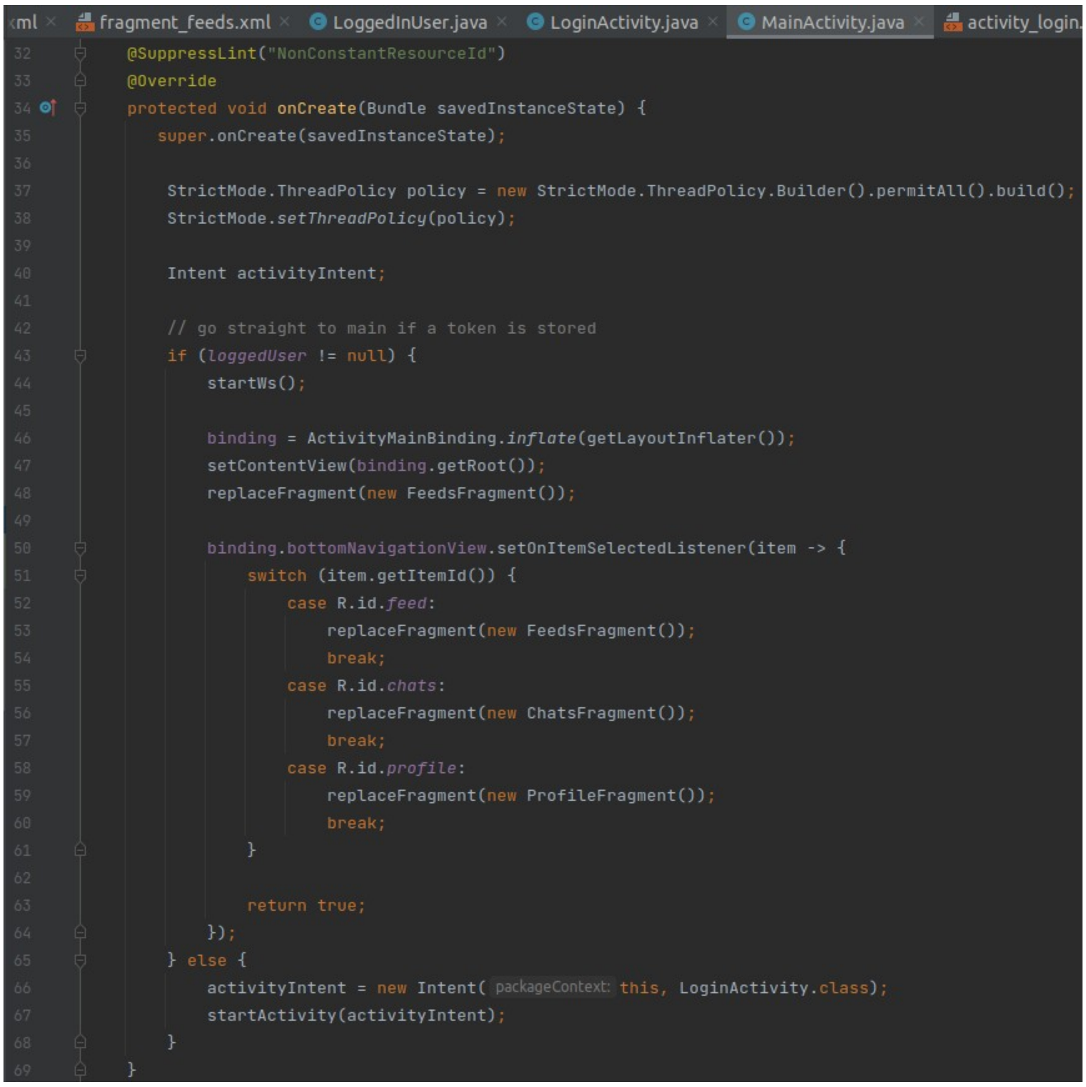
whitewolf 37-05 01:37:27
Wassup. How u doing?(creating msg test)

Auth activity implementation details:
For login and registration activities simple HTTP post request were made with the help of additional libraries like GSON and OkHTTPClient.



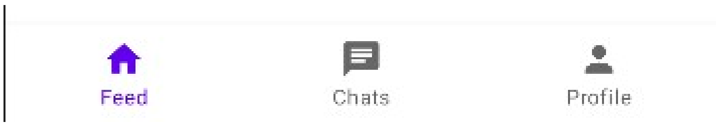
```java
public void login(String username, String password) throws IOException {
    try {
        MediaType JSON = MediaType.parse("application/json; charset=utf-8");
        RequestBody body = RequestBody.create(JSON,  content: "");

        Request request  = new Request.Builder()
                .url(endpoint).post(body)
                .addHeader( name: "Authorization", Credentials.basic(username, password))
                .build();
        OkHttpClient client = new OkHttpClient();
        Gson gson = new Gson();
        ResponseBody responseBody = client.newCall(request).execute().body();
        LoggedInUser responseEntity = gson.fromJson(responseBody.string(), LoggedInUser.class);

        MainActivity.loggedUser = responseEntity;
        Intent returnBtn = new Intent(getApplicationContext(),
                MainActivity.class);
        startActivity(returnBtn);
```

After successful response from the backend API value to global variable in MainActivity is being set. User gains access to app's functionalities.



```java
@SuppressLint("NonConstantResourceId")
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);

    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);

    Intent activityIntent;

    // go straight to main if a token is stored
    if (loggedUser != null) {
        startWs();

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        replaceFragment(new FeedsFragment());

        binding.bottomNavigationView.setOnItemSelectedListener(item -> {
            switch (item.getItemId()) {
                case R.id.feed:
                    replaceFragment(new FeedsFragment());
                    break;
                case R.id.chats:
                    replaceFragment(new ChatsFragment());
                    break;
                case R.id.profile:
                    replaceFragment(new ProfileFragment());
                    break;
            }

            return true;
        });
    } else {
        activityIntent = new Intent( packageContext: this, LoginActivity.class);
        startActivity(activityIntent);
    }
}
```

Navigation buttons implementation details:

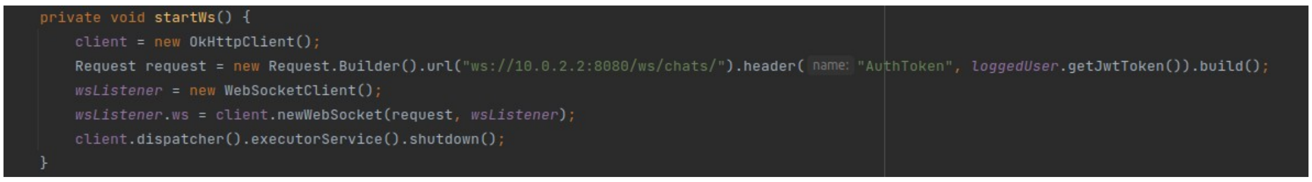


1. Creating items in res/menu/bottom_nav_menu.xml
2. Assigning to <item> objects their titles, icons and ids
3. Make reference in activity_main.xml and setting on item listener of that buttons.

Now every button controls which fragment to load

Chat Activity implementation detail:

- Used connection protocol – Websocket



```
private void startWs() {
    client = new OkHttpClient();
    Request request = new Request.Builder().url("ws://10.0.2.2:8080/ws/chats/").header( name: "AuthToken", loggedUser.getJwtToken()).build();
    wsListener = new WebSocketClient();
    wsListener.ws = client.newWebSocket(request, wsListener);
    client.dispatcher().executorService().shutdown();
}
```

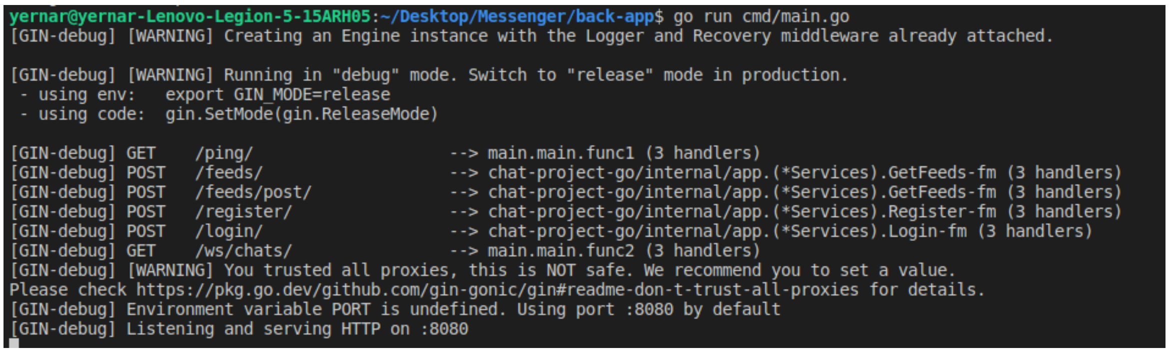
- Websocket connection kept in global variable. So that we could reuse it later. See implementation in java/websocket package.

For now we send following commands through websocket:

Get last messeges from all conversations	"type": 0
Send message to existing conversation	"type": 1, "conversation_id": "?", "text": "?"
Get messages of conversation	"type": 2, "conversation_id": "?"

- Because we have dynamic data it was appropriate to use RecyclerView.

Backend API
Used technology: Golang
Framework: Gin + gorilla



```
yernar@yernar-Lenovo-Legion-5-15ARH05:~/Desktop/Messenger/back-app$ go run cmd/main.go
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /ping/                    --> main.main.func1 (3 handlers)
[GIN-debug] POST   /feeds/                   --> chat-project-go/internal/app.(*Services).GetFeeds-fm (3 handlers)
[GIN-debug] POST   /feeds/post/              --> chat-project-go/internal/app.(*Services).GetFeeds-fm (3 handlers)
[GIN-debug] POST   /register/                --> chat-project-go/internal/app.(*Services).Register-fm (3 handlers)
[GIN-debug] POST   /login/                   --> chat-project-go/internal/app.(*Services).Login-fm (3 handlers)
[GIN-debug] GET    /ws/chats/                --> main.main.func2 (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Environment variable PORT is undefined. Using port :8080 by default
[GIN-debug] Listening and serving HTTP on :8080
```

Postman documentation for API: [Chat project | Postman API Network]