开发文档

这份文档将记录我都做了些什么…

目录

总体目录（概览）

1. 前端

前端采用 vue-cli 3.0 脚手架开发。

技术栈:	
http 拦截	axios
身份验证	JSON Web Token (JWT)
状态管理	Vuex
UI 界面	Element-UI

目前功能。

地址	描述
/register	注册页面
/login	登录页面
/index	主页
/home	欢迎页，由右侧导航访问
/InfoShow	显示个人信息
/expaperlist	试卷列表
/mypaper	创建答题卡
/expapercreate	创建试卷

2. 后端

后端使用 node.js 开发。

技术栈:	
框架	Express 框架
身份验证	Passport、passport-jwt、JSON Web Token (JWT)、bcryptjs
url 中间件	body-parser
UI 界面	Element-UI
数据持久化	Mysql2、sequelize

接口

已提供 API 列表	Method	
/api/getscript	GET	获取试卷列表
/api/del_expaper	GET	删除列表项
/api/add_expaper（post）	POST	添加答题卡
/api/add_expaper（get）	GET	添加答题卡
/api/user/register	POST	注册
/api/user/login	POST	登录，拿到 token

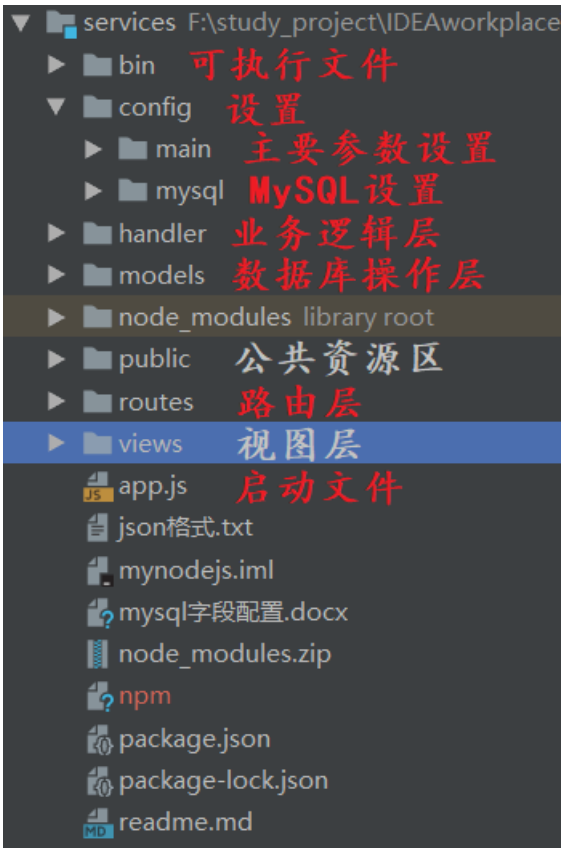
3. 数据库

设计数据库字段。
设计 expaper、user 表

功能详情

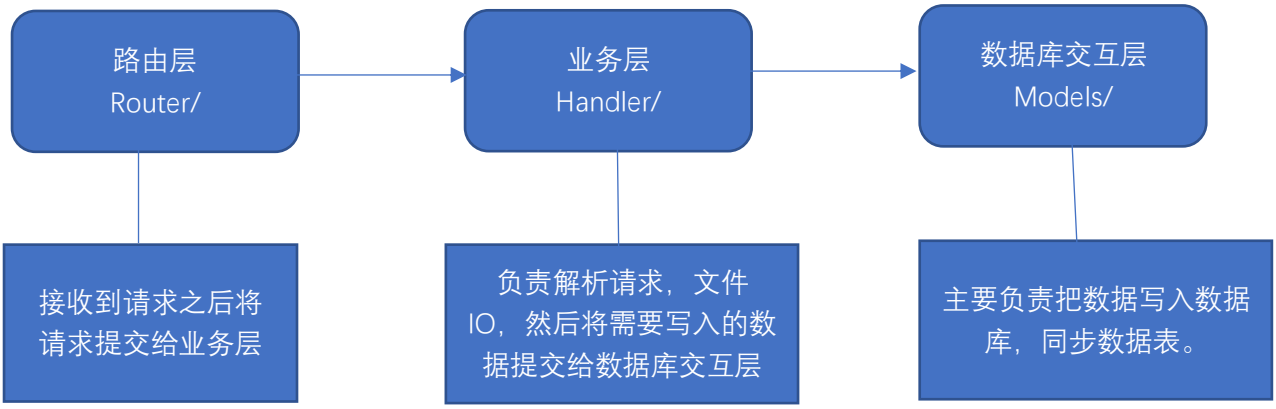
后端

目录结构说明：



接口详情：

整体处理流程为：



添加试卷接口（典例）

POST /api/add_expaper（post）

Router/expaper.js

```
/** post 添加试卷接口**/
router.post('/api/add_expaper', function (req, res, next) {
    handler.addExpaperApi(req, res);
});
```

Handler/expaperhandler.js

使用了 Promise 规范。避免地狱回调

```
/** 纯API 不去定向 与前端解耦**/
module.exports.addExpaperApi = function (req, res, callback) {
    setCORSallow(req, res);
    /** 先在磁盘写json文件 然后把信息写入数据库 如果出错则回滚操作删除文件 **/
    writeNewDate(context.json2str(req))
        .then( onfulfilled: (content) => {
            return sqlhandler.addSqlP(req, res, context.sqlobj(req, content))
        })
        .then( onfulfilled: (data) => {
                res.json(data)
            },
            onrejected: (err) => {
                fs.unlink(data);
                res.json(err+"数据写入错误，已经写入的文件已删除！");
            })
        .catch( onrejected: (err) => {
            res.json(err)
        });
    // writeNewsDate(context.json2str(req), function (con...

};
```

promise，避免地狱回调

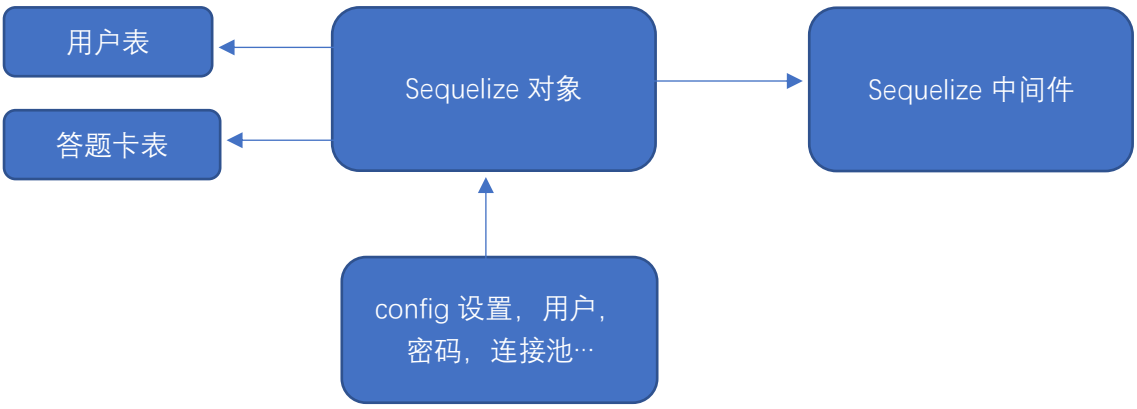
```
/**  防止文件重写、覆盖，所以加入了循环判断文件是否存在，安全系数用来避免无限循环**/
/** 这个操作用来确定文件以后可以被应用于 HDFS的分布式文件系统**/
function writeNewDate(data) {
    return new Promise( executor: (resolve, reject) => {
        var dataPath = path.join(config.dataPath, "data1");  //获取data路径
        var count = fs.readdirSync(dataPath).length + 1;
        dataPath = path.join(dataPath, "expaper" + count + ".json");
        var c = 0; //安全系数
        while (fs.existsSync(dataPath) && c != 100) {
            count++;
            dataPath = path.join(config.dataPath, "data1", "expaper" + count + ".json");
            console.log(count);
            c++;
        }
        fs.writeFile(dataPath, data, function (err) {
            if (err) {
                reject(err)
            } else {
                resolve(dataPath)
            }
        });
    });
}
```

\models\db\sqlhandler.js
依然使用 promise

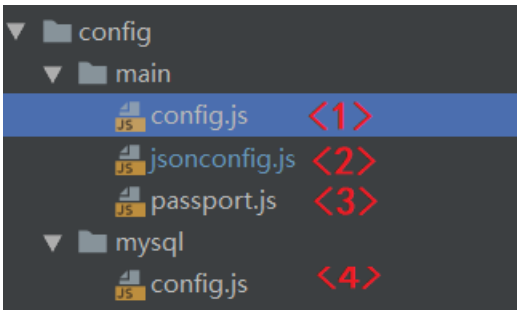
```
/** 判断请求来的数据是否可以被写入数据库，成功则返回写入的数据，失败返回错误**/
module.exports.addSqlP = function (req, res, message) {
    return new Promise( executor: (resolve, reject) => {
        //如果没有post数据或者数据为空,直接返回
        if (message.name == undefined || message.name == '') {
            var result = {
                status: 500,
                tips: "服务端接受了空数据!拒绝访问!"
            };
            reject(result);
            return;
        }
        //创建一条记录,创建成功后跳转回首页
        Expaper.create(message).then(function (msg) {
            var result = {
                status: 200,
                tips: "请求正常!",
                data: msg
            };
            resolve(result)
        });
    })
};
```

其他接口也类似。

数据持久化(mysql、Sequelize 对象):



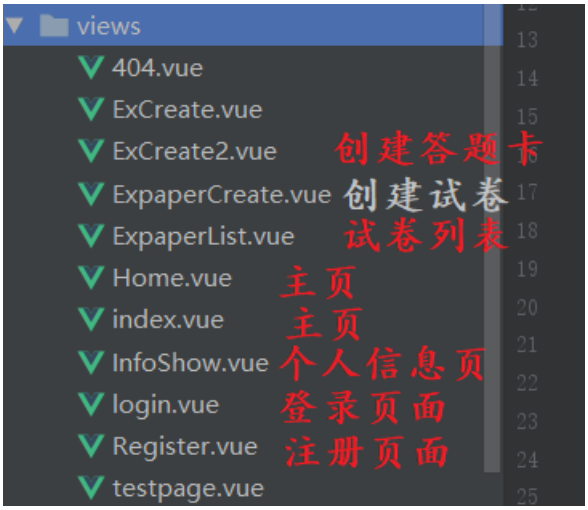
Models/db/db.js	*包装 Sequelize 中间件*
Models/db/tables	*声明表，同步表。*
Config/mysql	*设置 mysql 用户密码连接池*



<1> 存放设置 app 端口，数据存放路径等 app 相关设置。
<2>存放即将被写入的 json 数据和 sql 数据对象的格式。
<3>存放 jwt 相关设置
<4>mysql 用户密码设置。

前端

目录说明

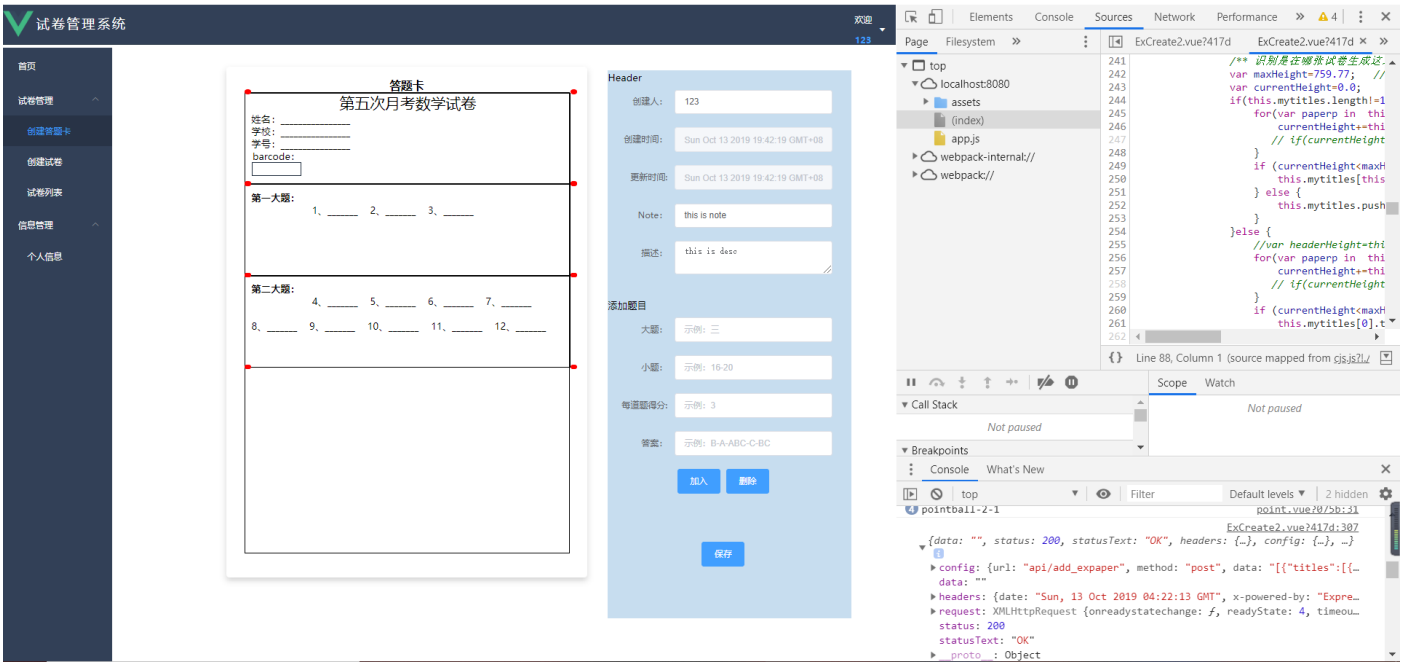


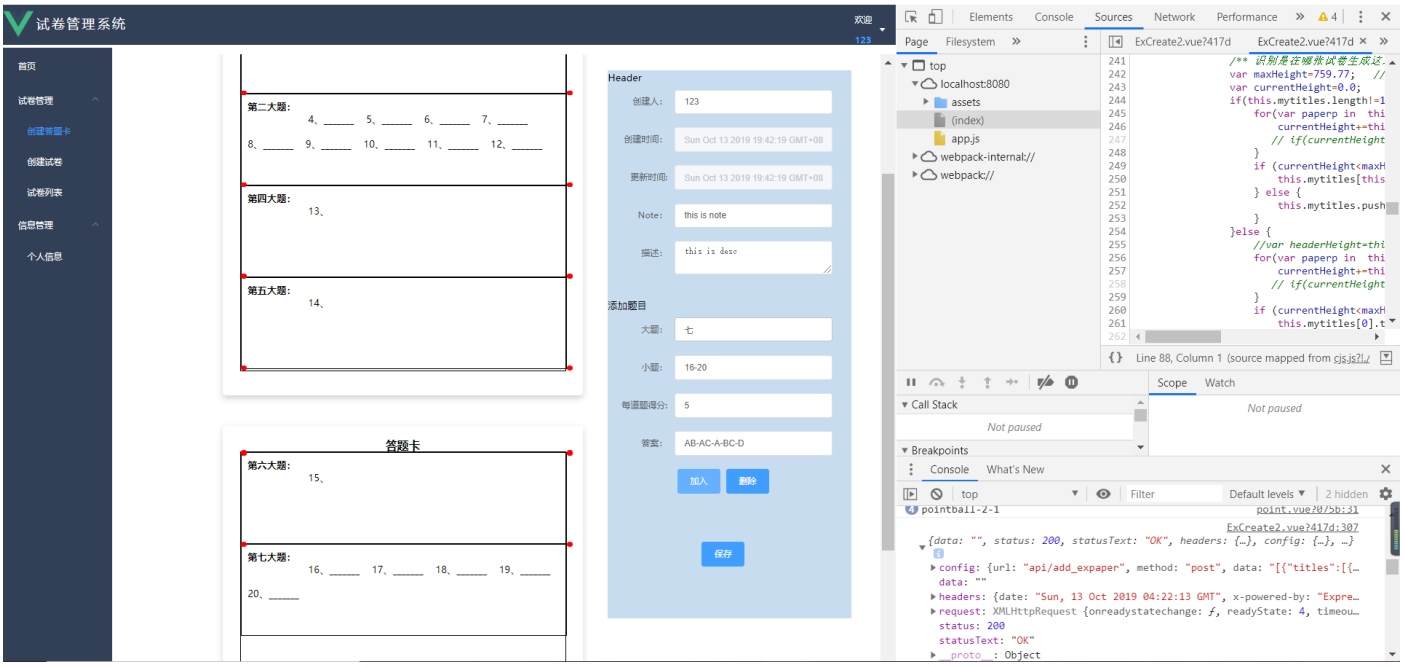
页面详情及实现：

/mypaper　　创建答题卡页面

源码路径：/views/excreate2.vue

视图如下:





原理：将答题卡全部信息以 json 数组形式存放形式。添加试卷时向数组加入 json。

页面加载：

1.当页面加载时，mounted()函数将 token 中获取的用户信息加入答题卡 header,

2.开始循环数组加入 A4 试卷纸。

3.把数据交给子组件 questions.vue，在试卷纸上循环数组加入题目，并且返回题目坐标

4.通过坐标为每一道题目进行定位。

5.updated()函数判断加入试卷的正确性来进行相关操作。

“加入”操作（为答题卡添加题目）：

1. 创建题目 json 数组。
2. 将数组 append 到 data
3. 判断应该在哪张试卷的哪个位置生成题目 div

此 vue 实例方法：

```
    methods: {
        //用来测试是否能正常加入的函数。
        insertTitle: function (form) {...},
        //删除题目
        deletepop: function () {...},
        //判断是否成功
        isFail:function () {...},
        //设置题目位置
        setItemPosition: function (data) {...},
        //提交数据给后端接口
        submitForm(formName) {...},
    },
    updated() {
        this.isFail();
    },
    mounted() {
        this.mytitles[0].titles[0].header.teacher=this.user.username
    },
```

```
<template>
    <div class="background">
<!--        <rightmenu ref="rightmenu"></rightmenu>-->
<!--        迭代生成试卷纸 A4-->
        <template v-for="(paperpage,index) in mytitles">
            <div class="paper" >
                <h3 style="text-align: center">答题卡</h3>
                <div class="container" ref="container">

                    <!--迭代每一道大题，并且返回大题坐标-->
                    <template v-for="(item,tindex) in paperpage.titles">
                        <!--迭代每道小题-->
                        <questions :item=item :index={outer:index,inner:tindex} @listenData="setItemPosition">
                        </questions>
<!--                        返回坐标-->
                        <point v-for="i in 4" :myposition="item" :index="i" :tindex="index+'-'+ tindex" ></point>
                    </template>

                </div>
            </div>
        </template>
<!--        右侧添加题目导航栏-->
        <template...>


    </div>
</template>
```

试卷纸

题目和headerdiv框

定位点div

/login 登录页面

其他页面逻辑基本类似

使用说明

1. mysql：services/config/mysql/config.js 里面配置用户密码，然后导入 sql 脚本 storage/expaper.sql
2. 后端：在 services 下，npm install 然后启动或者配置 idea 环境 启动命令“node bin/www”
3. 前端：在 gui/expaper-client 下，npm install 后启动命令“npm serve”