gmessenger

Functions

- registration, authentication and authorization
- one-to-one messages
  - to any registered user
- group messages
  - create public chat for all users
  - create private chat for a particular list of users

Some extra functions that might be implemented

- message status (sent, delivered, read)
- last seen
- Chat members list
- Notifications

Messenger will be implemented using microservices. Communication between microservices will occur via gRPC

Gateway and messaging service API (gets requests from client)

- `GET /` - returns HTML. If there is a cookie with sessionToken, it should send `GET /chats` request. Otherwise, register/login window should be shown. Right after registration/logging in and getting the sessionToken `GET /chats` request should be sent.
- **`POST /register`** - returns cookie with sessionToken

  request body example

```
{
    "username": "someusername",
    "login": "somelogin",
    "password": "somepassword"
}
```

▼ Response codes:

200 - user registered and authenticated, cookie `sessionToken` is set

400 - bad request

409 - login is already taken by another user

500 - internal server error

- **POST /login** - returns cookie with sessionToken

request body example

```
{
    "login": "somelogin",
    "password": "somepassword"
}
```

▼ Response codes:

200 - user logged in and authenticated, cookie `sessionToken` is set

400 - bad request

401 - wrong login or password

500 - internal server error

- `GET /chats` - for authorized users only

returns list of all users and chats, which user was invited in

Chat object:

```
{
    "type": "group", // or dialog
    "id": "groupID", // or receiver login
    "name": "groupName", // or receiver username
    "last_message": "hello",
  "unread": 3
}
```

▼ Response codes:

200 - ok, array of chats was sent

400 - bad request

401 - unauthorized, not token was found in cookie or token doesn`t exists

500 - internal server error

- `GET /dialog/{receiverLogin}` - get list of all messages from the dialog, for authorized users only

  Message object:

```
{
    "id": 24356102983745,
    "sender": "senderLogin",
    "receiver": "receiverLogin",
    "text": "hello",
  "status": "delivered",
    "time": time
}
```

  ▼ Response codes:

  200 - ok, array of chats was sent

  400 - bad request

401 - unauthorized, not token was found in cookie or token doesn`t exists

500 - internal server error

- `ws://{host:port}/chats/ws?token=sometoken` (not implemented)
- `ws://{host:port}/dialog/{receiverLogin}/ws?token=sometoken` - establishes websocket connection

  Server and client send events via websocket. Some possible events:

```
{
    "type": "send_message", // sent from client
    "payload":
    {
        "text": "hello"
    }
}
```

```
{
    "type": "new_message", // sent to client
    "payload":
    {
        "id": 435,
        "text": "hello",
        "sender": "senderLogin",
        "status": "read",
        "time": time
    }
}
```

```
{
    "type": "message_status", // sent to client
    "payload":
    {
```

```
        "id": 435,
        "status": "read",
    }
}
```

```
    Not implemented yet, might be changed
{
    "type": "user_activity", // sent from client
    "payload":
    {
        "user_login": "login1",
    }
}
```

```
    Not implemented yet, might be changed
{
    "type": "user_status", // sent from server
    "payload":
    {
        "user_login": "login1",
        "is_online": true,
        "last_seen": time
    }
}
```

- `GET /group/{groupId}` - get list of all messages from the dialog, for authorized users only

  Message object:

```
{
    "id": 24356102983745,
    "sender": "senderLogin",
```

```
    "text": "hello",
  "status": "delivered",
    "time": time
}
```

▼ Response codes:

200 - ok, array of chats was sent

400 - bad request

401 - unauthorized, not token was found in cookie or token doesn`t exists

403 - forbidden, user is not a member of the group

404 - group not found or doesn`t exist

500 - internal server error

- `ws://{host:port}/group/{groupId}/ws?token=sometoken` - establishes websocket connection

  Server and client send events via websocket. Some possible events:

  Used same events as in dialog

Microservices

Communication between microservices will be implemented via gRPC

Group service

- Proto files can be found in /proto folder

To be implemented...

Last seen service

Tracks users’ activity

- POST /user/active/update
- GET /user/active/

Relay service

Stores unsent messages

- POST /messages/new

Body:

```
{
    “sender_userID”: “SomeId”,
    “receiver_userID”: “someotherid”,
    “time”: “23:12:2023 21:30”,
    “text”: “some text”
}
```

- POST /messages/unsent

```
{
    “receiver_userID”: “someid”
}
```