Quiz前后端管理系统(项目大作业说明文档示例)

姓名(组长)：*　学号：*　班级：*

姓名：*　学号：*　班级：*

项目介绍、背景及意义：

本项目主要目的展示现代前后端的开发流程，通过本案例，让学生对常见的前端开发，及后端开发技术框架有一定的熟悉。为了屏蔽复杂业务逻辑的干扰，本项目选择一个简单的Quiz答题应用，只包含登录、注册、做题三个最基本功能，后端的管理端也仅包含用户管理和题目管理的功能。

后端也是采用成熟的MySQL及Java Spring等业界广泛使用的技术栈，前端使用Vue和Element前端框架。为了加深从原生html/css/javascript到Vue等现代前端框架的迁移，将用户的前端使用原生html开发，而将管理员的前端使用Vue开发，使得能够体会两者的差异，及现代前端框架相对于原生html的优势。

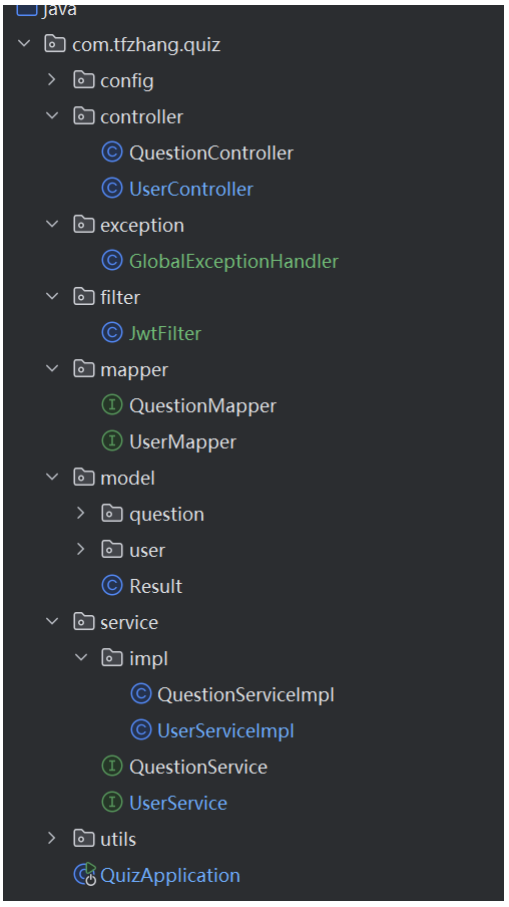
另外，后端开发中会引入很多重要概念，比如跨域访问、访问鉴权等，同时，除了开发及测试，项目还要求部署到Ubuntu/Nginx系统，这种原生的部署方式，可以与之前的课程《Linux系统实践与编程开发》相联系，承前启后，完成闭环。

技术栈及架构：

Quiz前后端管理系统，主要包括如下的前后端：

- 基于Vue2/Element UI框架的管理端前端；
- 基于html/css/js的用户Quiz测试前端；
- 基于Java Spring Web的后端；
- 部署平台：VirtualBox/Ubuntu/Nginx/Mysql;

Java后端采用经典的Controller/Service/Mybatis层次结构，数据库采用Mysql，下图是Java后端的代码结构图：



两个前端均部署到Nginx服务器的/var/www目录下；Nginx反向代理模式，将来自前端的请求转发给Java后端，同时Nginx负责前后端的跨域访问。

关键技术及解决方案（此处说明每个关键的实现人）：

由于本项目是入门演练性质的项目，总体上是对数据增删改查及前端显示，没有特别的技术难点。一定要提的话，个人觉得JWT-token访问鉴权和Nginx的跨域配置，这两块可能是学生容易有疑问的。

1、实现JWT-token（组员1负责）

要实现JWT-token，先理解概念原理，这方面可以从黑马程序员的视频入手，这里不赘述。

从前后端的交互流程看，JWT-token交互的几个关键时间点：

a) 登录的时候；根据用户信息，生成JWT-token;

b) 正常的api访问；后端读取前端发送过来的token，验证是否放行；

c) 退出登录；清理前端的token；

登录时，生成JWT-token的对应代码如下：

```
            User userResult = userService.login(username, password);
            if(userResult!=null){
                Claims claims = Jwts.claims();
                claims.put("id", userResult.getId());
                claims.put("username", userResult.getUserName());

                String token = JwtUtil.generateTokenWithClaims(claims);
                Result result = Result.success("用户登录成功");
                result.setData(token);
                return result;
            }else{
                return Result.error("用户登录失败");
            }
```

正常访问api时代码如下，验证是否可以放行：

```
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain
chain) throws IOException, ServletException {
        System.out.println("JwtFilter拦截到请求");

        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) res;


        //对于预检的options的放行；
        if ("OPTIONS".equalsIgnoreCase(request.getMethod())) {
            response.setStatus(HttpServletResponse.SC_OK);
            chain.doFilter(req, res);
            return;
        }

        String url=request.getRequestURL().toString();

        //2、判断url中是否包含login或register，如果包含，则说明是登录或注册操作，放行；
        if(url.contains("login") || url.contains("register")){
            chain.doFilter(req, res);
            return;
        }

        //3、获取请求头中的令牌(token)
        String token = request.getHeader("token");

        //4、判断令牌是否存在，如果不存在，返回未登录信息。
        if (!StringUtils.hasLength(token)){
            response.setStatus(HttpServletResponse.SC_UNAUTHORIZED); // 设置状态码
为 401
            Result result = Result.error("NOT_LOGIN");

            //手动将对象转为json，并传回前端；
            String noLogin= JSONObject.toJSONString(result);
            response.setContentType("application/json;charset=UTF-8");
            res.getWriter().write(noLogin);
            return;
        }
```

```
        //5、解析token，如果解析失败，返回未登录信息。
        try {
            JwtUtil.parseToken(token);//只要解析不成功，就说明有问题；
        }catch(Exception e){
            e.printStackTrace();
            response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
            Result result = Result.error("NOT_LOGIN");
            //手动将对象转为json，并传回前端；
            String noLogin=JSONObject.toJSONString(result);
            response.setContentType("application/json;charset=UTF-8");
            res.getWriter().write(noLogin);
            return;
        }
        //什么都对，就放行；
        chain.doFilter(req, res);
    }
```

退出后，前端需要清楚token：

```
//此处粘贴对应代码。
```

2、实现Nginx的跨域（组员2负责）

要实现Nginx跨域，可以对Nginx做如下的配置：

```
server {
    listen 80;
    server_name user-backend.66bond.com;

    location /api/ {
        proxy_pass http://127.0.0.1:8080/api/;

        add_header 'Access-Control-Allow-Origin' $http_origin;
            add_header 'Access-Control-Allow-Credentials' 'true';
            add_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';
            add_header Access-Control-Allow-Headers '*';
            if ($request_method = 'OPTIONS') {
                add_header 'Access-Control-Allow-Credentials' 'true';
                add_header 'Access-Control-Allow-Origin' $http_origin;
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-
Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range';
                add_header 'Access-Control-Max-Age' 1728000;
                add_header 'Content-Type' 'text/plain; charset=utf-8';
                add_header 'Content-Length' 0;
                return 204;
            }
    }
}
```

组内每个人负责的技术点及开发工作：

请根据自己项目的实际情况书写；

附录：

（保存有最终前端和后端代码的github仓库地址）

要求开发过程中，有阶段性的git commit/push；