软件工程小项目运行流程

目录

[1.登录 2](#_Toc21766727)

[1.1已有账户，直接登录 2](#_Toc21766728)

[1.2注册用户 2](#_Toc21766729)

[2.查看组队学习信息 3](#_Toc21766730)

[2.1第一次进入该界面，分页查询 3](#_Toc21766731)

[2.2根据选择的查询条件查询数据，分页查询 4](#_Toc21766732)

[2.3申请入队 4](#_Toc21766733)

[3.查看玩乐信息 5](#_Toc21766734)

[3.1第一次进入该界面，分页查询 5](#_Toc21766735)

[3.2根据选择的查询条件查询数据，分页查询 5](#_Toc21766736)

[3.3申请入队 6](#_Toc21766737)

[4.个人主页 6](#_Toc21766738)

[4.1显示个人信息 6](#_Toc21766739)

[4.2显示用户发布过的组队信息 7](#_Toc21766740)

[4.3查看用户参加过的组队信息 7](#_Toc21766741)

[4.4组队信息管理 8](#_Toc21766742)

[4.4.1发布组队信息 8](#_Toc21766743)

[4.4.2撤销组队信息 8](#_Toc21766744)

[4.4.3删除组队信息 9](#_Toc21766745)

[4.4.4审核接收到的申请消息 9](#_Toc21766746)

[4.5用户消息管理 10](#_Toc21766747)

[4.5.1显示未读消息的类型与对应的数量 10](#_Toc21766748)

[4.5.2显示“申请消息审核结果”详情 10](#_Toc21766749)

[4.5.3显示“接收到的入队申请消息”详情 11](#_Toc21766750)

[4.5.4显示“发送的入队申请信息”详情 11](#_Toc21766751)

[4.5.5撤销用户发送出去的入队申请信息 11](#_Toc21766752)

## 1.登录

### 1.1已有账户，直接登录

1.前台：

获取用户输入的‘用户名{username}’与‘密码{password}’

2.调用后台接口：

‘UserInfoController’下‘login’接口，传入‘user{username,password}’对象。

返回值：使用数字代表不同的状态：

1：登录成功

2：该用户名未注册

3：密码错误

### 1.2注册用户

1.前台：

获取用户的基本信息：

1.用户名{username}【必填】

2.密码{password}【必填】

3.昵称{nickname}【必填】

4.性别{sex}【必填】

5.所在城市{city}【必填】

6.所在学校{school}【必填】

7.所在院系{academy}

8.所在专业{major}

9.个性签名{personalized\_signature}

10.个人标签{personalized\_labels}

前台给用户设置默认的头像，修改用户头像在用户的个人主页实现，注册界面不添加此功能，该头像不传入后台。

2.调用后台接口：

‘UserInfoController’下‘insert’接口，传入 ‘userInfo’ 对象：

userInfo：{username，password，nickname，sex，city，school，academy，major，personalized\_signature，personalized\_labels}

返回值：使用数字表示不同的状态

1：注册成功

2：该昵称已被使用

3：该用户名已被使用

## 2.查看组队学习信息

### 2.1第一次进入该界面，分页查询

1.前台：

显示从后台拿到的数据。

2.调用后台接口：

‘StudyInfoController’下‘getStudyInfo’接口。传入‘username’参数，返回一个‘PageBean<StudyInfo>’给前台

PageBean：{

total：总条数

list：返回的结果集，List<StudyInfo>

pageNum：当前页

pages：总页数

pageSize：每一页的结果集的数量

size：当前页的结果集的数量

}

StudyInfo:{

id：id值

username：用户名

userSchool：用户所在的学校

actionName：活动名称

type：模式，1：线下，2：线上

maxMember：最大组队人数，包括用户

leftMember：剩余空位

label：标签

time：学习时间

site：学习地点

ps：备注

date：发布时间

valid：是否有效

}

3.后台代码实现：

根据传入的‘username’获取该用户所在的学校‘school’，根据模式的不同，各提取10条最新的数据。注意查询条件

1.查询方式为分页查询

2.查询线下模式时，表中school字段应与用户所在学校一致。

3.查询到的数据要为最新的

拿到数据之后，补全给条信息中，用户头像的地址，并将数据进行乱序处理，返回一个‘PageBean<StudyInfo>’给前台

### 2.2根据选择的查询条件查询数据，分页查询

1.前台：

提供选择框，保存用户选择的条件，

1：type，模式，【使用数字代替模式，1：线下，2：线上，必选】

2：label，学习标签，【可不选，选择只能是单选】

3：maxMember，组队人数，【可不选，也可指定组队人数，最多为6人。】

2.调用后台接口：

调用‘StudyInfoController’下‘selectStudyInfo’，传入‘queryInfo’对象【QueryStudyInfo{type，username，label，maxMember}】，‘pageNum’【当前页】，‘pageSize’【当前页显示的数据量】，返回一个‘PageBean<StudyInfo>’给前台。

3.后台代码实现：

后台根据传入的条件进行分页查询，返回一个‘PageBean<StudyInfo>’给前台。注意查询条件：

1.线下模式需要获取当前用户所在的学校，查询到的数据中的‘school’字段的值与该用户所在的学校一致。

2.查询到的结果应该倒序排列，因为向用户显示的数据顺序应该是最新插入的数据排在最前

3.使用分页操作对查询到的数据进行处理。

### 2.3申请入队

1.前台：

点击按钮申请组队，注意，不可申请无效的组队。前台注意，对于无效的组队信息，设置按钮无效或者不显示。

2.调用后台接口：

调用后台‘auditInfoController’下‘insert’接口，传入‘AuditInfo’对象。

AuditInfo：{

sender：申请人，填入当前用户名

receiver：接收者，填入选中的组队信息的发布人

actionType：活动类型，使用数字代替，1：线下学习，2：线上学习，3：吃喝玩乐

actionId：活动id，获取该活动的id

auditStatus：申请状态，填入0

auditResult：申请结果，填入0

}

3.后台代码实现：

1.向auditInfo表中插入一条数据

## 3.查看玩乐信息

### 3.1第一次进入该界面，分页查询

1.前台：

显示从后台拿到的数据

2.调用后台接口

调用‘enjoyInfoController’下‘getEnjoyInfo’接口，传入当前用户的‘username’，返回一个‘PageBean<EnjoyInfo>’数组，长度为20条

EnjoyInfo：{

`id`：主键,  
`username`：用户名,

`user\_city`：用户所在城市,

`actionName`：活动名  
`label`：标签,  
`sex\_specific` 性别限制，性别限制分三种，用数字代替，1：只限异性，2：只限同性，2：无限制  
`time`：玩乐时间,  
`site`：玩乐地点

`max\_member`：最大组队人数

`left\_member`：剩余空位  
`ps`：备注

`date`：消息发布的时间

`valid`：消息是否有效

}

3.后台代码实现：

注意查询条件：

1.用户只能查询到与自己城市一致的人发布的玩乐信息。

2.查询20条最新的数据

3.分页查询

### 3.2根据选择的查询条件查询数据，分页查询

1.前台：

提供选择框，保存用户选择的条件，

1：label，玩乐标签，【可不选，选择只能是单选】

2：maxMember，组队人数，【可不选，也可指定组队人数，最多为6人。】

3：sexSpecific，性别限制，用数字标注，1：只限异性，2：只限同性，3：无限制。

2.调用后台接口：

调用‘EnjoyInfoController’下‘selectEnjoyInfo’，传入‘queryInfo’对象【QueryStudyInfo{username，label，maxMember，sexSpecific }】，‘pageNum’【当前页】，‘pageSize’【当前页显示的数据量】，返回一个‘PageBean<EnjoyInfo>’对象

3.后台代码实现：

后台根据传入的条件进行分页查询，返回一个‘PageBean<EnjoyInfo>’给前台。注意查询条件：

1.查询到的数据中的城市（city字段）要与用户所在的城市一致

2.查询到的结果应该倒序排列，因为向用户显示的数据应该是最新插入的数据

3.使用分页操作对查询到的数据进行处理。

### 3.3申请入队

1.前台：

点击按钮申请组队，注意，不可申请无效的组队。前台注意，对于无效的组队信息，设置按钮无效或者不显示。

2.调用后台接口：

调用后台‘auditInfoController’下‘insert’接口，传入‘AuditInfo’对象。

AuditInfo：{

sender：申请人，填入当前用户名

receiver：接收者，填入选中的组队信息的发布人

actionType：活动类型，使用数字代替，1：线下学习，2：线上学习，3：吃喝玩乐

actionId：活动id，获取该活动的id

auditStatus：申请状态，填入0

auditResult：申请结果，填入0

}

## 4.个人主页

### 4.1显示个人信息

1.前台：

1.显示个人信息，注意界面的布局

2.设置图标，提示用户是否有接收到新的消息

2.调用后台接口：

1.获取用户个人信息

调用‘UserInfoController’下‘getUserInfo’接口，传入参数‘username’，后台返回一个‘UserInfo’对象。

userInfo：{

username：用户名，

nickname：昵称，

sex：性别，

city：所在城市，

school：所在学校，

academy：所在院系，

major：所学专业，

personalized\_signature：个性签名，

personalized\_labels：个人标签

}

2.是否有新的未读消息

调用‘userMessageController’下‘getMessageCountWithNotRead’，传入‘username’参数，返回一个布朗值。

3.后台逻辑实现：

1.补全用户头像的地址，方便前台做数据渲染

2.使返回给前台的 ‘password’为null值

### 4.2显示用户发布过的组队信息

1.前台

显示从后台拿到的数据。注意数据渲染，未失效的组队信息与失效的组队信息的显示应该要有明显的区别。

2.调用后台接口：

1.查询用户发布过的组队学习信息

调用‘StudyInfoController’下‘getStudyInfoByUserIssue’接口，传入用户名，返回一个‘PageBean<StudyInfo>’对象。

2.查询用户发布过的玩乐信息

调用‘EnjoyController’下‘getEnjoyInfoByUserIssue’接口，传入用户名，返回一个‘PageBean<EnjoyInfo>’对象。

3.后台代码实现：

1.分页查询

2.后台对数据进行排序处理，将插到的消息分为‘未失效’，‘失效’两类，‘未失效的信息’排在‘失效信息’前面，对于各类信息，按照时间‘由近及远’的顺序进行排序。

### 4.3查看用户参加过的组队信息

1.前台

显示从后台拿到的数据。注意数据渲染，未失效的组队信息与失效的组队信息的显示应该要有明显的区别。

2.调用后台接口：

1.查询用户参与过的组队学习信息

调用‘StudyInfoController’下‘getStudyInfoByUserAttend’接口，传入参数：用户名【username】，返回一个‘PageBean<StudyInfo>’对象。

2.查询用户发布过的玩乐信息

调用‘EnjoyController’下‘getEnjoyInfoByUserAttend’接口，传入参数：用户名【username】，返回一个‘PageBean<EnjoyInfo>’对象。

3.后台代码实现：

1.分页查询，分页用于查询‘team\_info’表

2.多表查询，先‘team\_info’表中根据‘username’，‘action\_type’查询对应的‘action\_id’，在根据‘action\_id’与‘action\_type’从‘study\_info’或者‘enjoy\_info’的拿到数据。

3.对于字段‘valid’的赋值，根据‘action\_id’与‘action\_type’从‘study\_info’或者‘enjoy\_info’中提取数据，两表中存在对应数据的，直接提取，两表中不存在对应数据的，将‘valid’赋值为‘2’，标明该条组队信息已被删除。

### 4.4组队信息管理

#### 4.4.1发布组队信息

1.前台：

1.发布组队学习信息界面

提供填写活动信息的界面，将信息封装成一个‘StudyInfo’对象

2.发布玩乐信息界面

提供填写玩乐信息的界面，将信息封装为一个‘EnjoyInfo’对象

2.调用后台接口

1.发布组队学习信息

调用‘StudyInfoController’下‘insert’方法 ，传入一个‘StudyInfo’对象。

2.发布玩乐信息

调用‘EnjoyInfoController’下‘insert’方法，传入一个‘EnjoyInfo’对象。

#### 4.4.2撤销组队信息

1.前台：

设计按钮使得用户能执行此功能，只能撤销有效的组队信息

2.调用后台接口

1.撤销发布的组队学习信息：

调用后台接口‘StudyInfoController’下‘withdraw’接口，传入要撤销的消息的‘id’。

2.撤销发布的玩乐信息：

调用后台接口‘EnjoyInfoController’下‘withdraw’接口，传入要撤销的消息的‘id’。

3.后台代码实现

1.撤销操作——修改对应的表的‘valid’字段的值为‘0’。

2.联动申请表‘audit\_info’与消息表‘user\_message’。向申请加入该队，但是申请请求未审核的用户发送“该组队信息已撤销，不在接收组队申请”的消息。同时修改该用户的审核状态为‘1’，审核结果为‘0’。

#### 4.4.3删除组队信息

1.前台：

设计按钮使得用户能执行此功能，只能删除用户自己发布的组队信息

2.调用后台接口

1.删除发布的组队学习信息：

调用后台接口‘StudyInfoController’下‘delete’接口，传入要撤销的消息的‘id’。

2.删除发布的玩乐信息：

调用后台接口‘EnjoyInfoController’下‘delete’接口，传入要撤销的消息的‘id’。

3.后台代码实现

1.删除操作。

2.联动申请表‘audit\_info’与消息表‘user\_message’。向申请加入该队，但是申请请求未审核的用户发送“该组队信息已删除，不在接收组队申请”的消息。同时修改该用户的审核状态为‘1’，审核结果为‘0’。

#### 4.4.4审核接收到的申请消息

1.前台：

提供按钮，确定审核通过与否

2.调用后台接口

调用‘AuditInfoController’下‘updateAuditInfo’接口，传入参数：‘id’申请信息的id，‘result’：审核结果。

使用数字代表审核结果，‘1’表示通过申请，‘0’表示拒绝申请。

3.后台代码实现

1.修改‘audit\_info’表中的‘audit\_status’值为‘1’，表示‘已审核’。修改‘audit\_result’为传入的‘result’

2.联动‘user\_message’表，审核通过或者失败，都需要由后台向申请者发送一条信息，提示其审核失败或成功

3.当审核结果为成功时，获取活动的类型与id，向‘team\_info’表中插入一条相应的数据。同时，在对应的活动表中，字段‘left\_size’值‘减1’。同时进行判定，当‘left\_size’的值变为‘0’时，修改活动表中的字段‘valid’值为‘0’，对于该活动的其他已递交申请但申请为审核的用户，将全部设置‘audit\_status’值为‘1’，‘audit\_result’值为‘0’

### 4.5用户消息管理

#### 4.5.1显示未读消息的类型与对应的数量

1.前台：

显示未读消息的类型以及对应的数量，目前有三种消息类型，一类为“接收到的入队申请消息”用数字“1”标识，表示其他用户想加入你发起的组队。一类为“申请消息审核结果”，用数字“2”标识，表示你发送给其他用户的申请消息的审核结果。一类为“发送的入队申请消息”，用数字“3”标识，表示你发送给其他用户的申请消息，而其他用户未审核。

2.调用后台接口：

调用‘userMessageController’下‘getMsgCountByType’传入参数：‘username’用户名。返回一个‘List<MessageInfo>’对象。

MessageInfo{

type：消息的类型

sum：该类消息的总数

unreadCount：该类消息的未读数量

}

3.后台代码实现：

1.入队申请信息在‘audit\_info’表中

#### 4.5.2显示“申请消息审核结果”详情

1.前台：

该界面用于显示用户申请加入其它人发布的组队信息的申请审核结果。“申请成功”与“申请失败”两类消息的显示应该明显区分开。

2.调用后台接口

调用‘UserMessageController’下‘getAuditResult’接口，传入参数：‘username’【用户名】。返回一个‘List<UserMessage>’对象。

UserMessage{

id：消息的主键

sender：消息的发送者

receiver：消息的接受者

type：消息的类型，‘申请成功’用‘3’表示，‘申请失败’用‘2’表示

context：消息的内容

date：审核时间

}

3.后台代码实现

1.对消息进行排序，将全部的‘申请成功’消息集中在‘申请失败’信息的前面。

#### 4.5.3显示“接收到的入队申请消息”详情

1.前台：

显示接收到的入队申请信息，“已审核”与“未审核”的消息的显示应该要有明显的区别

2.调用后台接口

调用‘AuditInfoController’下‘getAuditInfoWithReceived’接口，传入参数：username【用户名】，返回一个‘List<AuditInfo>’。

3.后台代码实现

查询用户接受到的信息，对查询到的数据进行排序处理，‘未审核’的消息集中在‘已审核’消息的前面。

#### 4.5.4显示“发送的入队申请信息”详情

1.前台：

显示用户发送出去的，未被审核的申请信息。用户能对该类信息进行“撤销”操作

2.调用后台接口

调用‘AuditInfoController’下‘getAuditInfoWithSend’接口，传入参数：username【用户名】，返回一个‘List<AuditInfo>’。

3.后台代码实现

查询用户发出去的申请信息。

#### 4.5.5撤销用户发送出去的入队申请信息

1.前台：

提供按钮，使用户能进行该类操作。撤销即删除

2.调用后台接口

调用‘AuditInfoController’下‘delete’接口，传入参数：id【消息的主键】