同濟大學
TONGJI UNIVERSITY

怡运动
运动爱好者的运动与社交平台

怡运动——运动爱好者的运动与社交平台

详细设计规约

2251093 冯伟航

2254300 王捷

2252721 韩坤甫

目录

1. 引言

1.1. 编写目的

本文档是用于细化“怡运动——运动爱好者的运动与社交平台系统”的详细设计的，我们将通过子系统设计、接口设计、类详细设计三个角度规范平台开发中的编码方向、不同微服务之间的接口调用以及方法设计等。

1.2. 背景与依据

“怡运动——运动爱好者的运动与社交平台”旨在通过运动社交功能，帮助用户找到合适的运动伙伴，为体育运动爱好者组局，提升他们的运动积极性和参与度。同时，系统还提供集中的运动场地信息平台，用户可以轻松地查找和预约各类运动场地，省去繁琐的预约过程，节省大量的时间和精力，提升整体的预约体验。

“怡运动”主要面向以下两类用户群体：一是希望通过运动结交伙伴并找到合适运动场地进行锻炼的普通用户；二是需要定期或不定期预约场地以进行训练、比赛等团体运动的运动俱乐部和社团。平台通过集中管理和高效便捷的功能，为这些用户提供了专业而全面的服务，致力于成为体育运动爱好者的最佳助手。

1.3. 参考资料

Roger S.Pressman&Bruce R.Maxim.软件工程—实践者的研究方法（第八版）.郑仁杰译 .北京:机械工业出版社.2016.12

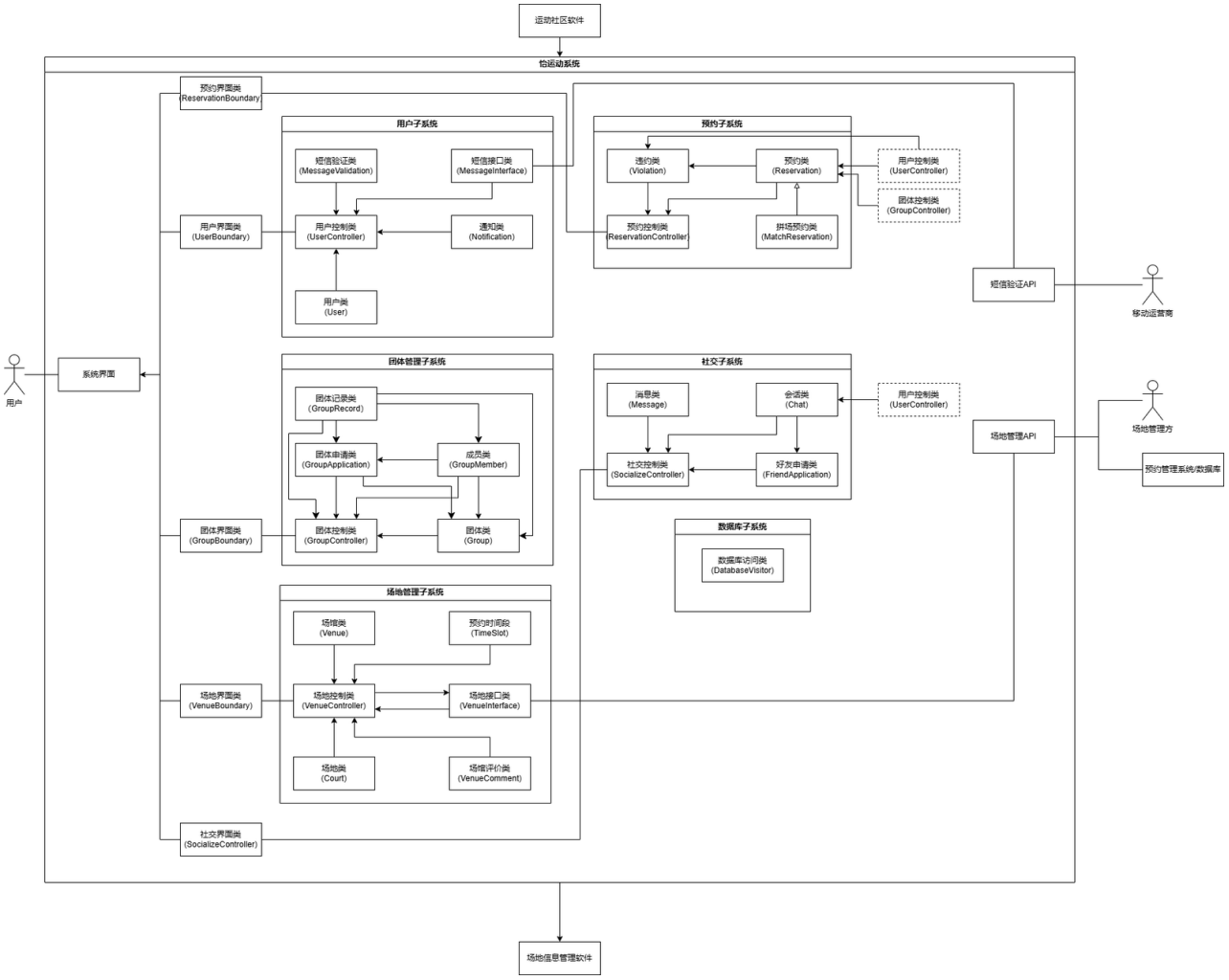
《SpringBoot 更好的开发》

《HTML5+CSS3 从入门到精通》

《更好的软件架构，更好的设计》

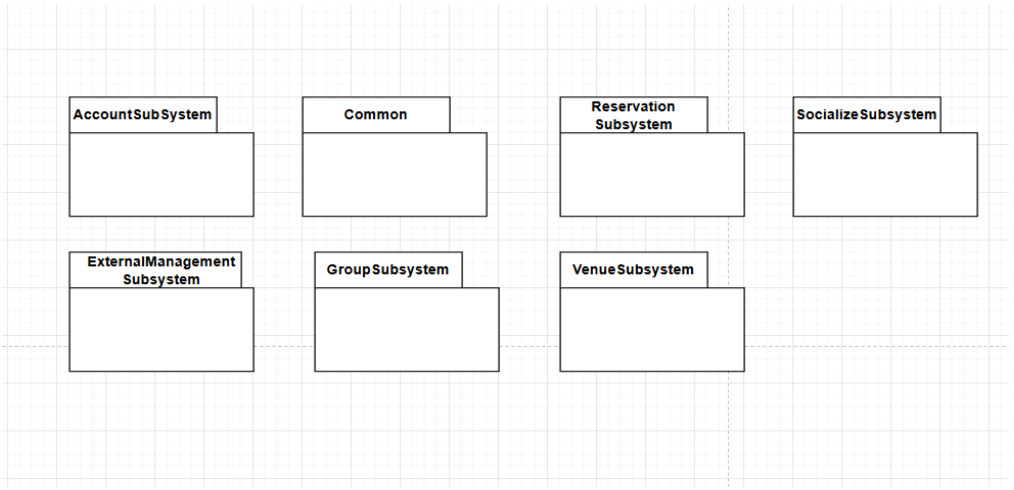
《Design Pattern》

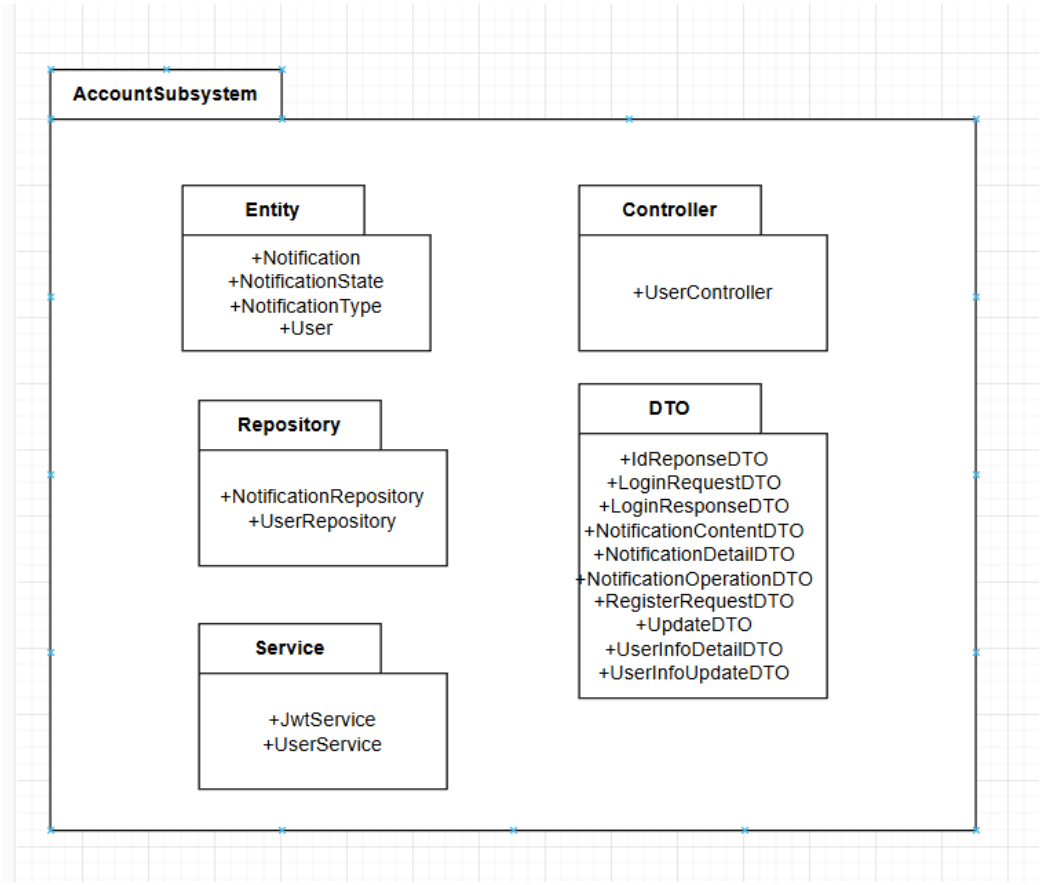
2. 系统软件体系结构



3. 子系统设计

本系统整体架构采用模块化设计，包含 6 个独立的子系统，每个子系统划分为明确的功能模块，分别封装在单独的软件包中。每个子系统都遵循分层设计，包含 Controller 层负责处理接口请求、Entity 层定义数据模型、DTO 层用于数据传输、Repository 层实现数据库操作、Service 层处理业务逻辑。系统还提供一个公共模块 common，用于共享全局配置、通用的 DTO 对象以及安全模块 security，以实现跨子系统的一致性与复用性。这样的设计确保了系统的模块独立性、可维护性和扩展性，为业务需求的快速响应奠定了基础。

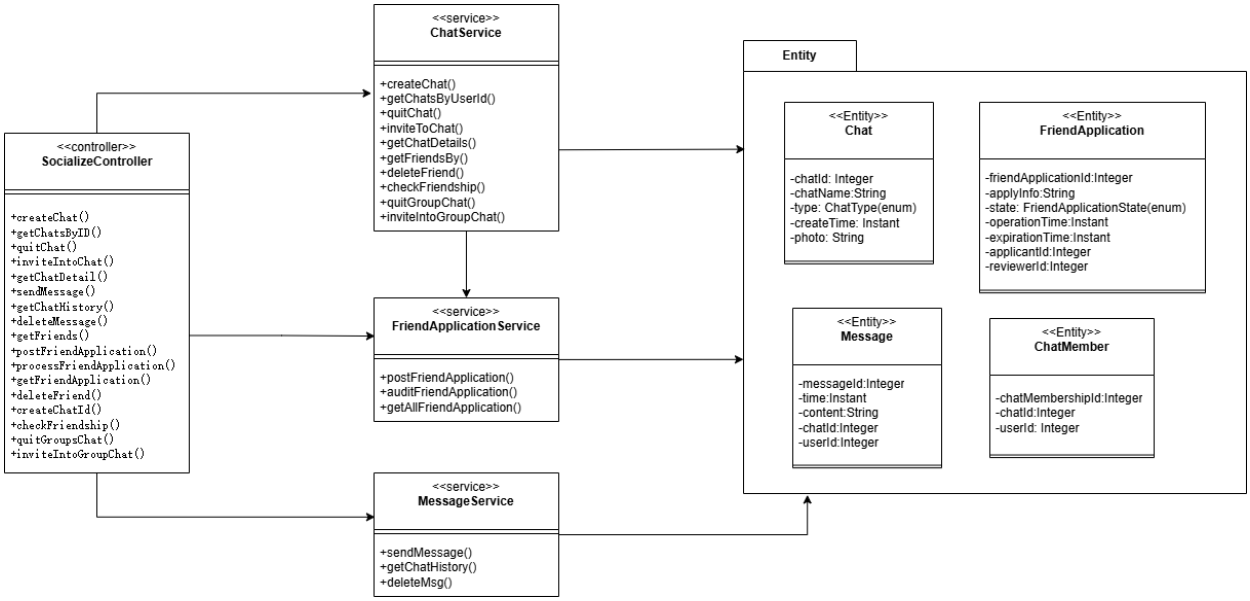

AccountSubSystem
Common
Reservation Subsystem
SocializeSubsystem
ExternalManagement Subsystem
GroupSubsystem
VenueSubsystem


AccountSubsystem
Entity
+Notification
+NotificationState
+NotificationType
+User
Controller
+UserController
Repository
+NotificationRepository
+UserRepository
DTO
+IdReponseDTO
+LoginRequestDTO
+LoginResponseDTO
+NotificationContentDTO
+NotificationDetailDTO
+NotificationOperationDTO
+RegisterRequestDTO
+UpdateDTO
+UserInfoDetailDTO
+UserInfoUpdateDTO
Service
+JwtService
+UserService

4. 类设计

4.1. 社交管理子系统

负责群聊的创建和管理，用户加入群聊，好友的申请和管理。



4.1.1. ChatController 类

4.1.1.1. createChat

功能:

创建一个新的聊天，可能是群聊或朋友聊天。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

chat: ChatDTO 对象，包含聊天相关信息（如聊天名称、参与者等）。

输出项:

成功时返回创建的聊天信息。

失败时返回错误信息。

实现流程:

将用户 ID 设为聊天的发起者。

调用 chatService.createChat()方法创建聊天。

返回成功或失败的响应。

限制条件及出错处理:

如果创建聊天时发生异常，返回 500 错误并传递错误信息。

4.1.1.2. getChatsByID

功能:

获取某个用户的聊天记录列表。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

输出项:

返回该用户参与的聊天列表。

实现流程:

调用 chatService.getChatsByUserId()方法获取该用户的聊天列表。

返回聊天列表或错误信息。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.3. quitChat

功能:

某用户退出某个群聊。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

chatId: 要退出的聊天 ID。

输出项:

成功时返回退出成功的消息。

失败时返回错误信息。

实现流程:

调用 chatService.quitChat()方法退出群聊。

返回成功或失败的响应。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.4. inviteIntoChat

功能:

邀请用户加入群聊。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

inviteDto: InviteDTO 对象，包含邀请信息。

输出项:

成功时返回邀请成功的消息。

失败时返回错误信息。

实现流程:

设置邀请者的用户 ID。

调用 chatService.inviteToChat()方法发送邀请。

返回成功或失败的响应。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.5. getChatDetail

功能:

获取群聊的详细信息。

输入项:

chatId: 群聊的 ID。

输出项:

返回群聊的详细信息。

失败时返回错误信息。

实现流程:

调用 chatService.getChatDetails()方法获取群聊详情。

返回群聊的详细信息或错误信息。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.6. sendMessage

功能:

发送消息到某个聊天。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

messageDto: MessageDTO 对象，包含消息内容。

输出项:

返回发送的消息信息。

失败时返回错误信息。

实现流程:

设置发送者的用户 ID。

调用 messageService.sendMessage()方法发送消息。

返回消息发送成功或失败的响应。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.7. getChatHistory

功能:

获取某个聊天的历史消息。

输入项:

chatId: 聊天 ID。

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

输出项:

返回聊天的历史消息。

失败时返回错误信息。

实现流程:

调用 messageService.getChatHistory()方法获取历史消息。

返回历史消息或错误信息。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.8. deleteMessage

功能:

删除指定的消息。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

messageId: 要删除的消息 ID。

输出项:

返回删除成功的消息。

失败时返回错误信息。

实现流程:

调用 messageService.deleteMsg()方法删除指定消息。

返回删除成功或失败的响应。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.9. getFriends

功能:

获取当前用户的好友列表。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

输出项:

返回当前用户的好友列表。

失败时返回错误信息。

实现流程:

调用 chatService.getFriendsBy()方法获取好友列表。

返回好友列表或错误信息。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.10. postFriendApplication

功能:

发送好友申请。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

friendApplication: FriendApplicationDTO 对象，包含申请信息。

输出项:

返回好友申请已发送的消息。

失败时返回错误信息。

实现流程:

设置申请者的用户 ID。

调用 friendApplicationService.postFriendApplication()方法发送申请。

返回成功或失败的响应。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.11. processFriendApplication

功能:

审核处理好友申请。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

auditResultDTO: AuditResultDTO 对象，包含审核结果信息。

输出项:

返回好友申请处理成功的消息。

失败时返回错误信息。

实现流程:

设置审核者的用户 ID。

调用 friendApplicationService.auditFriendApplication()方法审核申请。

返回成功或失败的响应。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.12. getFriendApplication

功能:

获取当前用户的所有好友申请。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

输出项:

返回当前用户的好友申请列表。

失败时返回错误信息。

实现流程:

调用 friendApplicationService.getAllFriendApplication()方法获取好友申请列表。

返回申请列表或错误信息。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.13. deleteFriend

功能:

删除好友。

输入项:

idFromToken: 当前用户的 ID（从请求的 Token 中获取）。

friendDeleteDTO: FriendDeleteDTO 对象，包含删除好友的相关信息。

输出项:

返回好友删除成功的消息。

失败时返回错误信息。

实现流程:

设置操作员的用户 ID。

调用 chatService.deleteFriend()方法删除好友。

返回成功或失败的响应。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.14. createChatId

功能:

创建聊天并返回聊天 ID。

输入项:

chat: ChatDTO 对象，包含聊天信息。

type: 聊天类型（如朋友或群聊）。

输出项:

返回创建的聊天 ID。

失败时返回-1。

实现流程:

调用 chatService.createChat()方法创建聊天。

返回创建的聊天 ID

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.15. checkFriendship

功能:

检查用户是否为好友。

输入项:

userId: 用户 ID。

friendId: 好友 ID。

输出项:

返回好友关系状态。

实现流程:

调用 chatService.checkFriendship()方法检查好友关系。

返回好友状态信息。

限制条件及出错处理:如果发生异常，返回 500 错误并传递错误信息。

4.1.1.16. quitGroupsChat

功能:

用户退出群聊。

输入项:

chatId: 群聊 ID。

输出项:

返回成功退出的消息。

失败时返回错误信息。

实现流程:

调用 chatService.quitGroupsChat()方法退出群聊。

返回退出成功或失败的响应。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.1.17. inviteIntoGroupChat

功能:

邀请用户进入群聊。

输入项:

inviteDto: 邀请 DTO，包含邀请信息。

输出项:

返回邀请成功的消息。

失败时返回错误信息。

实现流程:

调用 chatService.inviteIntoGroupChat()方法邀请用户加入群聊。

返回邀请结果或错误信息。

限制条件及出错处理:

如果发生异常，返回 500 错误并传递错误信息。

4.1.2. ChatService 类

4.1.2.1. createChat

功能:

创建新的聊天并将成员添加到聊天。

输入项:

chatDto: 包含聊天信息的 DTO 对象。

chatType: 聊天类型（如群聊或私聊）。

输出项:

返回 Chat 对象，表示成功创建的聊天。

实现流程:

创建一个新的 Chat 对象，并将 chatDto 中的属性复制到该对象。

设置聊天的类型和创建时间。

保存 Chat 对象到 chatRepository。

对成员列表进行处理，为每个成员创建 ChatMember 对象并将其保存到 chatMemberRepository。

限制条件及出错处理:

如果成员列表为空或聊天信息缺失，可能会导致错误。

4.1.2.2. getChatsByUserId

功能:

获取用户参与的所有聊天记录。

输入项:

userId: 用户 ID。

输出项:

返回包含该用户所有聊天的列表。

实现流程:

调用 chatRepository.findChatsByUserId(userId)获取所有聊天记录。

限制条件及出错处理:

若未找到聊天记录，返回空列表。

4.1.2.3. quitChat

功能:

用户退出某个聊天。

输入项:

chatId: 聊天 ID。

userId: 用户 ID。

输出项:

无输出项，若成功执行返回 void。

实现流程:

删除用户与聊天的关联。

如果该聊天没有其他成员，删除该聊天及相关消息。

限制条件及出错处理:

如果用户没有加入该群聊，抛出 RuntimeException。

如果没有成员，删除聊天和消息。

4.1.2.4. inviteToChat

功能:

邀请好友加入聊天。

输入项:

inviteDto: 包含邀请人、被邀请人、聊天 ID 等信息的 DTO。

输出项:

无输出项，若成功执行返回 void。

实现流程:

检查邀请人和被邀请人是否为好友。

检查群聊类型是否允许邀请好友。

检查邀请人是否为该群聊的成员。

如果被邀请人不是群聊成员，则将其添加到群聊。

限制条件及出错处理:

如果不是好友关系，抛出 IllegalArgumentException。

如果群聊类型不支持邀请好友，抛出 IllegalArgumentException。

如果邀请人不是群聊成员，抛出 IllegalArgumentException。

如果被邀请人已是群聊成员，抛出 RuntimeException。

4.1.2.5. getChatDetails

功能:

获取群聊的详细信息，包括成员信息。

输入项:

chatId: 聊天 ID。

输出项:

返回 ChatDetailDTO 对象，包含聊天的详细信息和成员列表。

实现流程:

根据 chatId 查找聊天。

获取该聊天的所有成员，并查询每个成员的基本信息。

将所有成员信息添加到 ChatDetailDTO 中。

限制条件及出错处理:

如果未找到聊天，抛出 RuntimeException。

4.1.2.6. getFriendsBy

功能:

获取用户的好友列表。

输入项:

userId: 用户 ID。

输出项:

返回 FriendDTO 对象的列表，表示用户的好友。

实现流程:

从 chatRepository 获取该用户所有的聊天记录。

将聊天记录映射为 FriendDTO。

对每个好友获取并复制其个人信息。

限制条件及出错处理:

若没有找到好友，返回空列表。

4.1.2.7. deleteFriend

功能:

删除好友关系。

输入项:

ff: FriendDeleteDTO，包含操作人、目标人的 ID 和聊天 ID。

输出项:

无输出项，若成功执行返回 void。

实现流程:

检查操作人与目标人是否是好友关系。

删除好友关系、聊天成员及相关消息。

删除好友申请记录。

限制条件及出错处理:

如果没有该好友关系，抛出 IllegalArgumentException。

4.1.2.8. checkFriendship

功能:

检查两人是否是好友。

输入项:

user1: 第一个用户 ID。

user2: 第二个用户 ID。

输出项:

返回 boolean，表示是否为好友。

实现流程:

调用 chatRepository.getFriendship()检查是否为好友。

限制条件及出错处理:

如果没有找到好友关系，返回 false。

4.1.2.9. quitGroupChat

功能:

用户退出群聊。

输入项:

chatId: 群聊 ID。

userId: 用户 ID。

输出项:

无输出项，若成功执行返回 void。

实现流程:

从 chatMemberRepository 中删除用户与群聊的关联。

如果群聊没有其他成员，删除该群聊和相关消息。

限制条件及出错处理:

如果群聊没有成员，删除群聊和消息。

4.1.2.10. inviteIntoGroupChat

功能:

邀请用户加入群聊。

输入项:

chatId: 群聊 ID。

userId: 被邀请用户 ID。

输出项:

无输出项，若成功执行返回 void。

实现流程:

检查群聊是否支持邀请好友。

如果用户不是群聊成员，则将其添加为成员。

限制条件及出错处理:

如果群聊不支持邀请好友，抛出 IllegalArgumentException。

如果用户已是群聊成员，抛出 RuntimeException。

4.1.3. FriendApplicationService 类

4.1.3.1. postFriendApplication

功能：

发送好友申请。

输入项：

application：包含申请人和被申请人信息的对象。

输出项

无。

实现流程：

检查是否已存在相同的好友申请记录。

如果不存在，创建新的好友申请，并设置状态为“等待”，操作时间为当前时间，过期时间为 3 天后。

将好友申请保存到数据库。

限制条件及出错处理：

如果已经存在相同的申请记录，抛出 IllegalArgumentException，提示“无需再发送好友申请”。

4.1.3.2. auditFriendApplication

功能：

审核好友申请。

输入项：

AuditResultDTO auditResultDTO：包含审核结果的对象，包含申请 ID、审核人 ID 及审核结果。

输出项：

无。

实现流程：

检查是否存在待处理的好友申请。

如果存在：如果审核结果为接受，将申请状态修改为“已接受”，并创建好友聊天。

如果审核结果为拒绝，将申请状态修改为“已拒绝”。

如果申请不存在或不处于等待状态，抛出 IllegalArgumentException，提示“找

不到该好友申请”。

限制条件及出错处理：

如果好友申请不存在或状态不是“等待”，抛出 IllegalArgumentException，提示“找不到该好友申请”。

4.1.3.3. getAllFriendApplication

功能：

获取用户的所有好友申请。

输入项：

Integer userId：用户 ID。

输出项：

List<ApplicationResponseDTO>：包含所有好友申请信息的 DTO 列表。

实现流程：

从数据库中查找所有针对指定用户的好友申请。

填充每个申请的申请人和审核人信息。

返回包含所有申请的 DTO 列表。

限制条件及出错处理：

无异常抛出。

4.1.4. MessageService 类

4.1.4.1. sendMessage

功能：

发送消息到指定的群聊中。

输入项：

MessageDTO messageDto: 包含要发送的消息内容、发送者用户 ID、群聊 ID 等信息。

输出项：

Message: 保存的消息实体对象。

实现流程：

检查发送者是否是该群聊的成员

（通过 chatMemberRepository.existsChatMemberByChatIdAndUserId）。

如果是成员，创建 Message 对象，并将 messageDto 的属性拷贝到 Message 对象中。

设置消息发送时间为当前时间。

保存消息到数据库，并返回消息对象。

限制条件及出错处理：

如果发送者不是群聊成员，抛出 RuntimeException，提示“该用户并非该群聊的成员”。

4.1.4.2. getChatHistory

功能：

获取指定群聊的历史消息。

输入项：

Integer chatId: 群聊 ID。

Integer userId: 请求历史消息的用户 ID。

输出项：

List<MessageUserDTO>: 包含历史消息的 DTO 列表，每个消息都包含发送者的信息。

实现流程：

检查用户是否是该群聊的成员

（通过 chatMemberRepository.existsChatMemberByChatIdAndUserId）。

如果是成员，获取群聊的历史消息

（通过 messageRepository.getHistoryByChatId）。

将每条消息转换为 MessageUserDTO，并拷贝消息和用户的属性。

返回 MessageUserDTO 对象的列表。

限制条件及出错处理：

如果用户不是群聊成员，抛出 RuntimeException，提示“该用户并非该群聊的成员”。

4.1.4.3. deleteMsg

功能：

撤回指定消息。

输入项：

Integer userId: 请求撤回消息的用户 ID。

Integer messageId: 需要撤回的消息 ID。

输出项：

无返回值。

实现流程：

调用 messageRepository.deleteByMessageIdAndUserIdAndTime 删除消息。

删除条件包括消息 ID、用户 ID 以及消息发送时间不超过 5 分钟。

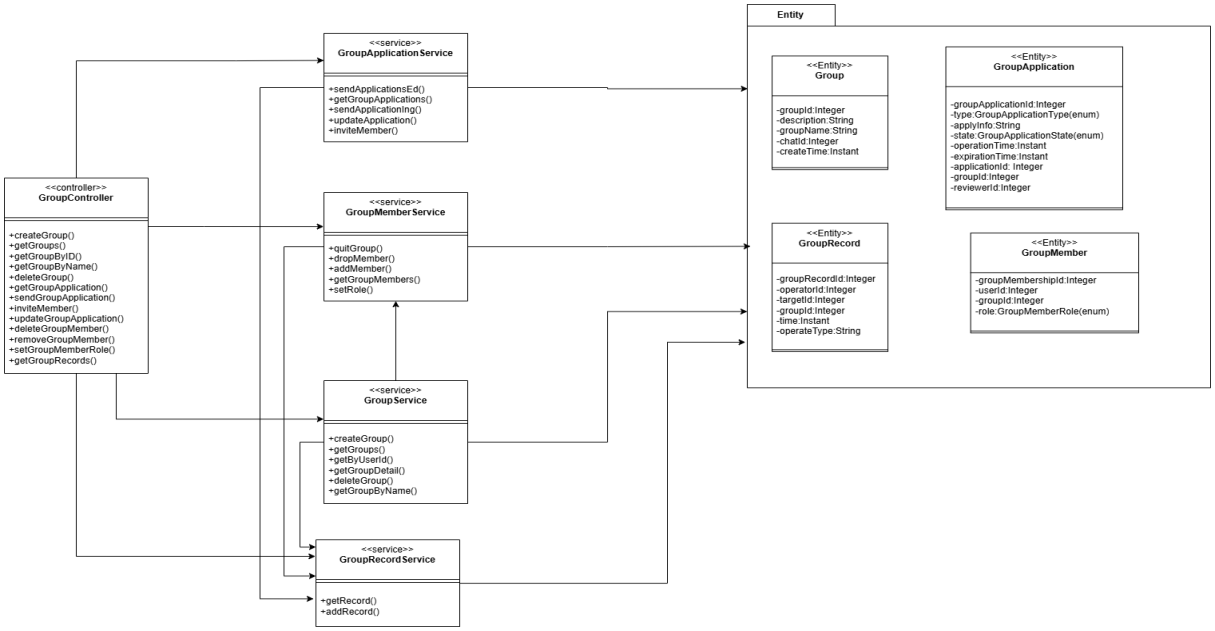
如果删除失败，抛出 RuntimeException，提示“撤回该信息失败”。

限制条件及出错处理：

如果消息无法删除（删除操作返回 0），抛出 RuntimeException，提示“撤回该信息失败”。

4.2. 团体管理子系统

负责团体的创建和管理，用户加入团体和团体成员的管理。



4.2.1. GroupController

4.2.1.1. createGroup

功能：

创建一个新的团体。

输入项：

CompleteGroupDTO completeGroup: 包含创建团体所需的详细信息，如团体名称、描述等。

输出项：

ResponseEntity<Object>: 成功时返回团体创建成功的信息，失败时返回错误信息。

实现流程：

从请求中获取当前用户的 ID（通过 idFromToken）。

设置团体的创建者为当前用户。

调用 groupService.createGroup 方法创建团体。

返回操作结果。

限制条件及出错处理：

若创建失败，抛出异常并返回错误信息。

4.2.1.2. getGroups

功能：

获取所有团体的列表。

输入项：

无。

输出项：

ResponseEntity<Object>: 返回团体列表，若失败返回错误信息。

实现流程：

调用 groupService.getGroups 方法获取所有团体信息。

返回团体列表或错误信息。

限制条件及出错处理：

若获取团体列表失败，抛出异常并返回错误信息。

4.2.1.3. getGroupByID

功能：

根据团体 ID 获取团体详细信息。

输入项：

Integer groupId: 要查询的团体 ID。

输出项：

ResponseEntity<Object>: 返回团体详细信息，若失败返回错误信息。

实现流程：

根据团体 ID 查询团体详情，调用 groupService.getGroupDetail 方法。

返回团体详情或错误信息。

限制条件及出错处理：

若团体不存在或查询失败，抛出异常并返回错误信息。

4.2.1.4. getGroupByName

功能：

根据团体名称获取团体详细信息。

输入项：

String groupName: 要查询的团体名称。

输出项：

ResponseEntity<Object>: 返回团体详细信息，若失败返回错误信息。

实现流程：

根据团体名称查询团体信息，调用 groupService.getGroupByName 方法。

返回团体详情或错误信息。

限制条件及出错处理：

若团体不存在或查询失败，抛出异常并返回错误信息。

4.2.1.5. deleteGroup

功能：

解散指定团体。

输入项：

Integer groupId: 要解散的团体 ID。

int idFromToken: 当前用户的 ID。

输出项：

ResponseEntity<Object>: 成功时返回解散团体成功的信息，失败时返回错误信息。

实现流程：

从请求中获取当前用户 ID（通过 idFromToken）。

调用 groupService.deleteGroup 方法删除指定团体。

返回操作结果。

限制条件及出错处理：

若解散团体失败，抛出异常并返回错误信息。

4.2.1.6. getGroupApplication

功能：

获取当前用户的团体申请。

输入项：

int idFromToken: 当前用户的 ID。

输出项：

ResponseEntity<Object>: 返回当前用户的团体申请列表，若失败返回错误信息。

实现流程：

调用 groupApplicationService.getGroupApplications 获取当前用户的团体申请列表。

返回团体申请列表或错误信息。

限制条件及出错处理：

若获取申请列表失败，抛出异常并返回错误信息。

4.2.1.7. sendGroupApplication

功能：

发送团体加入申请。

输入项：

GroupApplicationDTO groupApplicationDTO: 包含申请团体的详细信息，如团体 ID 和申请者 ID。

输出项：

ResponseEntity<Object>: 成功时返回发送申请成功的信息，失败时返回错误

信息。

实现流程：

设置申请者的 ID 为当前用户的 ID。

调用 groupApplicationService.sendApplicationIng 发送申请。

返回操作结果。

限制条件及出错处理：

若发送失败，抛出异常并返回错误信息。

4.2.1.8. inviteMember

功能：

邀请用户加入团体。

输入项：

InviteGroupDTO inviteDTO: 包含邀请信息，如团体 ID 和被邀请用户的 ID。

输出项：

ResponseEntity<Object>: 成功时返回邀请成功的信息，失败时返回错误信息。

实现流程：

调用 groupApplicationService.inviteMember 发送邀请。

返回操作结果。

限制条件及出错处理：

若邀请失败，抛出异常并返回错误信息。

4.2.1.9. updateGroupApplication

功能：

处理团体加入申请。

输入项：

AuditResultDTO auditResultDTO: 包含审核结果信息，如审核者 ID 和申请 ID。

输出项：

ResponseEntity<Object>: 成功时返回处理成功的信息，失败时返回错误信息。

实现流程：

设置审核者的 ID 为当前用户的 ID。

调用 groupApplicationService.updateApplication 处理团体加入申请。

返回操作结果。

限制条件及出错处理：

若处理失败，抛出异常并返回错误信息。

4.2.1.10. deleteGroupMember

功能：

退出团体。

输入项：

Integer groupId: 要退出的团体 ID。

int idFromToken: 当前用户的 ID。

输出项：

ResponseEntity<Object>: 成功时返回退出团体成功的信息，失败时返回错误信息。

实现流程：

调用 groupMemberService.quitGroup 退出团体。

返回操作结果。

限制条件及出错处理：

若退出失败，抛出异常并返回错误信息。

4.2.1.11. removeGroupMember

功能：

将用户移出团体。

输入项：

MemberDropDTO dropDTO: 包含移除成员的信息，如移除者 ID 和被移除者 ID。

输出项：

ResponseEntity<Object>: 成功时返回移除成员成功的信息，失败时返回错误信息。

实现流程：

设置操作用户的 ID 为当前用户的 ID。

调用 groupMemberService.dropMember 移除指定成员。

返回操作结果。

限制条件及出错处理：

若移除失败，抛出异常并返回错误信息。

4.2.1.12. setGroupMemberRole

功能：

设置团体成员的角色权限。

输入项：

RoleDTO roleDTO: 包含设置角色权限的信息，如成员 ID、角色等。

输出项：

ResponseEntity<Object>: 成功时返回角色设置成功的信息，失败时返回错误信息。

实现流程：

设置操作用户的 ID 为当前用户的 ID。

调用 groupMemberService.setRole 方法设置角色权限。

返回操作结果。

限制条件及出错处理：

若设置失败，抛出异常并返回错误信息。

4.2.1.13. getGroupRecords

功能：

获取团体的记录。

输入项：

Integer groupId: 团体 ID。

Integer targetId: 目标 ID（例如：特定用户 ID）。

int idFromToken: 当前用户的 ID。

输出项：

ResponseEntity<Object>: 返回团体记录，若失败返回错误信息。

实现流程：

调用 groupRecordService.getRecord 获取团体的记录。

返回团体记录或错误信息。

限制条件及出错处理：

若获取记录失败，抛出异常并返回错误信息。

4.2.2. GroupApplicationService 类

4.2.2.1. sendApplicationsEd

功能:

向指定目标用户发送团体邀请申请。

输入项:

List<Integer> targets: 被邀请用户的 ID 列表

Integer userId: 邀请者 ID

Group group: 目标团体

输出项:

无返回值

实现流程:

遍历目标用户 ID 列表，为每个目标创建一个 GroupApplication 对象。

设置申请类型为"invited"，并设置申请的状态为"waiting"。

生成并保存所有邀请申请。

创建相关的团体记录。

限制条件及出错处理:

无

4.2.2.2. getGroupApplications

功能:

获取指定用户的所有团体申请（包括申请和被邀请）。

输入项:

Integer userId: 用户 ID

输出项:

List<GroupApplication>: 用户的所有申请

实现流程:

查询用户作为审核者的所有申请记录。

查询用户作为团体成员的所有申请记录。

合并两个列表并返回。

限制条件及出错处理:

无

4.2.2.3. sendApplicationIng

功能:

创建并发送加入团体的申请。

输入项:

GroupApplicationDTO groupApplicationDTO: 团体申请数据传输对象

输出项:

无

实现流程:

将 groupApplicationDTO 的属性复制到 GroupApplication 对象中。

设置申请类型为"apply"并设置状态为"waiting"。

保存申请记录。

限制条件及出错处理:

无

4.2.2.4. updateApplication

功能:

审核并更新团体申请状态。

输入项:

AuditResultDTO auditResultDTO: 审核结果数据传输对象

输出项:

无

实现流程:

查找指定 ID 的申请记录。

检查申请是否过期、是否已被处理、审核权限等条件。

根据审核结果更新申请状态：若接受申请，则将申请者添加到团体并邀请加入团体聊天。

若拒绝申请，则记录拒绝行为。

保存更新后的申请记录和团体记录。

限制条件及出错处理:

如果找不到申请记录，抛出 IllegalArgumentException。

如果申请已过期或已处理过，抛出 RuntimeException。

如果没有权限审核该申请，抛出 IllegalArgumentException。

如果要加入的团体不存在，抛出 IllegalArgumentException。

4.2.2.5. inviteMember

功能:

邀请用户加入团体。

输入项:

InviteGroupDTO inviteDTO: 邀请数据传输对象

Integer invitor: 邀请者 ID

输出项:

无

实现流程:

验证邀请者是否有权限邀请该用户加入团体，并且双方是否为好友。

创建并保存团体邀请申请。

创建并保存团体邀请记录。

限制条件及出错处理:

如果邀请者没有权限，或不是好友，抛出 IllegalArgumentException。

4.2.3. GroupMemberService 类

4.2.3.1. quitGroup

功能:

使指定的成员退出团体并清理相关记录。

输入项:

Integer groupId: 团体 ID

Integer memberId: 成员 ID

输出项:

无返回值

实现流程:

检查成员是否已加入团体。

删除成员相关的团体申请记录。

从团体成员列表中删除该成员。

删除成员在团体中的相关记录。

使成员退出团体的聊天。

如果该团体没有成员，则删除团体。

限制条件及出错处理:

如果用户未加入团体，抛出 IllegalArgumentException。

4.2.3.2. dropMember

功能:

删除指定成员的团体成员身份。

输入项:

MemberDropDTO memberDropDTO: 包含团体 ID、操作员 ID 和成员 ID 的 DTO 对象

输出项:

无返回值

实现流程:

检查成员是否已加入团体。

检查操作员是否有权限删除该成员。

删除成员的团体申请记录、成员记录及相关团体记录。

使成员退出团体的聊天。

记录删除操作。

限制条件及出错处理:

如果成员未加入团体，抛出 IllegalArgumentException。

如果操作员没有权限删除该成员，抛出 IllegalArgumentException。

4.2.3.3. addMember

功能:

将指定成员添加到团体。

输入项:

Integer groupId: 团体 ID

Integer memberId: 成员 ID

输出项:

无返回值

实现流程:

将成员添加到团体成员表中，设置默认角色为 member。

限制条件及出错处理:

无

4.2.3.4. getGroupMembers

功能:

获取指定团体的所有成员信息。

输入项:

Integer groupId: 团体 ID

输出项:

List<GroupMemberDetailDTO>: 团体成员的详细信息列表

实现流程:

查询团体成员信息。

将成员信息转换为 GroupMemberDetailDTO 对象，并填充成员的角色、ID、用户名和头像。

返回成员信息列表。

限制条件及出错处理:

无

4.2.3.5. setRole

功能:

设置团体成员的角色（例如，设置为管理员）。

输入项:

RoleDTO roleDTO: 包含团体 ID、操作员 ID、目标成员 ID 和角色信息的 DTO 对象

输出项:

无返回值

实现流程:

检查操作员是否有权限修改角色。

更新目标成员的角色信息。

记录操作。

限制条件及出错处理:

如果操作员没有权限进行该操作，抛出 IllegalArgumentException。

4.2.4. GroupRecordService 类

4.2.4.1. addRecord

功能:

添加团体操作记录。

输入项:

Integer operator: 操作员 ID

Integer target: 目标成员 ID

Integer groupId: 团体 ID

String type: 操作类型（例如“加入团体”、“设置管理员”等）

输出项:

无返回值

实现流程:

创建一个新的 GroupRecord 对象。

设置该记录的团体 ID、操作时间、操作类型、操作员 ID 和目标成员 ID。

将操作记录保存到 groupRecordRepository 中。

限制条件及出错处理:

无

4.2.4.2. getRecord

功能:

获取指定团体成员的操作记录。

输入项:

Integer operator: 操作员 ID

Integer targetId: 目标成员 ID

Integer groupId: 团体 ID

输出项:

List<GroupRecord>: 指定成员的操作记录列表

实现流程:

检查操作员是否有权限查看该团体的记录。

如果有权限，调用 groupRecordRepository.findRecords 方法查询并返回相关记录。

如果没有权限，抛出 IllegalArgumentException，提示“没有权限查看团员记录”。

限制条件及出错处理:

如果操作员没有权限查看记录，抛出 IllegalArgumentException。

4.2.5. GroupService 类

4.2.5.1. createGroup

功能:

创建一个新的团体并初始化其成员和记录。

输入项:

CompleteGroupDTO completeGroup: 包含团体创建者 ID、团体名称和初始成员的详细信息。

输出项:

无返回值。

实现流程:

调用 socializeController 创建群聊，获取 chatId。

将 completeGroup 的属性复制到 Group 实体，设置创建时间和 chatId。

保存团体到数据库并返回团体实例。

创建并保存创建者为管理员的团体成员信息。

调用 groupRecordService 添加“创建团体”记录。

调用 groupApplicationService 发送邀请给初始成员。

限制条件及出错处理:

无显式错误处理。假定依赖服务和数据库操作正常运行。

4.2.5.2. getGroups

功能:

获取所有团体列表。

输入项:

无。

输出项:

List<Group>: 所有团体的列表。

实现流程:

调用 groupRepository.findAll()获取所有团体。

返回结果。

限制条件及出错处理:

无显式错误处理。

4.2.5.3. getByUserId

功能:

获取某用户参与的所有团体。

输入项:

Integer userId: 用户 ID。

输出项:

List<Group>: 用户参与的团体列表。

实现流程:

调用 groupMemberRepository 获取用户参与的团体成员关系。

遍历成员关系列表，根据团体 ID 查询对应的团体信息。

如果团体信息缺失，抛出 ServiceException 提示“未找到团体”。

返回用户参与的团体列表。

限制条件及出错处理:

如果查询不到团体，抛出 ServiceException。

4.2.5.4. getGroupDetail

功能:

获取指定团体的详细信息，包括成员列表。

输入项:

Integer groupId: 团体 ID。

输出项:

GroupDetailDTO: 团体详情数据传输对象，包括团体基本信息和成员信息。

实现流程:

根据 groupId 调用 groupRepository 获取团体实体。

如果团体不存在，抛出 IllegalArgumentException 提示“找不到该团体”。

将团体实体属性复制到 GroupDetailDTO 对象。

调用 groupMemberService.getGroupMembers 获取成员信息并设置到 GroupDetailDTO 中。

返回 GroupDetailDTO 对象。

限制条件及出错处理:

如果团体不存在，抛出 IllegalArgumentException。

4.2.5.5. deleteGroup

功能:

删除指定团体。

输入项:

Integer groupId: 团体 ID。

Integer userId: 操作用户 ID。

输出项:

无返回值。

实现流程:

检查用户是否有权限删除团体（调用 groupMemberRepository.checkAuth）。

如果有权限，删除与团体相关的成员关系、记录、申请信息，并删除团体本身。

如果无权限，抛出 IllegalArgumentException 提示“该用户没有权限解散团体”。

限制条件及出错处理:

如果用户无权限，抛出 IllegalArgumentException。

4.2.5.6. getGroupByName

功能:

根据团体名称模糊查询团体列表。

输入项:

String groupName: 团体名称。

输出项:

List<Group>: 匹配名称的团体列表。

实现流程:

调用 groupRepository.findByName 根据名称查询团体。

返回查询结果。

限制条件及出错处理:

无显式错误处理。