ENTITY RELATIONSHIP DIAGRAM (ERD)

PROJECT NAME: CHAT APP

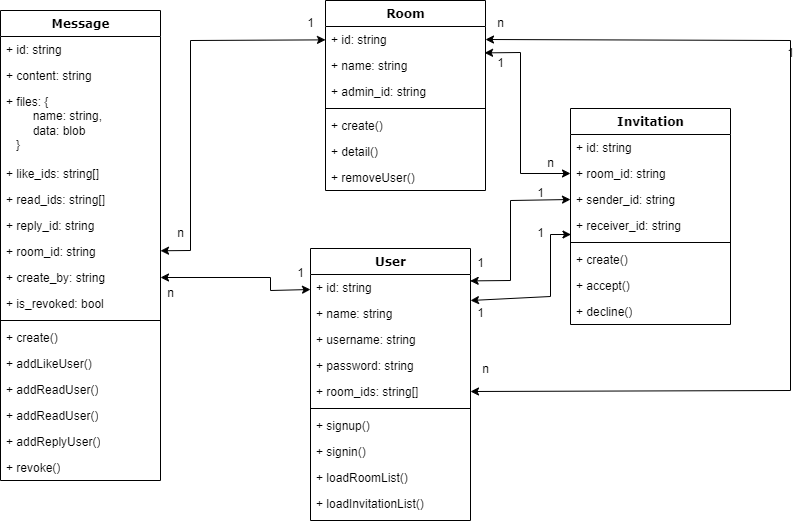


Diagram Explaination:

The above diagram is usually used in SQL database. However, I created it to let you imagine how data are arranged if SQL database is used. In sum, the fields of each entity are exactly the same at the MongoDB document.

Message Entity:

- Id: a built in ObjectID - uuid bson type that auto generated and indexed of MongoDB. I used it as primary key for every documents.

- content: the content of message in utf8 characters

- files: array of object that store model {name, data}. the name is file name of the file, and the data is blob type data of the file.

- like\_ids: the id of users who send a like to this message

- read\_ids: the id of users who read the message.

- reply\_id: the id of message that is the answer of this message

- room\_id: the id of room where this message is lie on

- create\_by: the id of user who created this message

- is\_revoked: the status of message that represents the absent or available in chat box

Room Entity:

- id: same at message id

- name: the name of the room

- admin\_id: the id of user who created this room

User Entity:

- id: same as message id

- name: the full name of the user

- username: the account name of the user to login

- password: a hash of password of the user account

- room\_ids: list of room id which this user can see and join to

Invitation Entity:

- id: same as message id

- room\_id: id of invited room that the sender requests a user to join in

- sender\_id: id of user who sent this request

- receiver\_id: id of user who receive this invitation.

REAL APPLICATION ENTITY DESIGN:

Message Entity:

id?: string;

  content?: string;

  file?: Uint8Array[];

  fileName?: string;

  senderId?: string | ObjectId;

  readerList?: UserModel[] | string[] | ObjectId[];

  likeList?: UserModel[] | string[] | ObjectId[];

  heartList?: UserModel[] | string[] | ObjectId[];

  createAt?: number;

  isRemoved?: boolean;

  replyToMessage?: string | ObjectId;

  roomId?: string | ObjectId;

Compare to first design model, the following are changed:

- The field “files” is converted to field “file” and “file name”. This is because I made it simple to develop in limited time. However in real world, it should be handled as first design to look more natural.

- Field “createAt” is added to mark the time of message created. This is very important function to show the message creating time.

Room Entity:

RoomModel {

  id?: string;

  name?: string;

  adminID?: string | ObjectId;

}

User Entity:

UserModel {

  id?: string;

  email?: string;

  name?: string;

  password?: string;

  roomList?: RoomModel[] | string[] | ObjectId[];

  roomInvitedList?: RoomModel[] | string[] | ObjectId[];

  avatar\_url?: string;

}

The field “roomInvitedList” is added and there is no invitation document. To be adapted with noSQL database, the list of room id (“roomInvitedList”) will be used to represent the invitation list. And to be more convenient, I used multiple types for this field because I will utilize it in many places of application (For example, to save list of room ids to database or to fetch database with embedded documents).