

浙江大学

本科实验报告

课程名称: B/S 体系软件设计

姓 名: 彭子帆

学 院: 计算机科学与技术学院

系: 软件工程

专 业: 软件工程

学 号: 3170105860

指导教师: 胡晓军

2020 年 05 月 23 日

浙江大学实验报告

课程名称: B/S 体系软件设计 实验类型: 软件系统开发

实验项目名称: 调查问卷网站

学生姓名: 彭子帆 专业: 软件工程 学号: 3170105860

同组学生姓名: 无 指导老师: 胡晓军

实验地点: 疫情期间于家中 实验日期: 2020 年 3 月 1 日-5 月 23 日

1. 实验目的

任选一种技术实现一个调查问卷的网站

2. 实验要求

需要实现的基本功能如下:

1. 实现用户注册、登录功能, 用户注册时需要填写必要的信息并验证, 如用户名、密码要求在 6 字节以上, email 的格式验证, 并保证用户名和 email 在系统中唯一。
2. 用户登录后可以设计问卷, 一个问卷由标题、问卷说明和多个问卷项目组成, 提供一个界面来动态设计, 问卷项目需要最少需要实现以下内容
 - 单选
 - 多选
 - 文本填写 (支持单行、多行)
 - 数字填写 (支持定义数字类型: 整数、小数)
 - 评分 (这是单选的特殊形式, 界面表示有所不同)

请您对此项目进行评价

☒ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

非常不满意

非常满意

- 级联选择 (对于单选的每个选择, 可以动态显示其他一个或多个问卷项目)

1 是否在校

☐ 是

☐ 否

如用户选“是”，就显示一些项目，选“否”就显示另外一些项目

3. 问卷设计完成后生成填写链接，通过分享链接由他人填写，分享时可以设定填写周期和填写方式，填写方式支持以下几种

- 仅限注册用户
- 无需注册，可填写 n 次
- 无需注册，每天可填写 n 次

自行设计如何确定同一人。

4. 保存填写结果，设计一个界面展现填写结果，要求展示界面直观清晰，有一定的统计量（如填报人数、起止时间等），对于数字类型的问卷项目支持汇总计算。

增强功能：

5. 样式适配手机端，能够在手机浏览器/微信等应用内置的浏览器中友好显示，支持额外的问卷项目

- 地理位置

一些可能需要的账号与密码：

阿里云服务器：47.94.46.115

账号：root

密码：5860@zju

MySQL 数据库：47.94.46.115:3306

用户名：root

密码：123456

Redis 数据库：47.94.46.115:2020

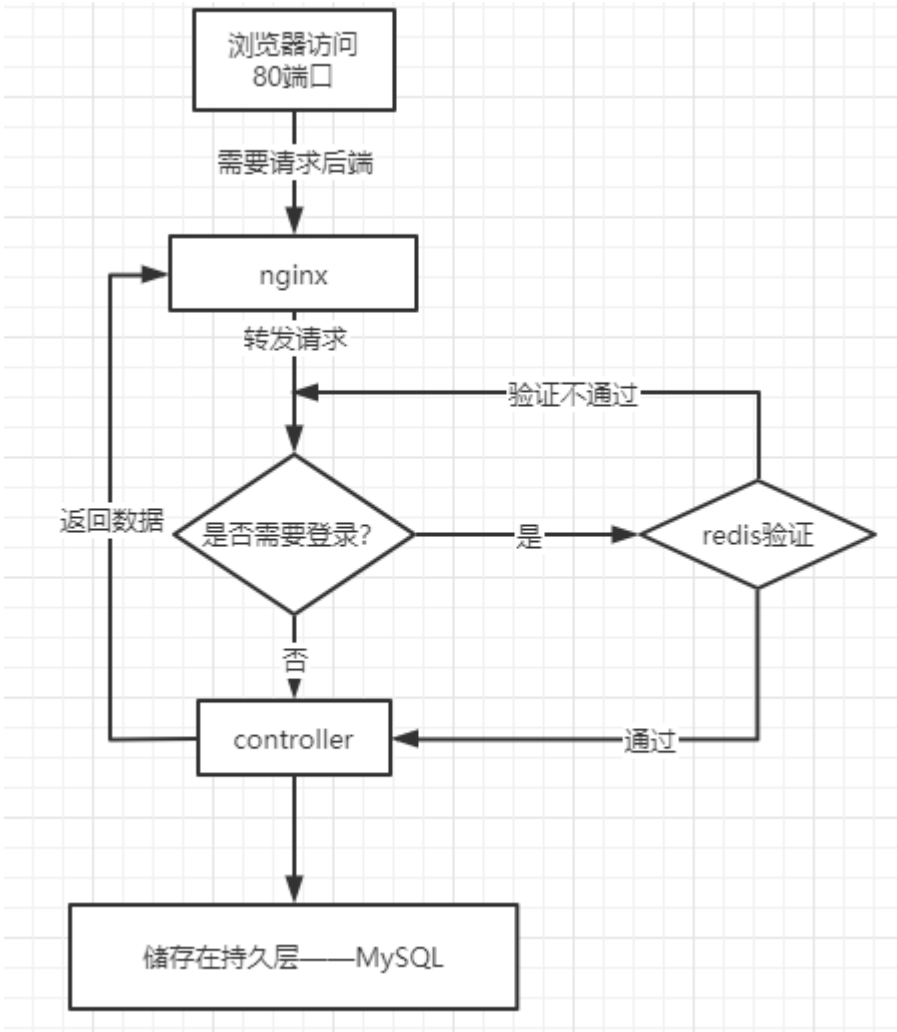
数据库 name：redis

3. 实验过程

整体架构思路：

前后端分离。前端采用 element-ui 以及，vue.js 并通过 nginx 转发请求到后端端口，后端采用 spring boot + mybatis 框架。Redis、Mysql、nginx 以及后端服务都通过 docker 部署在服务器 47.94.46.115 的不同端口上。

整体流程如下图：



3.1 购买并搭建服务器

疫情期间阿里云学生服务器免费领取：

实例ID/名称	标签	监控	可用区	IP地址	状态	网络类型	配置	付费方式	操作
i-2zeir6up2906ad31pzh3			华北 2 可用区 H	47.94.46.115(公)	运行中	专有网络	2 vCPU 4 GiB (I/O优化) ecs.t5-c1m2.large 1Mbps	包年包月 2020年11月4日 23:59 到期	管理 远程连接 升降配 续费 更多
iZ2zeir6up2906ad31pzh3Z				172.17.49.45(私有)					

配置信息	升降配	更多▼
CPU: 2核		
内存: 4 GiB		
实例类型: I/O优化		
操作系统: CentOS 7.7 64位		
弹性网卡: eni-2zefxpd65yi9x0p17j99 		
公网IP: 47.94.46.115 		
弹性公网IP: - 		
私有IP: 172.17.49.45 		
辅助私网IP: 管理辅助私网IP		
带宽计费方式: 按固定带宽		
当前使用带宽: 1Mbps		

3.2 Docker 的安装

卸载旧版本:

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

安装新版本, 由于阿里云自动启用阿里的仓库因此不用配置 ubuntu 的仓库:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

检查安装完毕

```
[root@iZ2zeir6up2906ad3lpzh3Z ~]# docker --version
Docker version 19.03.8, build afacb8b
```

由于镜像源位于国外, 安装其他软件较慢, 配置 docker 镜像加速, 找到阿里云的私人镜像加速:

镜像加速器

加速器

使用加速器可以提升获取Docker官方镜像的速度

加速器地址

<https://xi3mk5ux.mirror.aliyuncs.com>

复制

1. 安装 / 升级 Docker 客户端

推荐安装 1.10.0 以上版本的 Docker 客户端，参考文档 [docker-ce](#)

2. 配置镜像加速器

针对 Docker 客户端版本大于 1.10.0 的用户

您可以通过修改 daemon 配置文件 `/etc/docker/daemon.json` 来使用加速器

```
sudo mkdir -p /etc/docker
```

```
sudo tee /etc/docker/daemon.json <<-'EOF'
```

```
{  
  "registry-mirrors": ["https://xi3mk5ux.mirror.aliyuncs.com"]  
}
```

```
EOF
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart docker
```

3.3 MySQL 的安装部署

拉取镜像

```
docker pull mysql:latest
```

运行容器

```
docker run -itd --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=123456 mysql
```

3.4 Redis 的安装部署

拉取 image

```
docker pull redis:latest
```

运行容器

```
docker run -itd --name redis -p 2020:6379 redis:latest myredis
```

3.5 docker compose 的安装与使用

Compose 是用于定义和运行多容器 Docker 应用程序的工具。通过 Compose，您可以使用 YAML 文件来配置应用程序需要的所有服务。然后，使用一个命令，就可以从 YAML 文件配置中创建并启动所有服务。本次实验中只是简单的使用了 docker compose

Compose 的安装：

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
$ sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

前端的 docker-compose.yml 文件

(与 nginx.conf 和 vue 的 npm run build 生成的 dist 都同级放在同一文件夹中)

```
version: '3'
services:
  # 服务名称
  nginx:
    # 镜像:版本
    image: nginx:latest
    container_name: frontend
    # 映射容器 80 端口到本地 80 端口
    ports:
      - "80:80"
    # 数据卷 映射本地文件到容器
    volumes:
      # 映射 nginx.conf 文件到容器的/etc/nginx/conf.d 目录并覆盖 default.conf 文件
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
      # 映射 dist 文件夹到容器的/usr/share/nginx/html 文件夹
      - ./dist:/usr/share/nginx/html
    # 覆盖容器启动后默认执行的命令。
    command: /bin/bash -c "nginx -g 'daemon off;'"
```

后端的 docker-compose.yml 文件

```
version: '3'
services:
  # 服务名称
  java:
    # 镜像:版本
    image: openjdk:latest
    container_name: backend
    # 映射容器 8080 端口到本地 8080 端口
    ports:
```

```
- "8080:8080"
volumes:
  - ./usr/share
# 覆盖容器启动后默认执行的命令。
command: /bin/bash -c "java -jar /usr/share/questionnaire-0.1.jar"
```

3.6 Nginx 与前端的安装部署

首先和刚刚相同都进行拉取

```
$ docker pull nginx:latest
```

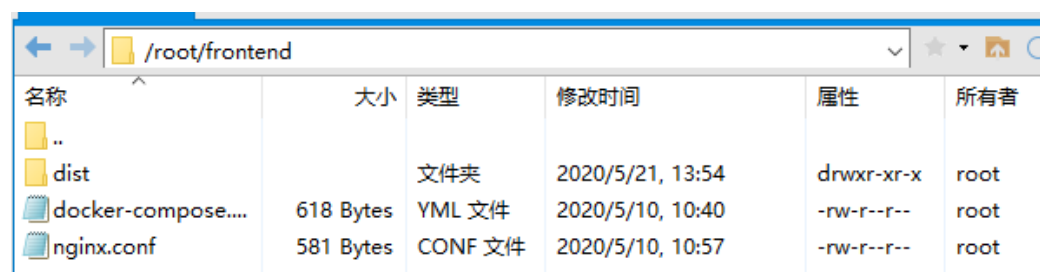
然后到下面图中的 frontend 文件夹中

```
$ docker-compose up -d
```

```
[root@iZ2zeir6up2906ad3lpzh3Z frontend]# docker-compose up -d
Starting frontend ... done
```



名称	大小	类型	修改时间	属性	所有者
..					
backend		文件夹	2020/5/5, 11:16	drwxr-xr-x	root
frontend		文件夹	2020/5/23, 20:25	drwxr-xr-x	root



名称	大小	类型	修改时间	属性	所有者
..					
dist		文件夹	2020/5/21, 13:54	drwxr-xr-x	root
docker-compose....	618 Bytes	YML 文件	2020/5/10, 10:40	-rw-r--r--	root
nginx.conf	581 Bytes	CONF 文件	2020/5/10, 10:57	-rw-r--r--	root

3.7 后端的部署

与前端同理，先要拉取镜像

```
$ docker pull openjdk
```

然后到下面的 backend 文件夹中

```
$ docker-compose up -d
```


/root/backend					
名称	大小	类型	修改时间	属性	所有者
..					
docker-compose....	373 Bytes	YML 文件	2020/5/8, 13:24	-rw-r--r--	root
questionnaire-0.1.j...	35.85MB	Executabl...	2020/5/23, 1:57	-rw-r--r--	root

通过 docker 安装的一些镜像如下:

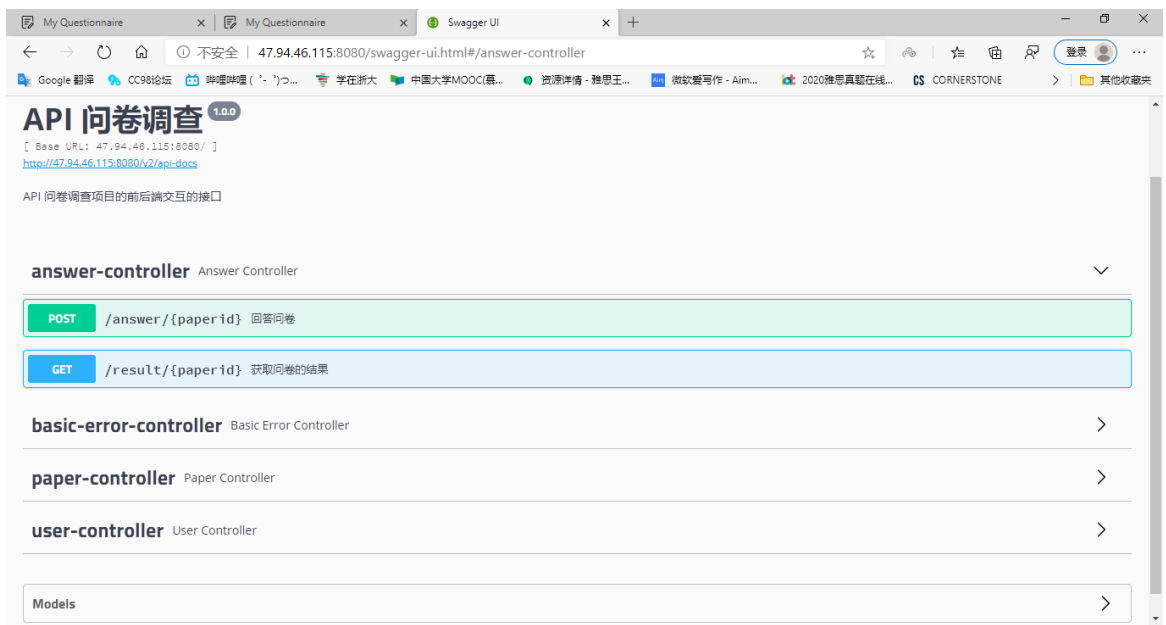
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
openjdk	latest	db8922a53903	2 weeks ago	497MB
redis	latest	f9b990972689	3 weeks ago	104MB
mysql	latest	a7a67c95e831	3 weeks ago	541MB
nginx	latest	602e111c06b6	4 weeks ago	127MB
eolinker/eolinker-api-management-system	latest	e89dadd3a32c	23 months ago	1.87GB

最后, 通过 docker 部署后的一些容器结果如下:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
815adea63f18	redis	"docker-entrypoint.s..."	13 days ago	Up 13 days	0.0.0.0:2020->6379/tcp	redis
33eca908e507	nginx:latest	"/bin/bash -c 'nginx..."	2 weeks ago	Up 19 hours	0.0.0.0:80->80/tcp	frontend
0be53b410697	openjdk:latest	"/bin/bash -c 'java ..."	2 weeks ago	Up 18 hours	0.0.0.0:8080->8080/tcp	backend
126ad592dd75	mysql	"docker-entrypoint.s..."	2 weeks ago	Up 2 weeks	0.0.0.0:3306->3306/tcp, 33060/tcp	mysql

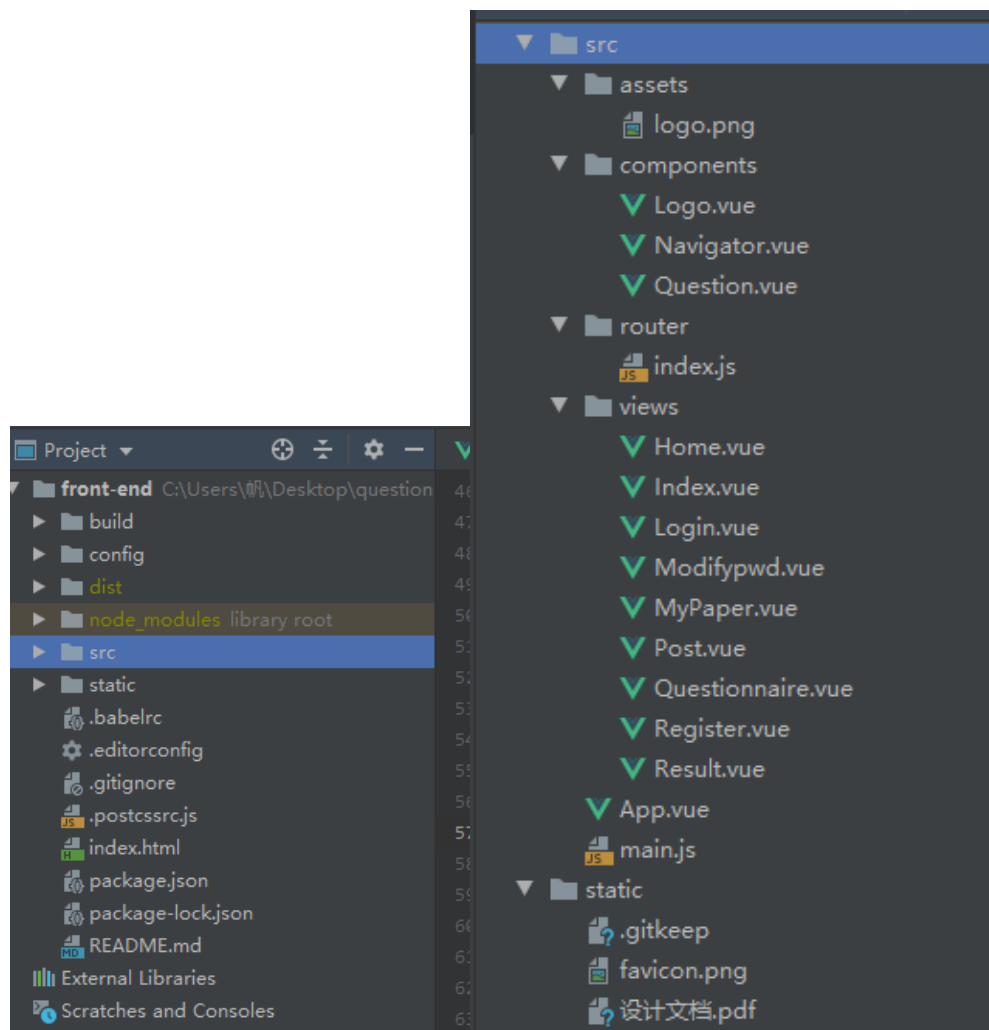
3.8 接口的设定

通过 spring 的 swagger 插件, 定义在网站: <http://47.94.46.115:8080/swagger-ui.html>



3.9 前端代码的开发

前端采用 vue.js 框架结合 element-ui 进行开发，具体代码在文件夹中：



路由：

```
import Vue from 'vue'
import Router from 'vue-router'
import Index from '@/views/Index'
import Login from '@/views/Login'
import Register from '@/views/Register'
import Questionnaire from '@/views/Questionnaire'
import Home from '@/views/Home'
import Modifypwd from "@/views/Modifypwd";
import MyPaper from "@/views/MyPaper";
import Post from "@/views/Post";
import Result from "@/views/Result";

Vue.use(Router);
```

```
export default new Router({
  routes: [
    {
      path: '/',
      name: 'Index',
      component: Index
    },
    {
      path: '/login',
      name: 'Login',
      component: Login
    },
    {
      path: '/register',
      name: 'Register',
      component: Register
    },
    {
      path: '/home',
      name: 'Home',
      component: Home
    },
    {
      path: '/questionnaire',
      name: 'Questionnaire',
      component: Questionnaire
    },
    {
      path: '/modifypwd',
      name: 'Modifypwd',
      component: Modifypwd
    },
    {
      path: '/post',
      name: 'Post',
      component: Post
    },
    {
      path: '/mypaper',
      name: 'MyPaper',
      component: MyPaper
    },
    {
      path: '/result',
```

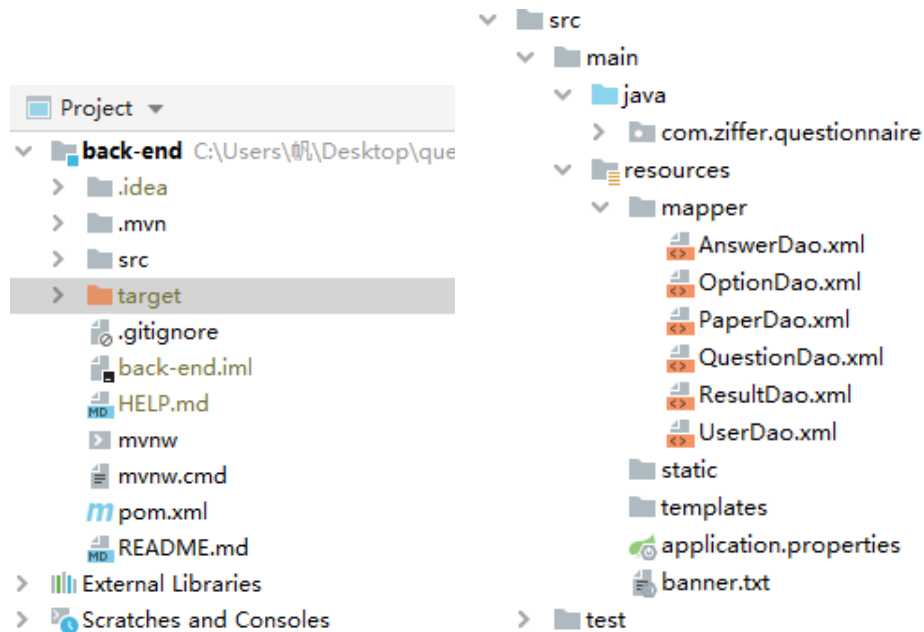
```

name: 'Result',
component: Result
}
]
})

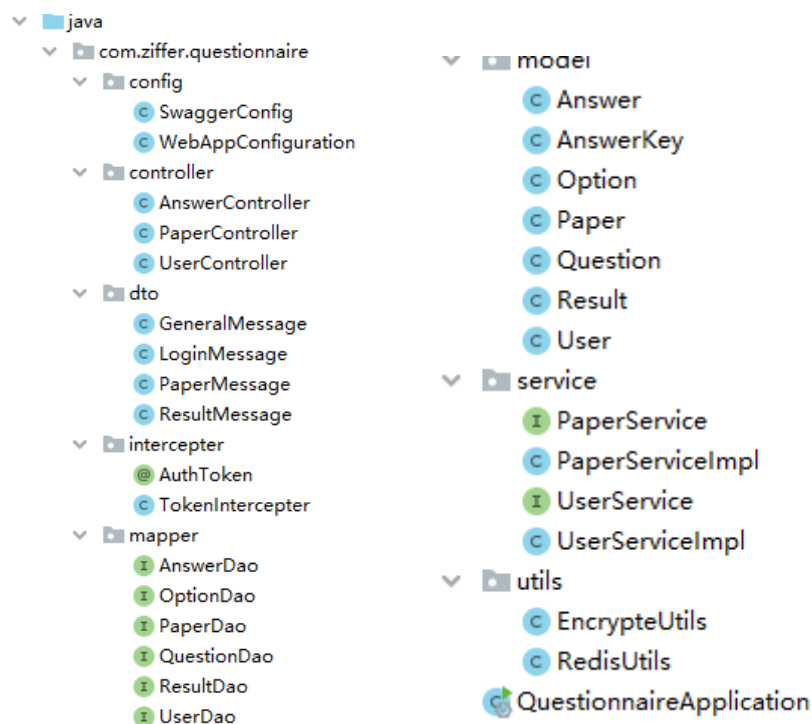
```

3.10 后端代码的开发

后端采用 Spring Boot+MyBatis 框架并使用了一些插件如 lombok 等进行开发，文件目录如下：



Java 代码部分：



下面展示部分代码

后端 springboot 主要配置 (application.properties):

#应用名

spring.application.name=questionnaire

#端口

server.port=8080

#编码格式

server.tomcat.uri-encoding=utf-8

#数据库配置

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://47.94.46.115:3306/questionnaire

spring.datasource.username=root

spring.datasource.password=123456

spring.datasource.type = com.alibaba.druid.pool.DruidDataSource

##指定使用redis 数据库索引(默认为0)

spring.redis.database=0

spring.redis.host=47.94.46.115

spring.redis.port=2020

###发送邮箱验证码

#spring.email.host=smtp.qq.com

#spring.email.username=937427981@qq.com

#spring.email.password=vkyppkpwvewbcaa

spring.authorize.salt=@(!(\$ (317ziffer010djlsf5860

#session 生命周期

server.servlet.session.timeout=30m

mybatis.mapper-locations=classpath:mapper/*.xml

登录注册部分主要代码 (UserController):

package com.ziffer.questionnaire.controller;

@RestController

@RequestMapping("/user")

public class UserController {

@Resource

UserServiceImpl **userServiceImpl**;

```

@Autowired
RedisUtils redisUtils;
@Autowired
private EncryptUtils encryptUtils;

// 等价@PostMapping("/login")
@ApiOperation("登录")
@RequestMapping(value = "/login",method =
RequestMethod.POST)
public LoginMessage login(@RequestParam("username") String
username,
                        @RequestParam("password") String
password){
    LoginMessage message = new LoginMessage();
    if(username=="||password==""){
        message.setState(false);
        message.setMessage("密码或用户名不能为空");
    } else if(username.length()<6){
        message.setState(false);
        message.setMessage("用户名至少 6 位");
    } else if(password.length()<6){
        message.setState(false);
        message.setMessage("密码至少 6 位");
    } else {
        User user = userServiceImpl.getByUsername(username);
        if(user==null){
            message.setState(false);
            message.setMessage("用户名或密码错误");
        }else if(!user.getPassword().equals(password)){
            message.setState(false);
            message.setMessage("用户名或密码错误");
        }else{
            String token =
encryptUtils.getMD5Code(username,password);
            redisUtils.set(token,username);
            message.setUsername(username);
            message.setState(true);
            message.setMessage(token);
        }
    }
    return message;
}

@ApiOperation("注册")

```

```

    @RequestMapping(value = "/register",method =
RequestMethod.POST)
    public LoginMessage register(@RequestParam("username")
String username,
                                @RequestParam("password") String
password,
                                @RequestParam("email") String
email){
        LoginMessage message = new LoginMessage();
        if(username=="||password==""){
            message.setState(false);
            message.setMessage("密码或用户名不能为空");
        }else if(username.length()<6){
            message.setState(false);
            message.setMessage("用户名至少 6 位");
        } else if(password.length()<6){
            message.setState(false);
            message.setMessage("密码至少 6 位");
        }else {
            User userSelect =
userServiceImpl.getByUsername(username);
            if(userSelect!=null){
                message.setState(false);
                message.setMessage("用户名已存在");
            }else{
                User user = new User();
                user.setEmail(email);
                user.setPassword(password);
                user.setUsername(username);
                //notice that 插入需要主键, 自动生成的mybatis 不包
                含主键
                if(userServiceImpl.register(user)>0){
                    String token =
encryptUtils.getMD5Code(username,password);
                    redisUtils.set(token,username);
                    message.setUsername(username);
                    message.setState(true);
                    message.setMessage(token);
                }else{
                    message.setState(false);
                    message.setMessage("邮箱已存在");
                }
            }
        }
    }
}

```

```

        return message;
    }

    @ApiOperation("修改密码")
    @RequestMapping(value = "/modifypwd", method =
RequestMethod.PUT)
    @AuthToken
    public GeneralMessage modifypwd(@RequestParam("username")
String username,
                                   @RequestParam("password")
String password){
        GeneralMessage message = new GeneralMessage();
        User user = userServiceImpl.getByUsername(username);
        if(password.length()<6){
            message.setMessage("密码长度至少 6 位");
            message.setState(false);
        } else
        if(user!=null&&!user.getPassword().equals(password)){
            user.setPassword(password);
            userServiceImpl.updatePassword(user);
            message.setState(true);
            message.setMessage("修改成功");
        }else{
            message.setMessage("不能和原密码相同");
            message.setState(false);
        }
        return message;
    }
}

```

拦截器部分主要代码（TokenInterceptor）,主要用于部分需要登录的接口加入注解:

```
package com.ziffer.questionnaire.interceptor;
```

```

@Service
public class TokenInterceptor implements HandlerInterceptor {
    @Resource
    private RedisUtils redisUtils;
    private final String httpHeaderName = "token";
    private final String httpParamUsername = "username";
    //    //鉴权失败后返回的错误信息，默认为401 unauthorized
    //    private String unauthorizedErrorMessage = "401
unauthorized";
    //    //鉴权失败后返回的HTTP 错误码，默认为401
    //    private int unauthorizedErrorCode =

```



```

HttpServletResponse.SC_UNAUTHORIZED;
//    //存放登录用户模型 Key 的 Request Key
//    public static final String REQUEST_CURRENT_KEY =
"REQUEST_CURRENT_KEY";

/**
 * @Description 在业务处理器处理请求之前被调用。预处理，可以进行编码、安全控制等处理;
 * @Data 2020-05-11
 * @Version 1.0
 */
@Override
public boolean preHandle(HttpServletRequest request,
HttpServletResponse response, Object handler) throws Exception
{
    if (!(handler instanceof HandlerMethod)) {
        return true;
    }
    HandlerMethod handlerMethod = (HandlerMethod) handler;
    Method method = handlerMethod.getMethod();

    // 如果打上了 AuthToken 注解则需要验证 token
    if (method.getAnnotation(AuthToken.class) != null ||
handlerMethod.getBeanType().getAnnotation(AuthToken.class) !=
null) {
        String token = request.getHeader(httpHeaderName);
        String transUsername =
request.getParameter(httpParamUsername);
        String username;
        PrintWriter writer;
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=utf-8");
        if (token != null && token.length() != 0) {
            username = redisUtils.get(token);
            if(username==null){
                String error = "token 信息有误";
                writer = response.getWriter();
                writer.print(error);
                return false;
            }else if(!username.equals(transUsername)){
                String error = "token 信息有误";
                writer = response.getWriter();
                writer.print(error);
                return false;
            }
        }
    }
}

```

```

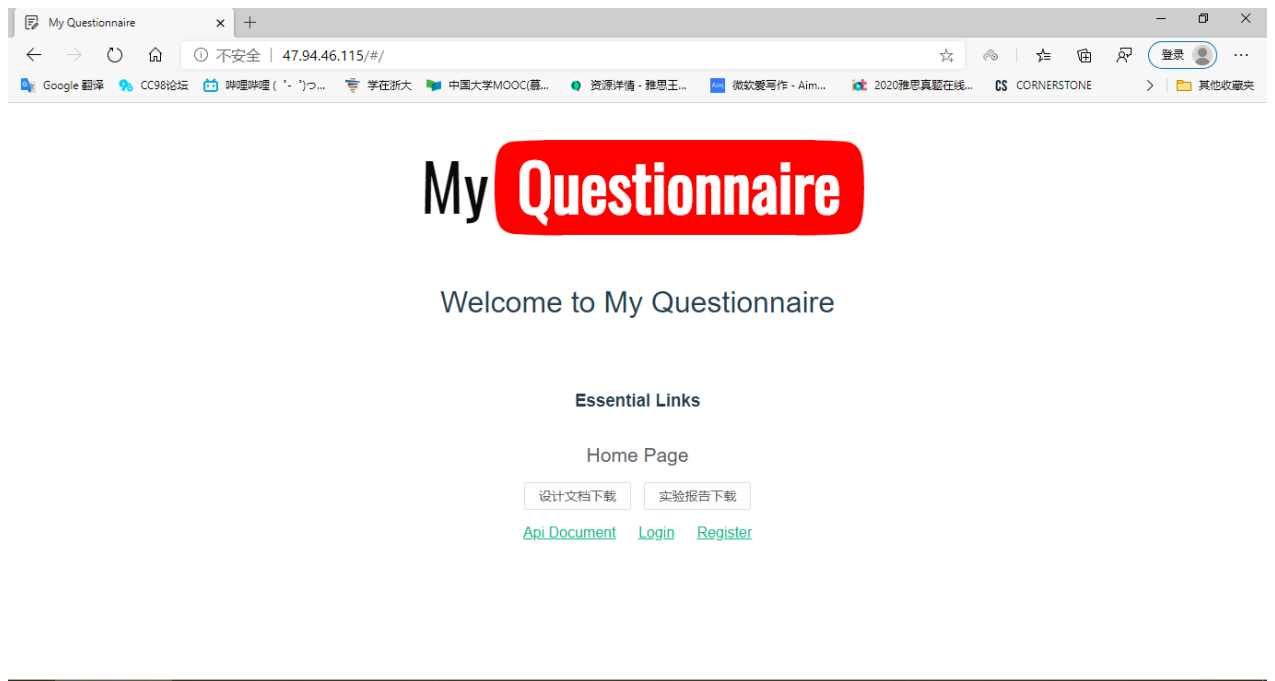
    }
} else {
    String error = "token 信息有误";
    writer = response.getWriter();
    writer.print(error);
    return false;
}
}
return true;
}
}
}

```

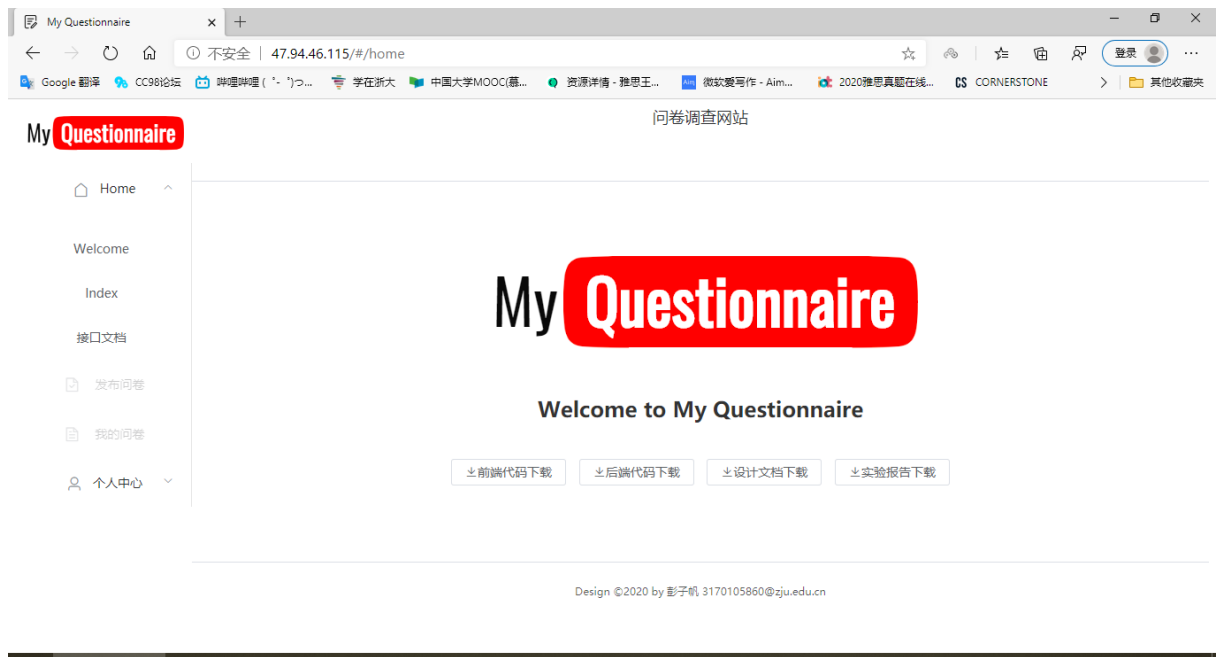
4 实验结果

最后成果的网站如下: <http://47.94.46.115/#/>

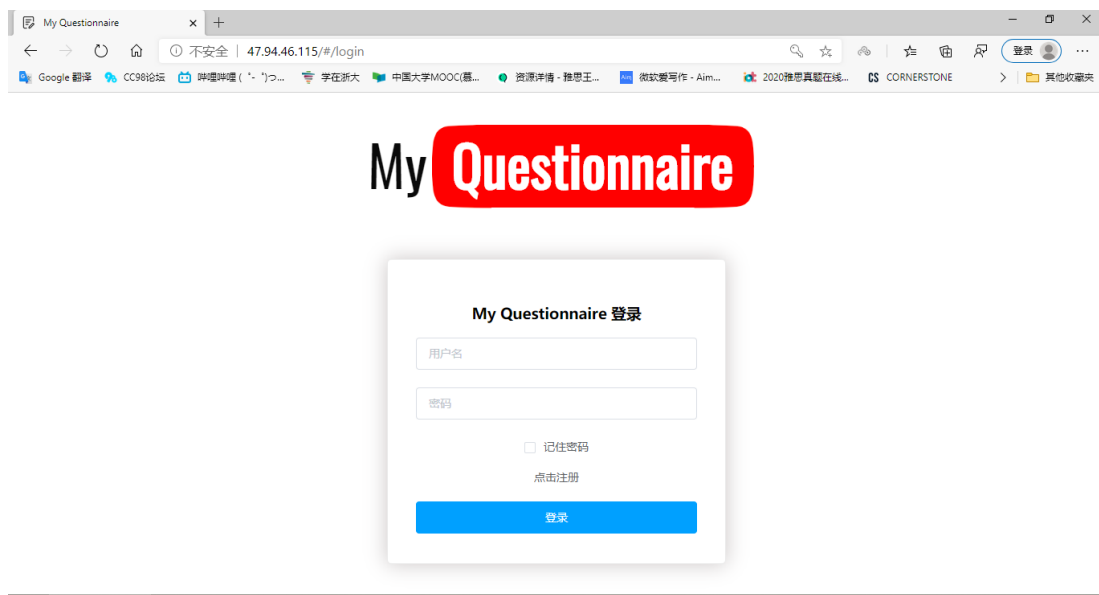
Index 页:



HomePage:



登录、注册与修改密码类似：



发布问卷：

My Questionnaire

My Questionnaire

Swagger UI

47.94.46.115/#/post

Google 翻译

CC98论坛

哔哩哔哩 (゜-゜)つ_

学在浙大

中国大学MOOC(慕)

资源详情 - 雅思王...

微软爱写作 - Aim...

2020雅思真题在线...

CORNERSTONE

其他收藏夹

My Questionnaire

问卷调查网站

Home

Welcome

Index

接口文档

发布问卷

我的问卷

个人中心

* 问卷名称

test

问卷描述

test

问卷类型

☐ 仅允许登录作答
 ☒ 未登录可作答N次
 ☐ 未登录每日可作答N次

- 3 +

问卷发布后即可开始作答

☒

问题1

题目内容

题目类型

☒ 单选
 ☐ 多选
 ☐ 文本问答
 ☐ 数字 (整数)

问题1

题目内容 单选

题目类型

☒ 单选
 ☐ 多选
 ☐ 文本问答
 ☐ 数字 (整数)

选项1 单选项项1

选项2 单选项项2

选项3 单选项项3

+ 添加选项

我的所有问卷:

My Questionnaire

问卷调查网站

Home

Welcome

Index

接口文档

发布问卷

我的问卷

个人中心

修改密码

回答页面:

5 开发体会

经过了不短的时间的学习，尤其是对 B/S 体系架构的学习，学习到了很多知识。又经过不断边学习以及边开发，在实践中获得了不少难以从理论上获取的知识，包含但不局限于一下知识：

1. 独立设计数据库
2. Github 版本控制与代码管理
3. Docker
4. Nginx
5. Redis
6. MySQL
7. Spring Boot
8. Vue.js
9. Docker compose 的使用
10. CI/CD

本次开发算是真正自己、从头到尾地开发一个网站，并且遇到许多问题需要不断地查询解决方案，从中不仅仅锻炼了自己信息检索能力，还增长了数据库设计、运营维护、版本控制、代码管理、部署等等知识，最重要的是解决问题的能力。

除了技术上的提高，还有对 B/S 体系结构的项目的理解，为以后学习、工作等等打下了深厚的基础。本次课程对每周迭代完成功能的要求是十分有意义的，模拟更真实，更高效的项目开发过程。之前的课程往往是以功能为单元，小组完成大作业，容易出现时间管理不当，从而影响工作效率和结果的问题。

最后，感谢老师批阅我的实验项目~