

Fluent Chat：即时聊天应用

项目说明书

目录

1	项目说明.....	3
1.1	开发目标	3
1.2	需求分析	3
1.1.1	用户需求分析	3
1.1.2	功能需求分析	3
1.3	技术选用	3
1.1.1	服务端.....	3
1.1.2	客户端.....	4
1.1.3	网页端.....	4
1.4	开发环境	4
2	用例说明.....	4
2.1	注册账号	4
2.2	登录账号	4
2.3	添加好友	5
2.4	删除好友	5
2.5	创建群聊	5
2.6	加入群聊	6
2.7	退出群聊	6
2.8	删除用户	6
2.9	修改用户权限	7
3	架构设计.....	7
1.1	数据库设计.....	7
1.2	服务端设计.....	8
1.3	客户端设计.....	12
1.4	网页端设计.....	16
4	功能测试.....	17

4.1	注册账号	17
4.2	登录账号	18
4.3	添加好友	18
4.4	删除好友	19
4.5	创建群聊	20
4.6	加入群聊	21
4.7	退出群聊	21
4.8	用户管理	22

1 项目说明

1.1 开发目标

随着计算机科学技术的发展，网络越来越深刻地改变着人们的生活。各种基于网络的应用技术在人们的政治、经济、生活等各个方面都发挥着重要的作用。基于英特网的即时聊天工具，比传统媒介具有数据量大、实时性强、操作简单成本低廉等优点，所以在现实生活中受到了广泛的欢迎。

本项目的目标是开发一个即时通讯系统，同时使用 CS/BS 两种架构。能够实现用户之间高效的交流通信，而且支持群聊功能，能够支持多个用户在一个群中发言，软件简洁美观，操作人性化。

1.2 需求分析

1.1.1 用户需求分析

- 用户第一次打开软件时可以方便快速地注册账号
- 用户可以发送添加好友请求，对方同意后可以成为好友，好友之间可以相互发送信息。
- 用户可以创建群聊，并且为群聊设置加入密码，其他用户可以通过密码加入该群聊后进行发言，群中用户的发言可以被其他群成员看到。
- 管理员可以在后台管理用户，设置用户权限，删除用户。

1.1.2 功能需求分析

- 通信模块能够实现客户端与服务端之间的通信。
- 用户模块能够支持用户的注册、登录，并且管理用户状态，用户权限。
- 消息模块能够支持用户之间的消息传递，服务端对客户端的消息推送。
- 各种界面组件能够正常显示各种信息，能够稳定运行，并且对于操作、错误有着明显的提示。

1.3 技术选用

1.1.1 服务端

服务端采用 Java 编写，并使用 SpringBoot 框架。
数据库部分使用 Jpa 框架，能够同时兼容 Oracle 和 Mysql

使用 HTTP 协议与前端进行通讯，使用 TCP 协议与客户端进行通讯。

1.1.2 客户端

客户端采用 C++ 编写，使用 WinUI3 框架。

1.1.3 网页端

网页端采用 Typescript 编写，使用 React 框架。

UI 部分选用 MaterialUI 框架，前后端通信使用 Axios。

开发过程中使用 Eslint 对代码进行检查。

1.4 开发环境

服务端：IntelliJ IDEA 2022.1.3

客户端：Visual Studio 2022

网页端：Visual Studio Code

操作系统：Windows 11

2 用例说明

2.1 注册账号

用例描述	注册账号
参与者	用户
前置条件	无
后置条件	用户可以登录
基本事件流	1、用户打开浏览器注册页面或打开应用进入注册页面
异常事件流	1、昵称、邮箱、密码其中之一没有输入，需要输入。 2、邮箱已存在，更换邮箱或取消注册直接登录。

2.2 登录账号

用例描述	登录账号
------	------

参与者	用户
前置条件	已注册账号
后置条件	登录成功
基本事件流	1、用户打开浏览器登录页面或打开应用登录页面
异常事件流	1、用户名输入错误，重新输入或取消登录

2.3 添加好友

用例描述	添加好友到好友列表
参与者	用户
前置条件	登录成功
后置条件	发送好友消息
基本事件流	1、输入好友昵称或邮箱进行查询 2、发送好友请求 3、对方同意好友请求
异常事件流	1、无法搜索到好友，取消添加好友 2、对方拒绝添加好友，取消添加好友

2.4 删除好友

用例描述	将好友从好友列表中删除
参与者	用户
前置条件	登录成功且已添加好友
后置条件	好友删除成功
基本事件流	1、切换到好友列表页面，右键好友名称，点击删除好友。
异常事件流	无

2.5 创建群聊

用例描述	创建群聊
参与者	用户
前置条件	登录成功
后置条件	加入群聊
基本事件流	1、转到创建群聊页面，输入群名称以及加入密码创建群聊。

异常事件流	1、群名称为空，输入群名称或取消创建 2、加入密码为空，输入加入密码或取消创建。
-------	---

2.6 加入群聊

用例描述	加入到已创建的群聊
参与者	用户
前置条件	登录成功
后置条件	发送群消息
基本事件流	1、输入群名称或群号进行查询。 2、输入加群密码加群。
异常事件流	1、加群密码错误，重新输入或取消加群。

2.7 退出群聊

用例描述	从已加入的群聊中退出
参与者	用户
前置条件	登录并且已加入到群聊中
后置条件	退出群聊成功
基本事件流	切换到好友列表页面，右键好友名称，点击退出群聊。
异常事件流	无

2.8 删除用户

用例描述	删除用户
参与者	管理员
前置条件	登录到后台且为管理员账号
后置条件	删除用户成功
基本事件流	1、在搜索框中输入用户昵称或邮箱进行查询。 2、点击删除用户。
异常事件流	无

2.9 修改用户权限

用例描述	修改用户权限
参与者	管理员
前置条件	登录到后台且为管理员账号
后置条件	修改用户权限成功
基本事件流	3、在搜索框中输入用户昵称或邮箱 进行查询。 4、设置用户权限。
异常事件流	无

3 架构设计

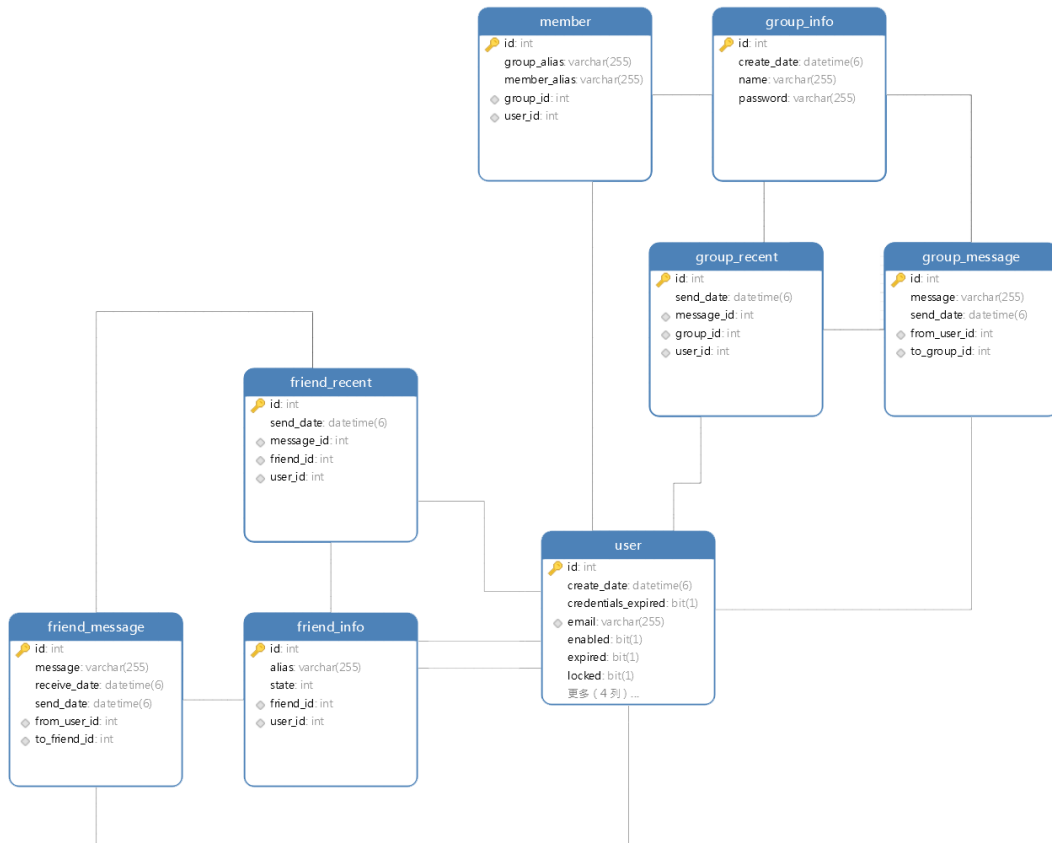
1.1 数据库设计

数据库使用 Jpa，在 Oracle 以及 Mysql 数据库上经过测试，均能正常使用。数据库一共有 8 个表，它们之间相互关联。

group_info 表储存群的信息，每个群的群成员储存在 member 表里，群的信息储存在 group_message 表里，群的最近动态储存在 group_recent 里。

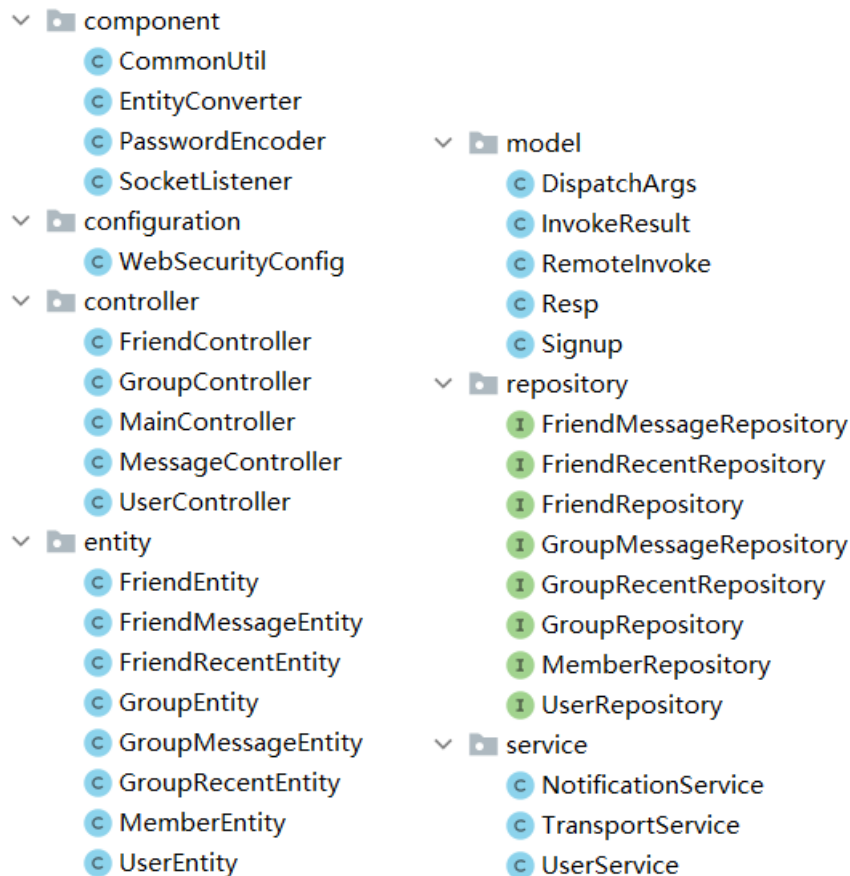
friend_info 表储存好友的信息，与 user 多对一关联，一个 user 可以有多个好友，friend_message 储存好友之间的聊天信息，friend_recent 储存好友的最近动态。

user 表则是用户表，储存着用户的昵称、邮箱、密码、权限等信息，提供用户登录时以及验证权限时使用。



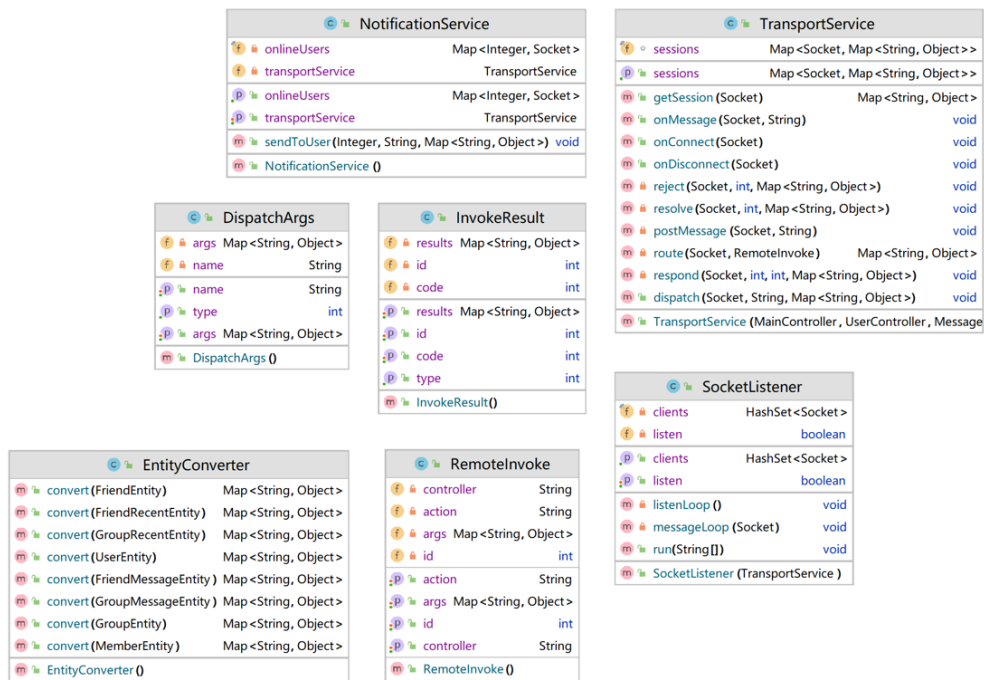
1.2 服务端设计

服务端总体使用 MVC 的架构设计

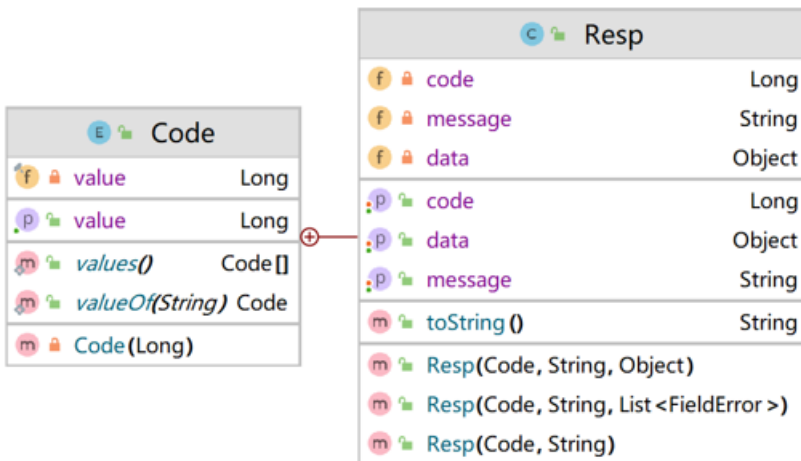


数据传输,网页端与后端通过 HTTP 进行通信,使用 Springboot 内置的 Servlet,客户端与后端通过 TCP 进行通信,首先通过 SocketListener 类监听端口,并且把数据流转为数据报文的格式,投递给 TransportService,完成第一层封装。TransportService 解析报文内容,报文内容为 JSON 格式,客户端与服务器使用类似 JSONRPC 的方式进行通讯,格式为报文内容中含有 Controller、Action、Args、ID 字段,通过这些字段,将 Args 投递到对应控制器的对应方法上,当方法执行完毕后,返回数据给客户端,并且带有客户端发来的 ID,客户端通过 ID 识别到是哪次远程调用后,就 Resume 到对应的异步方法,这样完成两层封装后,客户端可以方便地调用服务端的方法。

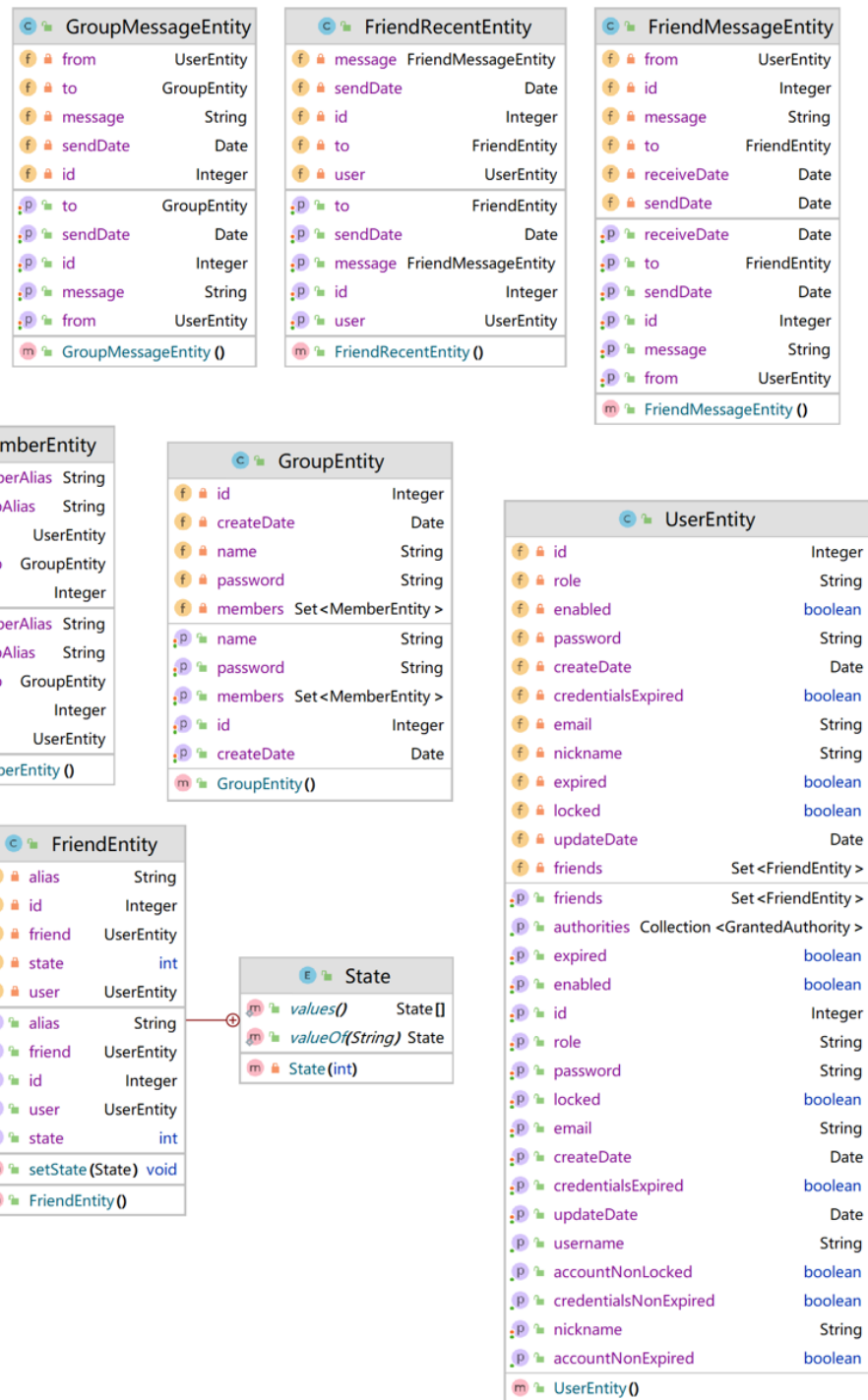
除了客户端主动调用外,服务端还可以通过 NotificationService 服务,将消息主动投递到客户端。



服务端与前端的通讯格式由一个简单的 Resp 类定义，里面有 Code，表示是否成功执行，有 data，表示执行结果，message，表示消息（如失败原因）



Jpa 实体类如下图，结构和数据库表结构一致，由 Jpa 实现数据库的表的结构同步。



除此之外，还定义了一系列查询数据库的方法。

FriendRepository		
findPending	(UserEntity, UserEntity)	FriendEntity
findNormal	(UserEntity, UserEntity)	FriendEntity
deleteFriend	(UserEntity, UserEntity)	void
findPending	(UserEntity, Pageable)	Page<FriendEntity>
deleteAllFriend	(UserEntity)	void
find	(UserEntity, UserEntity)	FriendEntity
findNormal	(UserEntity, Pageable)	Page<FriendEntity>

UserRepository		
searchByEmailOrNickname	(String, Pageable)	Page<UserEntity>
findByEmail	(String)	UserEntity
selectAll	(Pageable)	List<UserEntity>
selectAll	(Pageable, String)	List<UserEntity>

GroupRepository		
searchByNameOrId	(String, int, Pageable)	Page<GroupEntity>

GroupRecentRepository		
findByUser	(UserEntity, Pageable)	Page<GroupRecentEntity>

FriendRecentRepository		
deleteByFriend	(UserEntity, UserEntity)	void
findByUser	(UserEntity, Pageable)	Page<FriendRecentEntity>

GroupMessageRepository		
findByGroup	(GroupEntity, Pageable)	Page<GroupMessageEntity>

而具体的操作逻辑，则定义在 Controller 里，GroupController 实现群的管理、UserController 实现用户的管理、MessageController 实现消息的管理、FriendController 实现好友的管理。其中UserController 底下还有UserService 一层。

MainController		
route	(String, Map<String, Object>, Map<String, Object>)	Map<String, Object>
MainController	()	

GroupController		
quit	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
list	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
create	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
route	(String, Map<String, Object>, Map<String, Object>)	Map<String, Object>
search	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
join	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
GroupController	(GroupRepository, MemberRepository, EntityConverter)	

UserController		
signup	(Map<String, Object>)	Map<String, Object>
isInvalidEmailAddress	(String)	boolean
signup	(Signup, BindingResult)	Resp
editRole	(int, String, String)	Resp
route	(String, Map<String, Object>, Map<String, Object>)	Map<String, Object>
login	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
list	(int, int, String)	Resp
info	(Principal)	Resp
editNickname	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
delete	(int)	Resp
UserController	(UserRepository, PasswordEncoder, UserService, EntityConverter, FriendRepository)	

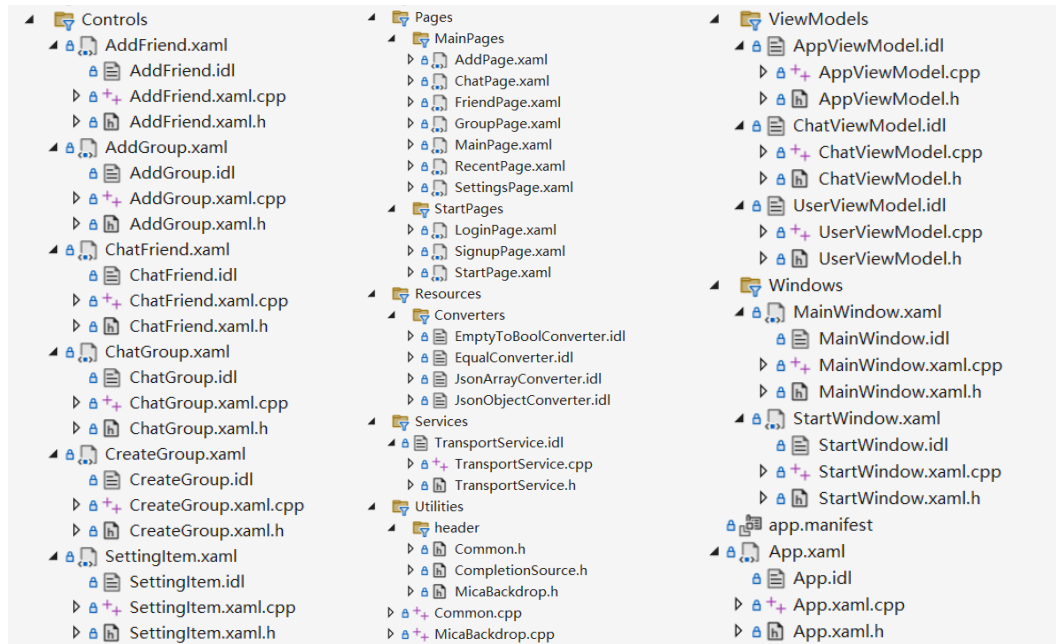
MessageController		
getGroupMessages	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
sendGroupMessage	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
route	(String, Map<String, Object>, Map<String, Object>)	Map<String, Object>
sendFriendMessage	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
getRecent	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
getFriendMessages	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
MessageController	(FriendMessageRepository, UserRepository, FriendRepository, FriendRecentRepository)	

UserService		
loadUserByUsername	(String)	UserDetails
loadUserByEmail	(String)	UserEntity
saveUser	(UserEntity)	UserEntity
loadUserById	(Integer)	UserEntity
UserService	(UserRepository, PasswordEncoder, CommonUtil)	

FriendController		
addAccept	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
listNormal	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
addRequest	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
addRefuse	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
listRequest	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
search	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
delete	(Map<String, Object>, Map<String, Object>)	Map<String, Object>
route	(String, Map<String, Object>, Map<String, Object>)	Map<String, Object>
FriendController	(FriendRepository, UserRepository, EntityConverter, NotificationService)	

1.3 客户端设计

客户端总体使用 MVVM 架构设计。

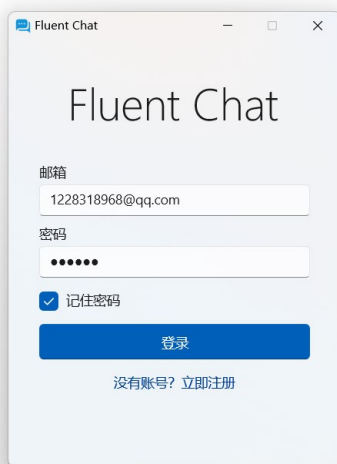


TransportService 类里，首先将消息封装为 JSON 格式，然后封装为 TCP 报文格式，最后发送给服务端，发送时生成一个会话 ID，并记录下来，然后等到服务器回复时，通过 ID 找到对应的 CompletionSource，并设置一个结果后，自动 Resume 到对应的异步方法继续执行。

```
Windows::Foundation::IAsyncOperation<JsonObject> TransportService::InvokeAsync(hstring controller, hstring action, JsonObject args)
{
    JsonObject jsonMsg;
    auto id = m_autoIncrementId++;
    jsonMsg.Insert(L"controller", JsonValue::CreateStringValue(controller));
    jsonMsg.Insert(L"action", JsonValue::CreateStringValue(action));
    jsonMsg.Insert(L"id", JsonValue::CreateNumberValue(id));
    jsonMsg.Insert(L"args", args);
    auto completionSource = std::make_shared<Utilities::CompletionSource<JsonObject>>();
    m_sessionMap.insert(std::pair<uint32_t, std::shared_ptr<Utilities::CompletionSource<JsonObject>>>(id, completionSource));
    co_await PostMessage(jsonMsg.ToString());
    auto& resp = co_await *completionSource;
    if (resp.GetNamedNumber(L"code") != 0)
        throw_hresult(hresult(COR_E_EXCEPTION));
    co_return resp.GetNamedObject(L"results");
}
```

封装完成之后，前后端的通讯就如同方法调用一样简单。

程序有两个窗口，分别是初始窗口和主窗口



Fluent Chat

邮箱
1228318968@qq.com

密码
•••••

☒ 记住密码

登录

没有账号? 立即注册



Fluent Chat

注册账号

昵称

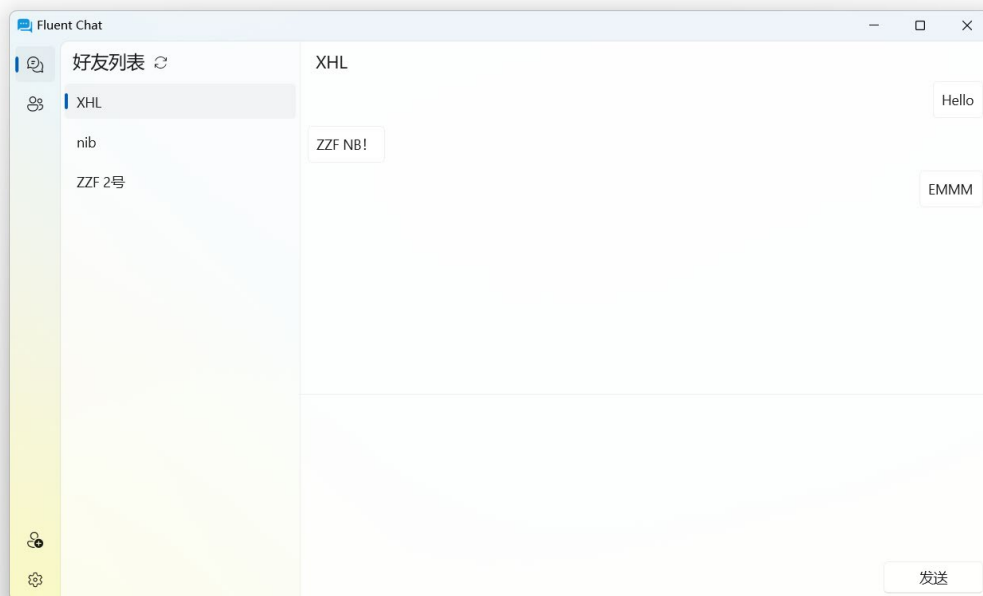
邮箱

密码

确认密码

注册

返回登录



Fluent Chat

好友列表

- XHL
- nib
- ZZF 2号

XHL

Hello

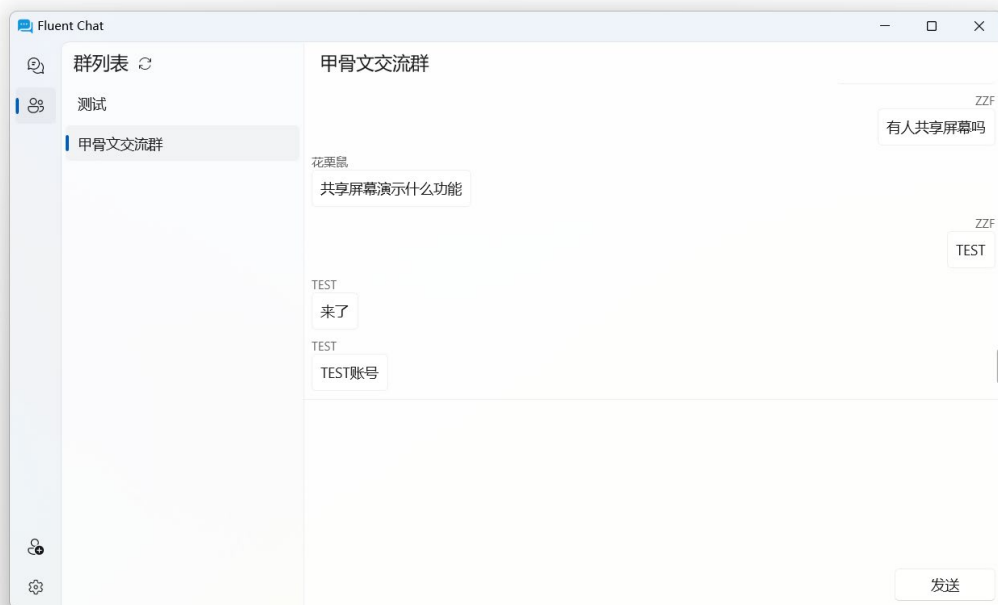
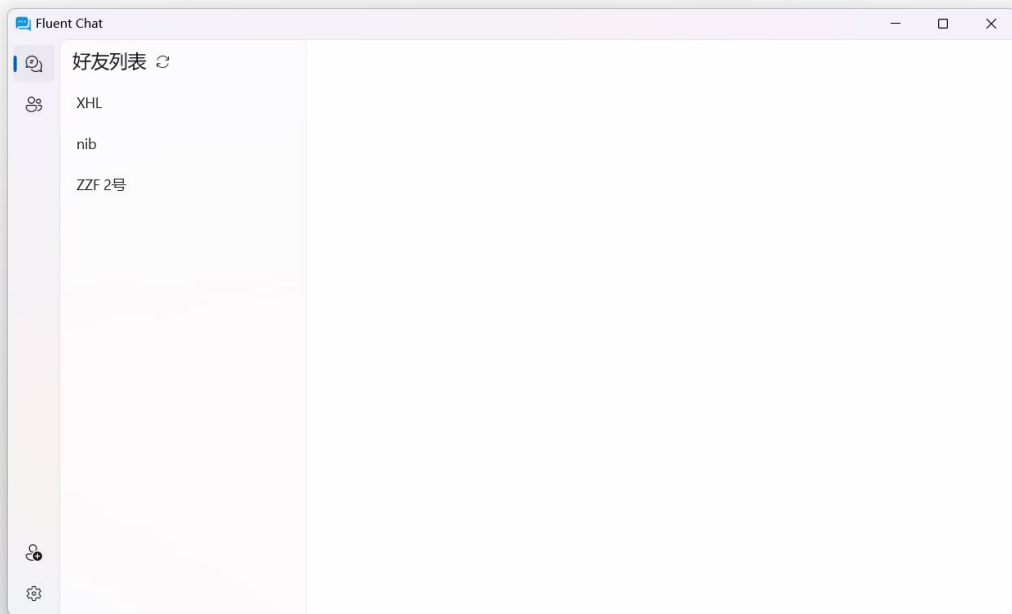
ZZF NB!

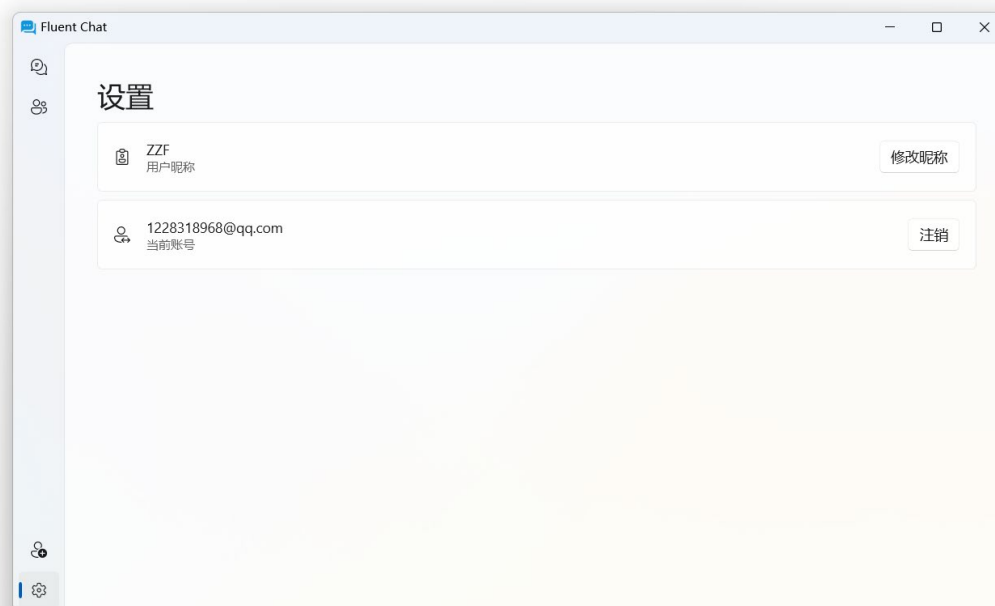
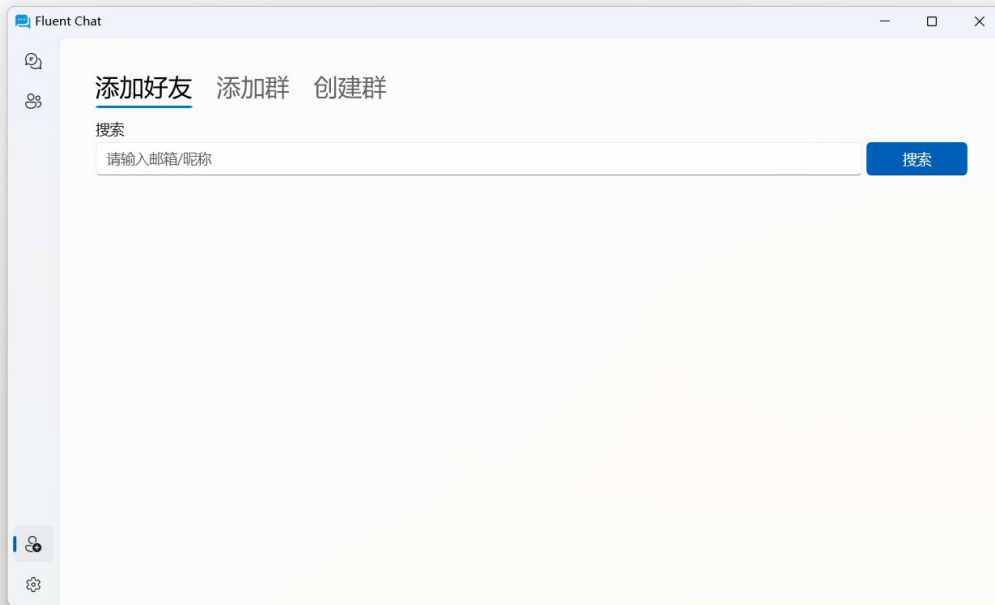
EMMM

发送

在初始窗口中又分为两个页面，分别为登录页面和注册页面。
除了这两个页面之外，其他页面都在主窗口。

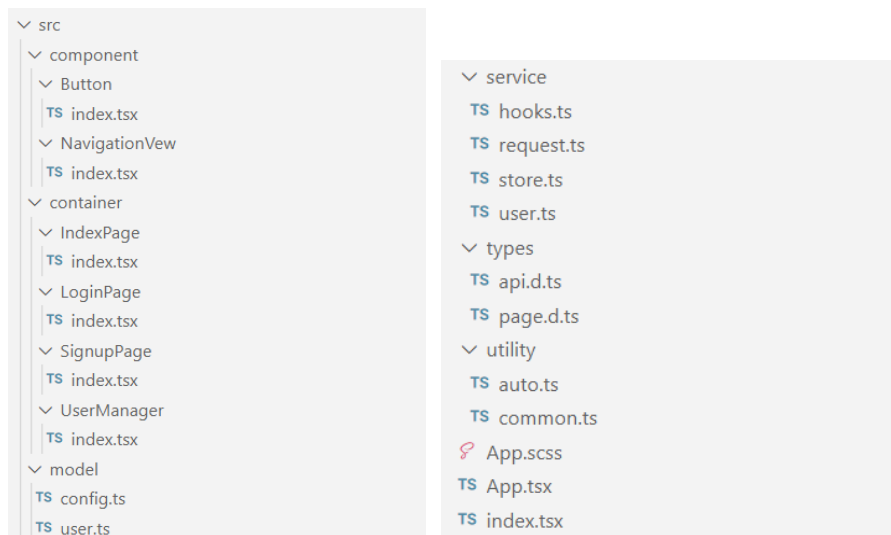
主窗口左侧为导航栏，可以选择好友列表、群列表、添加好友页面、设置页面。



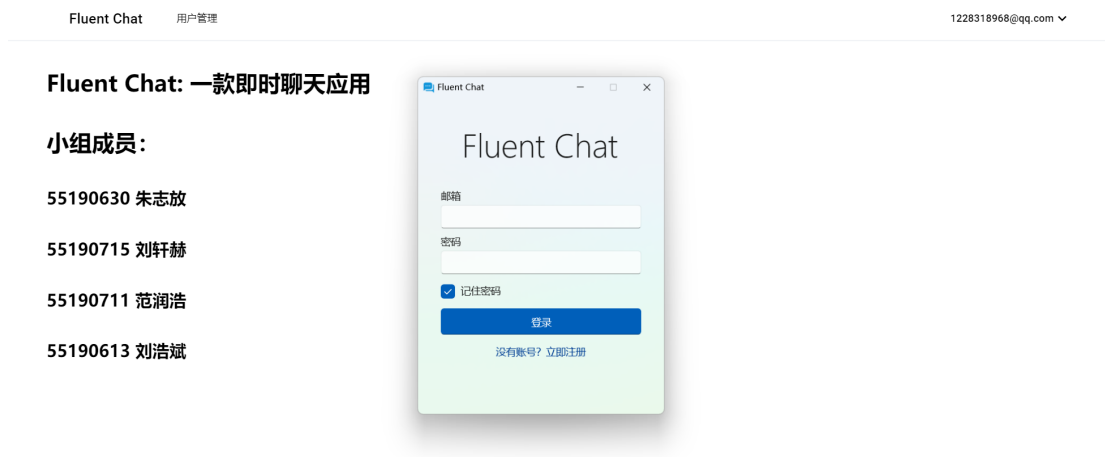


1.4 网页端设计

网页端总体使用单向数据流的架构设计，响应式编程。



实现用户的管理、注册、登录。



4 功能测试

4.1 注册账号

既可以在网页端注册，也可以在客户端注册



Fluent Chat

注册账号

昵称

邮箱

密码

确认密码

[注册](#)

[返回登录](#)



注册

昵称

电子邮箱

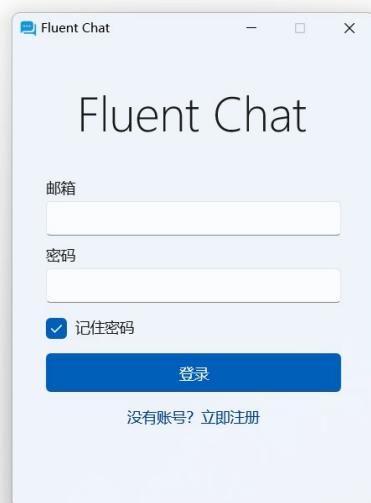
密码

[注册](#)

[返回登录](#)

4.2 登录账号

既可以在网页端登录，也可以在客户端登录



Fluent Chat

邮箱

密码

☒ 记住密码

[登录](#)

[没有账号? 立即注册](#)



登录

电子邮箱

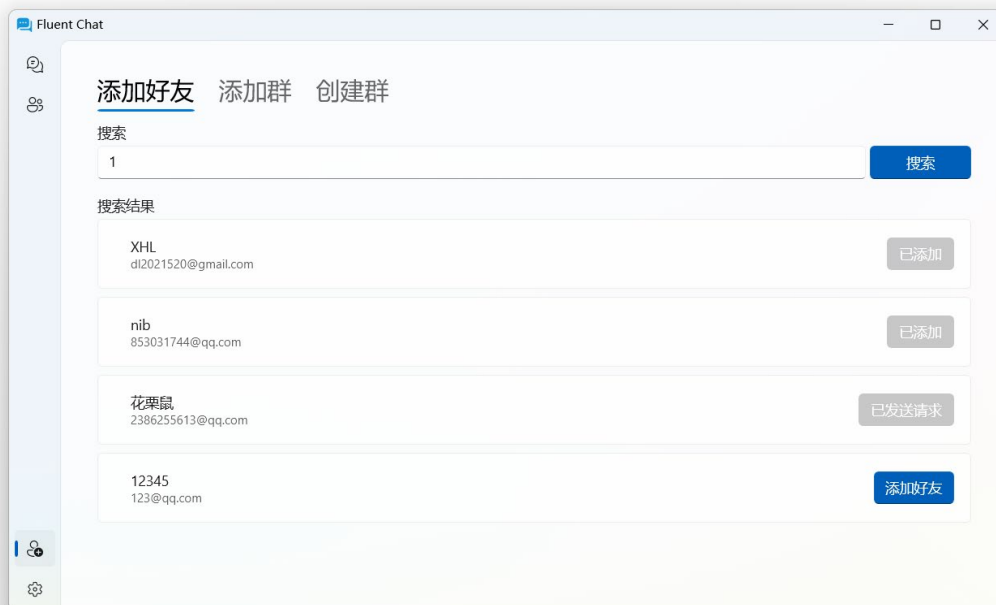
密码

[登录](#)

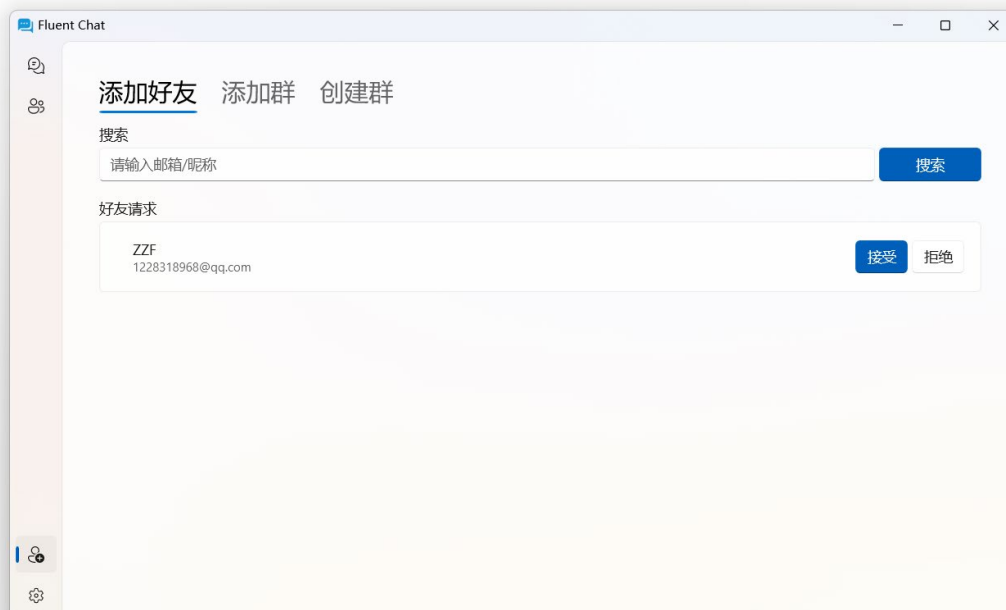
[没有账号? 立即注册](#)

4.3 添加好友

在添加好友页面可以搜索到好友，点击添加好友可以发送添加好友请求，

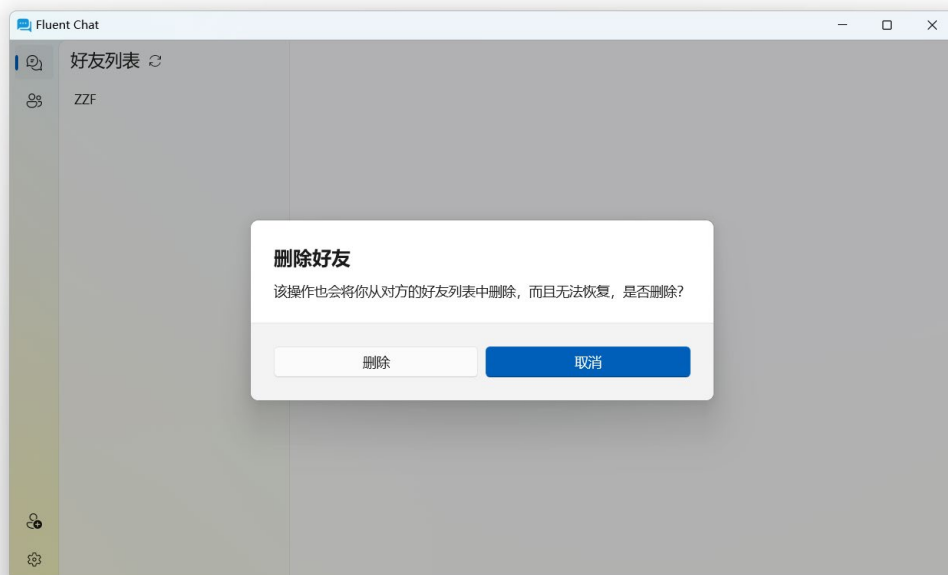
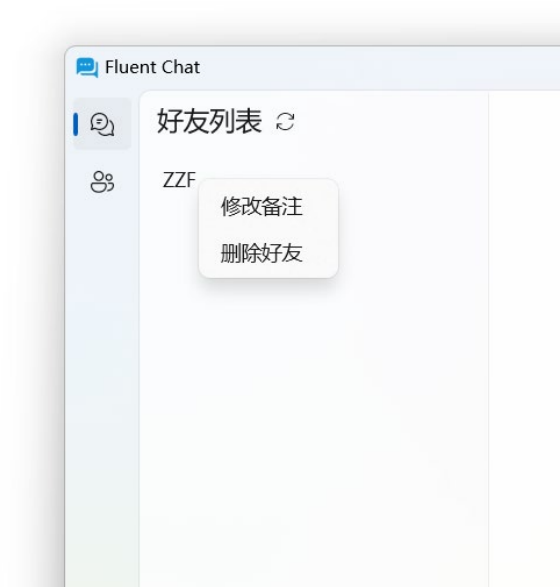


发送添加好友请求后，对面可以看到请求，点击接受可成为好友，拒绝则拒绝将其添加为好友。



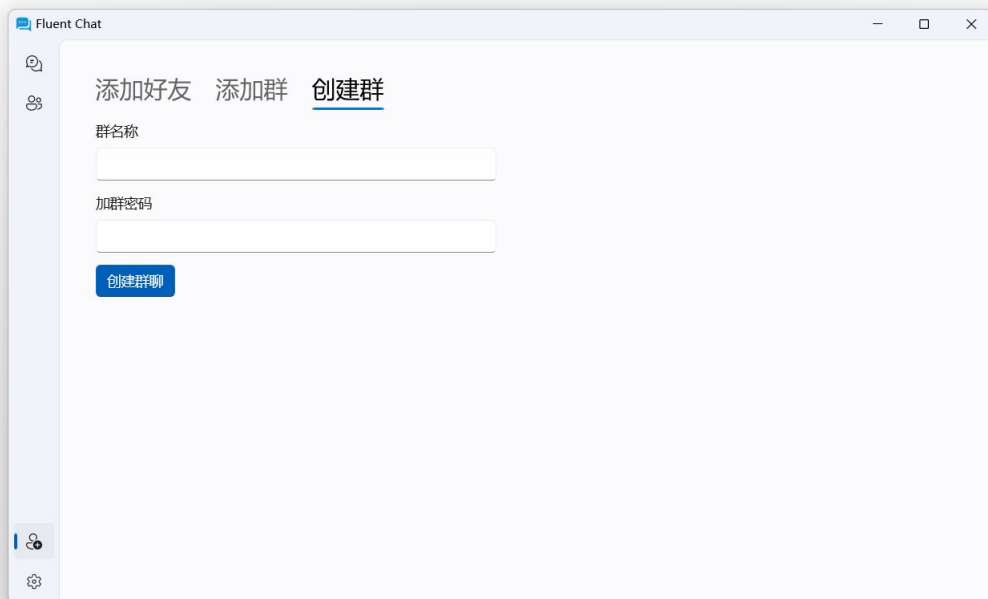
4.4 删除好友

在好友列表中右键可以删除好友，同时也会将自己从对方好友列表中删除。



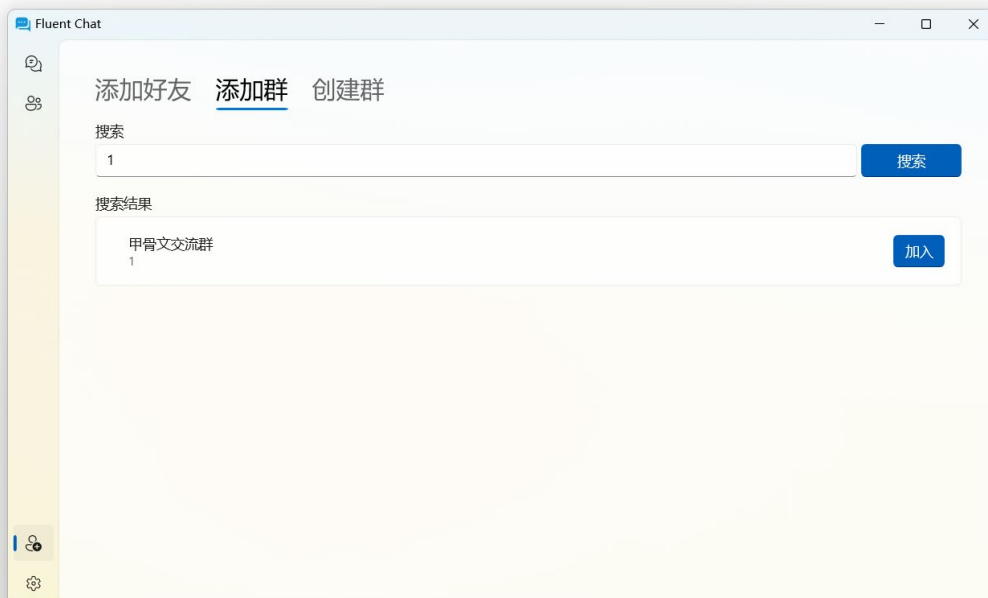
4.5 创建群聊

通过群聊名称和加群密码，可以创建一个群



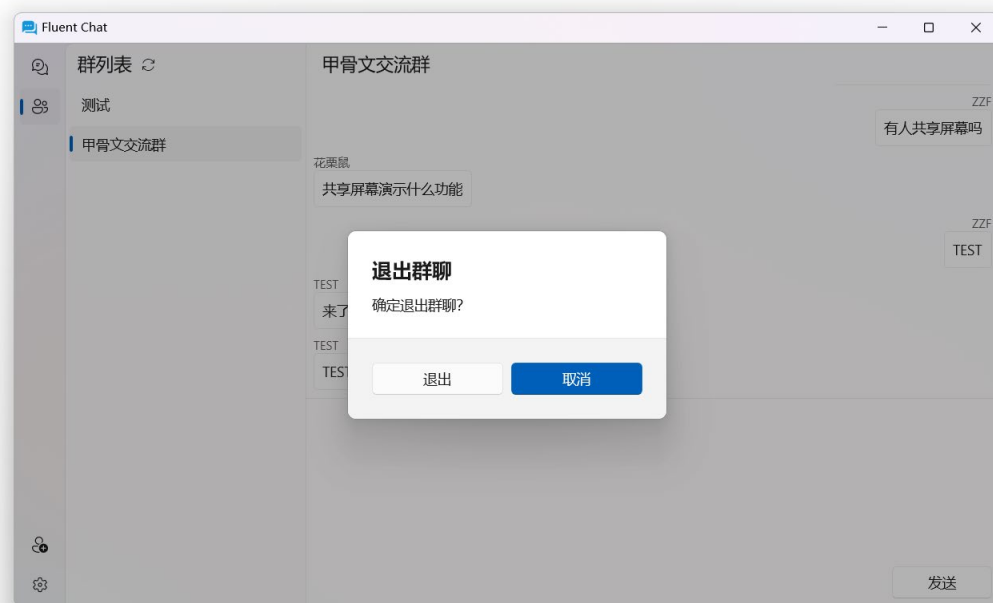
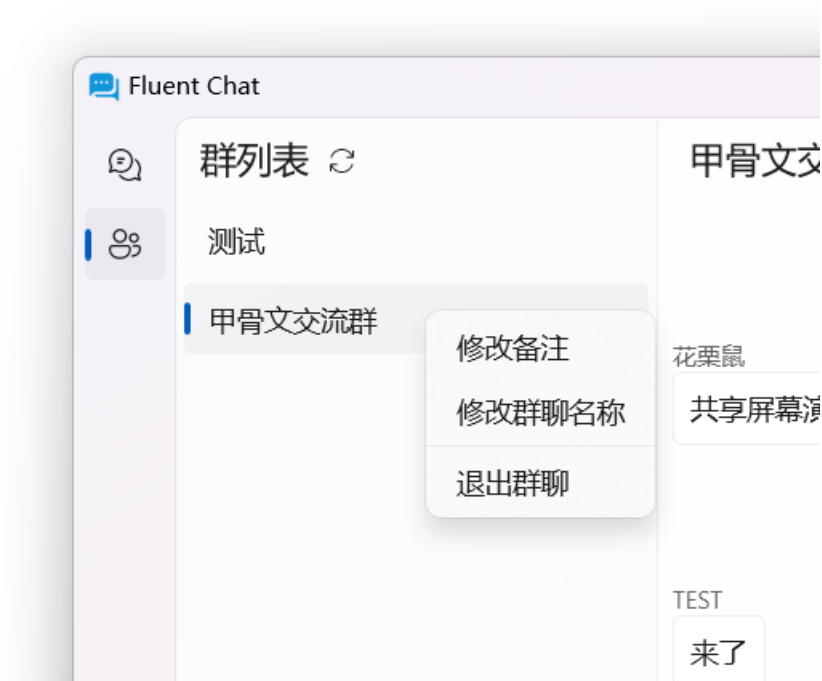
4.6 加入群聊

通过群号和群名称可以搜索到相应的群，然后输入正确的加群密码，可以加入群聊



4.7 退出群聊

在群列表右键确定后可以退出群聊。



4.8 用户管理

网页端可以管理用户

用户管理

筛选: 用户名/邮箱

昵称	邮箱	身份	操作
12345	123@qq.com	普通用户	编辑 删除
TEST	test@test.com	普通用户	编辑 删除
ZZF 3号	zhuzhifa9968@gmail.com	普通用户	编辑 删除
ZZF 2号	zhuzhifa9968@qq.com	普通用户	编辑 删除
花果山	2386255613@qq.com	普通用户	编辑 删除

< 1 2 >

通过左上角的搜索框可以筛选到相应的用户

用户管理

筛选: 用户名/邮箱

TE

昵称	邮箱
TEST	test@test.com

点击编辑可以编辑用户昵称、权限

昵称	邮箱	身份	操作
<input type="text" value="TEST"/>	test@test.com	<div>普通用户</div>	<div>保存</div> <div>取消</div>

点击删除可以删除用户

用户管理

筛选：用户名/邮箱
TE

昵称	邮箱	身份	操作
TEST	test@test.com	普通用户	编辑 删除

< 1 2 >

删除用户

确认删除“TEST”吗

[删除](#) [取消](#)