SOCKET CHATTING

김지윤, 연창현

주제

자체 회원가입을 이용한 1대1과 단체 채팅 시스템

개발 환경

개발 도구

- Eclipse

개발 언어

- Java

DBMS

- Oracle

협업 툴

- GitHub

주요 기능

회원 관리

- 로그인 정보를 저장 및 관리하는 세션 객체를 만들어 인증 상태를 유지

1대1 채팅

- 사용자가 특정 상대방을 초대하여 개인 채팅방을 생성
- 생성된 채팅방은 메세지 송수신, 방 나가기, 방 삭제 기능

그룹 채팅

- 사용자가 여러 명(최대 5명)을 초대하여 그룹 채팅방 생성
- 생성된 채팅방은 다수 인원이 실시간 메세지 송수신
- 사용자는 채팅방에서 개별적으로 나갈 수 있는 기능, 방 참가자가 모두 나가면 방 삭제

공통

- 채팅 메세지는 DB에 저장되어 1시간 내 대화 이력 조회
- 방 삭제 시, 메세지 기록 및 방 자체가 자동 삭제

역할

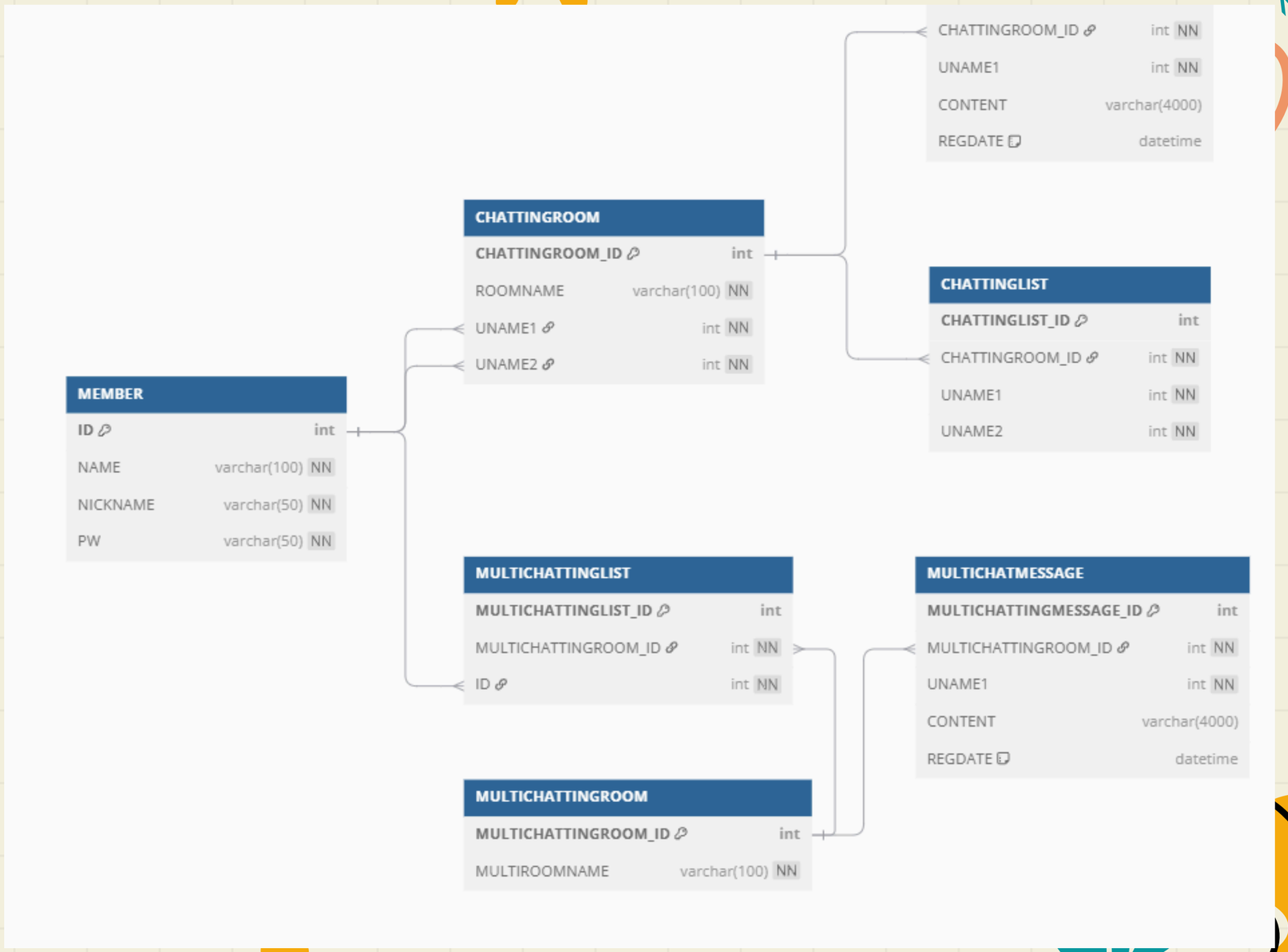
창현

- JDBC를 사용하여 쿼리문 작성 및 DB연동

지윤

- 자체 회원가입 구현 (CRUD)
- Server, Client 구현
- Thread와 Socket을 사용하여 실시간 채팅 기능 구현

DB 설계


CHATTINGROOM_ID int NN
UNAME1 int NN
CONTENT varchar(4000)
REGDATE datetime
CHATTINGROOM
CHATTINGROOM_ID int
ROOMNAME varchar(100) NN
UNAME1 int NN
UNAME2 int NN
CHATTINGLIST
CHATTINGLIST_ID int
CHATTINGROOM_ID int NN
UNAME1 int NN
UNAME2 int NN
MEMBER
ID int
NAME varchar(100) NN
NICKNAME varchar(50) NN
PW varchar(50) NN
MULTICHATTINGLIST
MULTICHATTINGLIST_ID int
MULTICHATTINGROOM_ID int NN
ID int NN
MULTICHATMESSAGE
MULTICHATTINGMESSAGE_ID int
MULTICHATTINGROOM_ID int NN
UNAME1 int NN
CONTENT varchar(4000)
REGDATE datetime
MULTICHATTINGROOM
MULTICHATTINGROOM_ID int
MULTIROOMNAME varchar(100) NN

서버

- 채팅 기능(3, 4, 5번) 선택 시 서버를 최초 1회 실행해 프로그램 종료 전까지 유지

클라이언트

- 채팅방 기능마다 새로운 클라이언트를 실행해 통신

```
while (true) {
    try {

        System.out.println("----- 메인메뉴 -----");
        System.out.println("1. 회원가입");
        System.out.println("2. 로그인");
        System.out.println("3. 채팅방 보기");
        System.out.println("4. 개인 채팅방 생성");
        System.out.println("5. 단체 채팅방 생성");
        System.out.println("6. 회원정보 수정"); // -> 탈퇴
        System.out.println("7. 로그아웃");
        System.out.println("8. 종료");

        System.out.print(">> 원하시는 번호를 입력해주세요 : \n");
        num = sc.nextInt();
        sc.nextLine();

        switch (num) {
        // 회원가입
        case 1:
            userService.signUp();
            break;
        // 로그인
        case 2:
            userService.signIn();
            break;
        // 채팅방 보기
        case 3:
            openServer();
            chattingService.showAndJoinChatRooms(); // 채팅방 목록 보기 → 입장
            break;
        // 개인 채팅방 생성
        case 4:
            openServer();
            chattingService.makeOneToOneChatting(); // 유저 초대 → 클라 실행
            break;
        // 단체 채팅방 생성
        case 5:
            openServer();
            chattingService.makeGroupChatting();
            break;
        // 회원정보 수정
        case 6:
            userService.UpdateUser();
            break;
        // 로그아웃
        case 7:
            userService.logOut();
            break;
        case 8:
            System.out.println("종료되었습니다.");
            System.exit(0);
        default:
            break;
        }
    } catch (Exception e) {
        System.out.println("잘못된 번호를 입력하였습니다.");
        sc.nextLine();
    }
}
```

```
//서버가 열려있지 않으면 처음 실행
//실행 후 다른 기능을 사용할 때 서버가 열려있으면 기존 서버를 그대로 사용
//회원 관리엔 서버 필요 X, openServer에서 검사하기 때문에 3,4,5에서 호출
public static void openServer() {
    if (serverThread == null || !serverThread.isAlive()) {
        serverThread = new Thread(() -> {
            ServerMain.startServer(); // 서버 실행
        });
        serverThread.start();
        // 서버는 join하지 않고 그냥 실행 (클라이언트가 join)
    }
}
```

```
// 클라이언트 스레드 실행
Thread clientThread = new Thread(() -> {
    Client client = new Client();
    client.start();
});
clientThread.start();

try {
    clientThread.join(); // 클라이언트 끝날 때까지 대기
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

이중 MAP 구조

- Map<String, Map<Integer, Integer>>

  구조를 사용하여

  {방 이름 → (번지 번호, 사용자 ID)}

  형태로 유저들을 체계적으로

  저장하고 관리

```java
//최대인원 5명
String inviteUserNickName[] = new String[5];
//채팅방이름, 번지, userId
Map<String, Map<Integer, Integer>> groupUserInfo = new HashMap<String, Map<Integer, Integer>>();
//번지, userId
Map<Integer, Integer> userInfo = new HashMap<Integer, Integer>();
Integer inviteUserKey[] = new Integer[5];

//0번 인덱스는 무조건 그룹채팅방 만든 유저정보
userInfo.put(1,loginUser.getKey());
groupUserInfo.put(roomName, userInfo);

//초대할 유저 닉네임 입력
for (int i = 2; i <= inviteUserNickName.length; i++) {
    System.out.print("초대할 유저 닉네임 : ");
    inviteUserNickName[i] = strSc.nextLine();

    if(inviteUserNickName[i].equals("끝")) {
        if(inviteUserNickName[2]==null) {
            System.out.println("최소 1명은 초대해야합니다.");
        }
        break;
    }

    if(!userDAO.userExistByNickName(inviteUserNickName[i])) {
        System.out.println("유저가 존재하지 않습니다.");
        i--;
        continue;
    }

    inviteUserKey[i]=userDAO.findUserIdByName(inviteUserNickName[i]);
    //초대한 사용자 key값을 뽑아서 Map에 저장
    userInfo.put(i,inviteUserKey[i]);
    groupUserInfo.put(roomName, userInfo);
}
```

THANK YOU