

# FlyBook即时通信服务端(Apifox + WebSocket)

## 准备工作

在开始之前，请确保项目已经启动（默认端口 `8081`）。

首先需要建SQL表。创建数据库flybook，在application.yaml中配置密码、数据库名。创建表：

代码块

```
1  create table conversation_members
2  (
3      id                bigint auto_increment
4      primary key,
5      conversation_id    bigint                not null,
6      user_id            bigint                not null,
7      last_ack_seq       bigint                default 0                null comment '已确认同步
到的序列号',
8      unread_count       int                  default 0                null comment '未读消息
数',
9      role               tinyint              default 1                null comment '角色：1=成
员，2=管理员',
10     is_muted            tinyint              default 0                null comment '免打扰',
11     is_pinned           tinyint              default 0                null comment '是否置顶',
12     joined_time         timestamp            default CURRENT_TIMESTAMP null,
13     constraint uk_conv_user
14         unique (conversation_id, user_id)
15 )
16     comment '会话成员表';
17
18  create table conversations
19  (
20      conversation_id    bigint auto_increment
21      primary key,
22      type               tinyint              default 1                not null comment
'类型：1=单聊(P2P)，2=群聊(Group)',
23      name               varchar(128)         null comment '群名
称',
24      avatar_url         varchar(255)         null comment '群头
像',
25      owner_id           bigint                default 0                null comment '群主
ID',
26      current_seq        bigint                default 0                null comment '当前会
话最新序列号',
```

```

27     last_msg_content varchar(512) default '' null comment '最新消
    息摘要',
28     last_msg_time    timestamp    default CURRENT_TIMESTAMP null comment '最新消
    息时间',
29     created_time     timestamp    default CURRENT_TIMESTAMP null
30 )
31 comment '会话表';
32
33 create table message_reactions
34 (
35     id                bigint auto_increment
36     primary key,
37     message_id        bigint                not null comment '关联消息
    ID',
38     user_id           bigint                not null comment '操作用户
    ID',
39     reaction_type     varchar(32)          not null comment '表情代
    码, 如: thumbsup, heart',
40     created_time      timestamp default CURRENT_TIMESTAMP null,
41     constraint uk_msg_user_react
42     unique (message_id, user_id, reaction_type)
43 )
44 comment '消息表情回应表';
45
46 create table messages
47 (
48     message_id        bigint auto_increment comment '全局唯一消息ID'
49     primary key,
50     conversation_id   bigint                not null,
51     sender_id         bigint                not null,
52     seq               bigint                not null comment '会话内
    序号',
53     quote_id          bigint    default 0    null comment '引用的消息
    ID (若不为0, 则是回复消息)',
54     msg_type          tinyint                not null comment
    '1=Text, 2=Image, 3=Video, 4=File, 5=TodoCard, 6=RichPost',
55     content           json                  not null comment '消息内
    容载体',
56     mentions          json                  null comment '被@用户ID
    列表',
57     is_revoked        tinyint    default 0    null comment '是否已撤
    回',
58     created_time      timestamp default CURRENT_TIMESTAMP null,
59     constraint uk_conv_seq
60     unique (conversation_id, seq)
61 )
62 comment '消息存储表';

```

```

63
64 create index idx_conv_time
65     on messages (conversation_id, created_time);
66
67 create table users
68 (
69     user_id      bigint auto_increment comment '用户唯一ID'
70     primary key,
71     username     varchar(64)                not null comment '用户
名/姓名',
72     avatar_url   varchar(255) default ''      null comment '头像链接',
73     password     varchar(255)                null comment '密码（可为
空，用于测试）',
74     created_time timestamp default CURRENT_TIMESTAMP null
75 )
76     comment '用户信息表';

```

在数据库执行以下 SQL，以模拟用户：

代码块

```

1  -- 1. 清理旧数据（防止 ID 冲突或数据干扰）
2  DELETE FROM messages;
3  DELETE FROM conversation_members;
4  DELETE FROM conversations;
5  DELETE FROM users;
6
7  -- 2. 创建三个测试用户
8  -- 用户 A: ZhangSan (ID: 1001) -> 扮演群主/发送者
9  -- 用户 B: LiSi      (ID: 1002) -> 扮演接收者/被邀请者
10 -- 用户 C: WangWu    (ID: 1003) -> 扮演路人
11
12 -- 注意: password 字段设置为 NULL，表示测试用户没有密码，登录时传空字符串即可
13 INSERT INTO users (user_id, username, avatar_url, password) VALUES
14 (1001, 'ZhangSan', 'https://api.dicebear.com/7.x/avataaars/svg?seed=Zhang',
15 NULL),
16 (1002, 'LiSi', 'https://api.dicebear.com/7.x/avataaars/svg?seed=Li', NULL),
17 (1003, 'WangWu', 'https://api.dicebear.com/7.x/avataaars/svg?seed=Wang', NULL);

```

## 场景脚本：一次完整的群聊体验

我们将模拟以下剧情：

1. LiSi (1002) 打开手机（连接 WebSocket），准备接收消息。

2. ZhangSan (1001) 创建了一个叫“周末去哪玩”的群。
3. ZhangSan 把 LiSi 拉进群。
4. ZhangSan 在群里发了一句“Hello”。
5. LiSi 收到推送，并刷新列表看到未读消息。

## 步骤 1：模拟接收者上线 (WebSocket)

这里也可以多开几个ws窗口 多建立几个在线用户，发信息选的是http，服务端推送选的是ws协议。

我们需要一个工具来模拟手机端的长连接，看着服务器推什么数据过来。

- 工具: 可使用在线工具 [PieSocket](#) 或 [WebSocket King](#) (或者直接用 Apifox 的 WebSocket 功能)。
- 操作:
  - a. 在浏览器打开上述工具。
  - b. 输入地址: `ws://localhost:8081/ws/1002`，其中1002是LiSi的user\_id。
  - c. 点击 Connect。
- 预期:
  - 工具显示状态为 `Connected` (绿色)。
  - 后端控制台日志输出：

代码块

```
1  用户 connected: 1002, 当前在线人数: 1
2  用户 connected: 1001, 当前在线人数: 2
```

⚠ 注意：测试过程中请不要关闭这个网页窗口，把它挂在旁边。

## 步骤 2：登录 (HTTP)

- 接口: `POST` `http://localhost:8081/api/users/login`
- Headers: ``Authorization`` `Bearer 1001`
- Body:

代码块

```
1  {
2    "username": "ZhangSan",
3    "password": ""
```

```
4 }
```

返回示例：

代码块

```
1  {
2      "code": 0,
3      "msg": "success",
4      "data": {
5          "userInfo": {
6              "userId": 1001,
7              "username": "ZhangSan",
8              "avatarUrl": "https://api.dicebear.com/7.x/avataaars/svg?
seed=Zhang",
9              "password": null,
10             "createTime": "2025-11-29T17:03:34"
11         },
12         "token":
"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMDAxIiwiaWF0IjoxNzY0NDExMzAxLCJleHAiOiJlE3NjUwM
TYxMDF9.TY4dhmnW5Q3AhI0fGMVkhGOxIiraofv9Ucnh7Unzm08"
13     }
14 }
```

### 步骤 3：创建群聊 (HTTP)

现在切换到 Apifox，模拟 ZhangSan (1001) 创建群组。

- 接口: POST <http://localhost:8081/api/conversations/create>
- Headers: `Authorization` Bearer
- Body (JSON): `eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMDAxIiwiaWF0IjoxNzY0NDExMzAxLCJleHAiOiJlE3NjUwM`

代码块

```
1  {
2      "ownerId": 1001,
3      "type": 2,
4      "name": "周末去哪玩"
5  }
```

- 预期:

- 返回JSON:

代码块

```
1  {
2      "code": 0,
3      "msg": "success",
4      "data": 1
5  }
```

其中 `data` (即 `conversationId`), 假设为1。

### 步骤 3: 拉人入群 (HTTP)

ZhangSan (1001) 将 LiSi (1002) 拉入群聊。这是检验“系统通知”的关键步骤。

- 接口: `POST http://localhost:8081/api/conversations/members/add`
- Body (JSON):

代码块

```
1  {
2      "conversationId": 1,
3      "inviterId": 1001,
4      "targetUserIds": [1002]
5  }
6
```

- 操作: 点击发送。
- 预期 (重要):

a. Apifox 返回 `success` :

代码块

```
1  {
2      "code": 0,
3      "msg": "success",
4      "data": null
5  }
```

b. 在ZhangSan (1001) 的WebSocket端收到 JSON 推送:

代码块

```
2      "msgType": 1,  
3      "conversationId": 1,  
4      "messageId": 1,  
5      "content": "{\"text\":\"ZhangSan 邀请 LiSi 加入了群聊\"}",  
6      "senderId": 1001,  
7      "createdTime": 1764420831006,  
8      "seq": 1  
9  }
```

## 步骤 4：发送消息 (HTTP)

ZhangSan (1001) 发言。

- 接口: POST `http://localhost:8081/api/messages/send`
- Body (JSON):

代码块

```
1  {  
2      "conversationId": 1,  
3      "text": "Hello"  
4  }
```

- 预期:
  - a. Apifox 返回成功, 且 `seq` 增加:

代码块

```
1  {  
2      "code": 0,  
3      "msg": "success",  
4      "data": {  
5          "messageId": 3,  
6          "conversationId": 1,  
7          "senderId": 1001,  
8          "seq": 2,  
9          "quoteId": null,  
10         "msgType": 1,  
11         "content": "{\"text\":\"Hello\"}",  
12         "mentions": null,  
13         "isRevoked": null,  
14         "createdTime": "2025-11-29T21:10:19.0016336"
```

```
15     }
16 }
```

b. 查看 LiSi 的 WebSocket 网页，应该收到：

代码块

```
1  {
2      "msgType": 1,
3      "conversationId": 1,
4      "messageId": 3,
5      "content": "{\"text\":\"Hello\"}",
6      "senderId": 1001,
7      "createTime": 1764421810379,
8      "seq": 2
9  }
```

## 步骤 5：查看会话列表 (HTTP)

假设 LiSi 刚才没看手机（虽然 WebSocket 收到了，但他想打开 App 看列表）。

- 接口: GET <http://localhost:8081/api/conversations/list>
- Params:
  - `userId: 1002`
- 操作: 点击发送。
- 预期:
  - 应该返回一个数组。
  - 找到名为 `周末去哪玩` 的对象。
  - 检查 `unreadCount`：应该是 1 或更多（取决于你发了几条）。
  - 检查 `lastMsgContent`：应该是 `"兄弟们，周六去爬山吗？"`。

代码块

```
1  {
2      "code": 0,
3      "msg": "success",
4      "data": [
5          {
6              "conversationId": 1,
7              "type": 2,
```



```
8         "name": "周末去哪玩",
9         "avatarUrl": null,
10        "lastMsgContent": "Hello",
11        "lastMsgTime": "2025-11-29T21:10:19",
12        "unreadCount": 3
13    },
14    {
15        "conversationId": 2,
16        "type": 2,
17        "name": "周末去哪玩",
18        "avatarUrl": null,
19        "lastMsgContent": "ZhangSan 邀请 LiSi 加入了群聊",
20        "lastMsgTime": "2025-11-29T21:09:33",
21        "unreadCount": 1
22    }
23 ]
24 }
```

---