# UserServer 设计与开发

## 版权说明



对比特项目感兴趣，可以联系这个微信。

## 代码 & 板书链接

https://gitee.com/bitedu-tech/cpp-chatsystem

# 功能设计

用户管理子服务，主要用于管理用户的数据，以及关于用户信息的各项操作，因此在上述项目功能中，用户子服务需要提供以下接口：

1. 用户注册：用户输入用户名(昵称)，以及密码进行用户名的注册
2. 用户登录：用户通过用户名和密码进行登录
3. 短信验证码获取：当用户通过手机号注册或登录的时候，需要获取短信验证码
4. 手机号注册：用户输入手机号和短信验证码进行手机号的用户注册
5. 手机号登录：用户输入手机号和短信验证码进行手机号的用户登录
6. 用户信息获取：当用户登录之后，获取个人信息进行展示
7. 头像修改：设置用户头像
8. 昵称修改：设置用户昵称
9. 签名修改：设置用户签名
10. 手机号修改：修改用户的绑定手机号

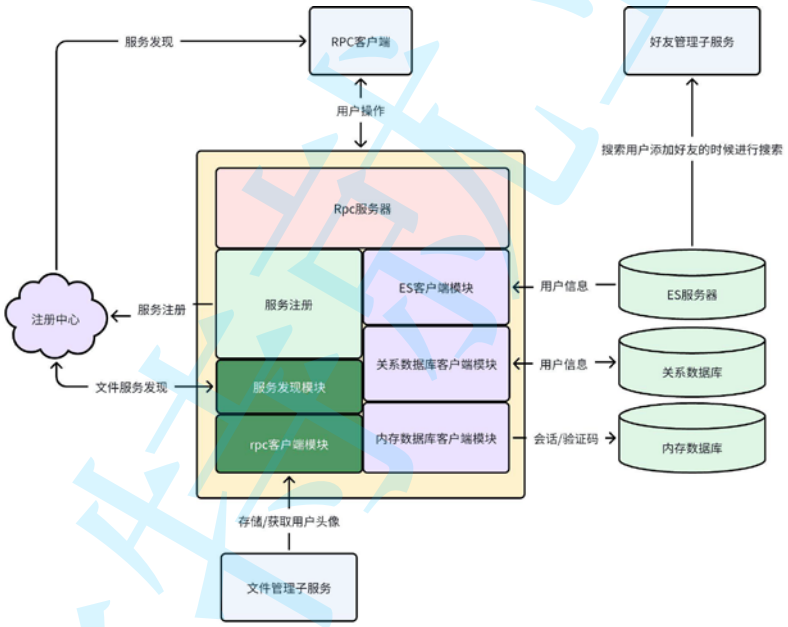
# 模块划分

1. 参数/配置文件解析模块：基于 gflags 框架直接使用进行参数/配置文件解析。
2. 日志模块：基于 spdlog 框架封装的模块直接使用进行日志输出。
3. 服务注册模块：基于 etcd 框架封装的注册模块直接使用，进行聊天消息存储子服务的注册。
4. 数据库数据操作模块：基于 odb-mysql 数据管理封装的模块，实现关系型数据库中数据的操作。
    a. 用户进行用户名/手机号注册的时候在数据库中进行新增信息
    b. 用户修改个人信息的时候修改数据库中的记录
    c. 用户登录的时候，在数据库中进行用户名密码的验证
5. redis 客户端模块：基于 redis++封装的客户端进行内存数据库数据操作
    a. 当用户登录的时候需要为用户创建登录会话，会话信息保存在 redis 服务器中。
    b. 当用户手机号进行获取/验证验证码的时候，验证码与对应信息保存在 redis 服

务器中

6. rpc 服务模块：基于 brpc 框架搭建 rpc 服务器。
7. rpc 服务发现与调用模块：基于 etcd 框架与 brpc 框架封装的服务发现与调用模块，
   a. 连接文件管理子服务：获取用户信息的时候，用户头像是通过文件的形式存储在文件子服务中的。
   b. 连接消息管理子服务：在打开聊天会话的时候，需要获取最近的一条消息进行展示。
8. ES 客户端模块：基于 elasticsearch 框架实现访问客户端，向 ES 服务器中存储用户简息，以便于用户的搜索
9. 短信平台客户端模块：基于短信平台 SDK 封装使用，用于向用户手机号发送指定验证码。

## 功能模块示意图：




# 数据管理

## 关系数据库数据管理：

在关系型数据库中，对于用户子服务来说，总体只进行了一个信息数据的存储与管理，那就是用户信息数据，因此只需要构建好用户信息表，提供好对应的操作即可。

## 用户数据表：

包含字段：

1. 主键 ID：自动生成
2. 用户 ID：用户唯一标识
3. 用户昵称：用户的昵称，也可用作登录用户名
4. 用户签名：用户对自己的描述吧...
5. 登录密码：登录验证
6. 绑定手机号：用户可以绑定手机号，绑定后可以通过手机号登录、
7. 用户头像文件 ID：头像文件存储的唯一标识，具体头像数据存储在文件子服务器中。

提供的操作：

1. 通过昵称获取用户信息
2. 通过手机号获取用户信息
3. 通过用户 ID 获取用信息
4. 新增用户
5. 更新用户信息

### ODB 映射数据结构：

```
C++
#include <string>
#include <cstddef>
#include <odb/core.hxx>
#include <odb/nullable.hxx>

#pragma db object
class user {
    public:
        user () {}
    private:
        friend class odb::access;

        #pragma db id auto
        unsigned long _id;
        #pragma db unique type("VARCHAR(127)")
        std::string _user_id;
        #pragma db unique type("VARCHAR(63)")
```

```
        odb::nullable<std::string> _nickname;
        #pragma db type("VARCHAR(255)")
        odb::nullable<std::string> _passwd;
        #pragma db type("VARCHAR(127)")
        odb::nullable<std::string> _avatar_id;
        #pragma db unique type("VARCHAR(15)")
        odb::nullable<std::string> _phone_number;
        #pragma db type("VARCHAR(255)")
        odb::nullable<std::string> _description;
};
```

## 内存数据库数据管理：

## 会话信息映射键值对：

> 映射类型：字符串键值对映射
>
> 映射字段：
>
> 1. 会话 ID(key) - 用户 ID(val)：便于通过会话 ID 查找用户 ID，进行后续操作时的连接身份识别鉴权
>    a. 在用户登录的时候新增数据，
>    b. 在用户登录后的操作时进行有无验证及查询
>    c. 该映射数据在用户退出登录的时候删除（目前并未提供实现）
> 2. 用户 ID(key) - 空(val)：这是一个用户登录状态的标记，用于避免同时重复登录
>    a. 在用户登录的时候新增数据
>    b. 在用户连接断开的时候删除数据

## 验证码信息映射键值对：

> 映射类型：字符串键值对映射
>
> 映射字段：
>
> 1. 验证码 ID(key) - 验证码(val)： 用于生成一个验证码 ID 和验证码，
>    a. 在用户获取短信验证码的时候新增数据。
>    b. 验证码通过短信平台发送给用户手机，
>    c. 而验证码 ID 直接响应发送给用户，用户登录的时候通过这两个信息进行验

> 证。
>
> d. 该映射字段需要设置一个 60s 过期自动删除的事件，以及在验证完毕后删除。

## 文档数据库数据管理：

用户信息的用户 ID，手机号，昵称字段需要在 ES 服务器额外进行一份存储，其目的是因为有用户搜索的功能，用户搜索通常会是一种字符串的模糊匹配方式，用传统的关系型数据库进行模糊匹配效率会极差，因此采用 ES 服务对索引字段进行分词后构建倒排索引，根据关键词进行搜索，效率会大大提升。

## 创建用户索引

```C++
POST /user/_doc
{
    "settings" : {
        "analysis" : {
            "analyzer" : {
                "ik" : {
                    "tokenizer" : "ik_max_word"
                }
            }
        }
    },
    "mappings" : {
        "dynamic" : true,
        "properties" : {
            "nickname" : {
                "type" : "text",
                "analyzer" : "ik_max_word"
            },
            "user_id" : {
                "type" : "keyword",
                "analyzer" : "standard"
            },
            "phone" : {
                "type" : "keyword",
                "analyzer" : "standard"
            },
            "description" : {
                "type" : "text",
```

```
                "index": "not_analyzed"
            },
            "avatar_id" : {
                "type" : "text",
                "index": "not_analyzed"
            }
        }
    }
}
```

## 新增测试数据

```
C++
POST /user/_doc/_bulk
{"index":{"_id":"1"}}
{"user_id" : "USER4b862aaa-2df8654a-7eb4bb65-
e3507f66","nickname" : "昵称 1","phone" : "手机号 1","description" :
"签名 1","avatar_id" : "头像 1"}
{"index":{"_id":"2"}}
{"user_id" : "USER14eeeaa5-442771b9-0262e455-
e4663d1d","nickname" : "昵称 2","phone" : "手机号 2","description" :
"签名 2","avatar_id" : "头像 2"}
{"index":{"_id":"3"}}
{"user_id" : "USER484a6734-03a124f0-996c169d-
d05c1869","nickname" : "昵称 3","phone" : "手机号 3","description" :
"签名 3","avatar_id" : "头像 3"}
{"index":{"_id":"4"}}
{"user_id" : "USER186ade83-4460d4a6-8c08068f-
83127b5d","nickname" : "昵称 4","phone" : "手机号 4","description" :
"签名 4","avatar_id" : "头像 4"}
{"index":{"_id":"5"}}
{"user_id" : "USER6f19d074-c33891cf-23bf5a83-
57189a19","nickname" : "昵称 5","phone" : "手机号 5","description" :
"签名 5","avatar_id" : "头像 5"}
{"index":{"_id":"6"}}
{"user_id" : "USER97605c64-9833ebb7-d0455353-
35a59195","nickname" : "昵称 6","phone" : "手机号 6","description" :
"签名 6","avatar_id" : "头像 6"}
```

## 进行搜索测试

```C++
GET /user/_doc/_search?pretty
{
    "query": {
        "match_all": {}
    }
}
```

```C++
GET /user/_doc/_search?pretty
{
    "query" : {
        "bool" : {
            "must_not" : [
                {
                    "terms" : {
                        "user_id.keyword" : [
                            "USER4b862aaa-2df8654a-7eb4bb65-
e3507f66",
                            "USER14eeeaa5-442771b9-0262e455-
e4663d1d",
                            "USER484a6734-03a124f0-996c169d-
d05c1869"
                        ]
                    }
                }
            ],
            "should" : [
                {
                    "match" : {
                        "user_id" : "昵称"
                    }
                },
                {
                    "match" : {
                        "nickname" : "昵称"
                    }
                },
                {
                    "match" : {
                        "phone" : "昵称"
                    }
                }
```

```
            ]
        }
    }
}
```

### 删除用户索引

```
C++
DELETE /user
```

# 接口实现流程

## 用户注册

1. 从请求中取出昵称和密码
2. 检查昵称是否合法（只能包含字母，数字，连字符-，下划线_，长度限制 3~15 之间）
3. 检查密码是否合法（只能包含字母，数字，长度限制 6~15 之间）
4. 根据昵称在数据库进行判断是否昵称已存在
5. 向数据库新增数据
6. 向 ES 服务器中新增用户信息
7. 组织响应，进行成功与否的响应即可。

## 用户登录

1. 从请求中取出昵称和密码
2. 通过昵称获取用户信息，进行密码是否一致的判断
3. 根据 redis 中的登录标记信息是否存在判断用户是否已经登录。
4. 构造会话 ID，生成会话键值对，向 redis 中添加会话信息以及登录标记信息
5. 组织响应，返回生成的会话 ID

## 获取短信验证码

1. 从请求中取出手机号码
2. 验证手机号码格式是否正确（必须以 1 开始，第二位 3~9 之间，后边 9 个数字字

符）

3. 生成 4 位随机验证码
4. 基于短信平台 SDK 发送验证码
5. 构造验证码 ID，添加到 redis 验证码映射键值索引中
6. 组织响应，返回生成的验证码 ID

## 手机号注册

1. 从请求中取出手机号码和验证码
2. 检查注册手机号码是否合法
3. 从 redis 数据库中进行验证码 ID-验证码一致性匹配
4. 通过数据库查询判断手机号是否已经注册过
5. 向数据库新增用户信息
6. 向 ES 服务器中新增用户信息
7. 组织响应，返回注册成功与否

## 手机号登录

1. 从请求中取出手机号码和验证码 ID，以及验证码。
2. 检查注册手机号码是否合法
3. 从 redis 数据库中进行验证码 ID-验证码一致性匹配
4. 根据手机号从数据数据进行用户信息查询，判断用用户是否存在
5. 根据 redis 中的登录标记信息是否存在判断用户是否已经登录。
6. 构造会话 ID，生成会话键值对，向 redis 中添加会话信息以及登录标记信息
7. 组织响应，返回生成的会话 ID

## 获取用户信息

1. 从请求中取出用户 ID
2. 通过用户 ID，从数据库中查询用户信息
3. 根据用户信息中的头像 ID，从文件服务器获取头像文件数据，组织完整用户信息
4. 组织响应，返回用户信息

## 设置头像

1. 从请求中取出用户 ID 与头像数据
2. 从数据库通过用户 ID 进行用户信息查询，判断用户是否存在
3. 上传头像文件到文件子服务，
4. 将返回的头像文件 ID 更新到数据库中
5. 更新 ES 服务器中用户信息
6. 组织响应，返回更新成功与否

## 设置昵称

1. 从请求中取出用户 ID 与新的昵称
2. 判断昵称格式是否正确
3. 从数据库通过用户 ID 进行用户信息查询，判断用户是否存在
4. 将新的昵称更新到数据库中
5. 更新 ES 服务器中用户信息
6. 组织响应，返回更新成功与否

## 设置签名

1. 从请求中取出用户 ID 与新的签名
2. 从数据库通过用户 ID 进行用户信息查询，判断用户是否存在
3. 将新的签名更新到数据库中
4. 更新 ES 服务器中用户信息
5. 组织响应，返回更新成功与否

## 设置绑定手机号

1. 从请求中取出手机号码和验证码 ID，以及验证码。
2. 检查注册手机号码是否合法
3. 从 redis 数据库中进行验证码 ID-验证码一致性匹配
4. 根据手机号从数据数据进行用户信息查询，判断用用户是否存在
5. 将新的手机号更新到数据库中

6. 更新 ES 服务器中用户信息
7. 组织响应，返回更新成功与否