课程内容目录（19/5）

1、搜索后端需求分析

A、知识点（1）

- 【熟悉】搜索原型中的功能需求

2、搜索后端定义

A、知识点（1）

- 【了解】项目实现的接口定义

3、搜索后端开发

A、知识点（8）

- 【熟悉】查询搜索记录接口的开发及Mock测试
- 【熟悉】删除搜索记录接口的开发及Mock测试
- 【熟悉】清空搜索记录接口的开发及Mock测试
- 【熟悉】查询今日热词接口的开发及Mock测试
- 【熟练】查询联想词接口的开发及Mock测试
- 【熟练】文章搜索接口的开发及Mock测试
- 【熟练】ES在项目中的集成和使用
- 【熟悉】项目ES搜索索引的设计技巧和API使用

B、拓展知识点（2）

- 【熟悉】spring.autoconfigure.exclude可以排除某项三方框架自动装载的功能

  > 在中大型项目中，源码工程管理，经常会遇见在一个通用模块进行配置管理，然后每个模块按需进行框架加载；此场景Spring Boot的自动装载功能就不显得有些累赘。但幸好的是spring给我们提供了对应的解决方案，关闭自动装载功能，然后在需要的地方手动装载即可。
  >
  > 这里的ES封装在common项目中，并在配置文件中配置不自动转载，然后提供自动转载的类ElasticsearchConfig。

- 【熟悉】JestClient使用示例

  > - 创建索引
  >
  > ```
  > //创建es索引
  > @Override public void createIndex() {
  >  try {
  >      jestClient.execute(new CreateIndex.
  > Builder(this.ES_INDEX_NAME).build());
  >  } catch (Exception e) {
  >      System.out.println(e.getMessage());
  >  }
  > }
  > ```
  >
  > - 添加文章信息

```
//向索引中添加文章信息
@Override
public void addArticle(ApArticle article) {
 Index index = new Index.Builder(article).index(this.ES_INDEX_NAME).
type(this.ES_TYPE_NAME).build();
    try { //获取jestClient实例并向索引插入文章信息
    JestResult jestResult = jestClient.execute(index);
     System.out.println(jestResult.isSucceeded());
 } catch (Exception e) {
     e.printStackTrace();
 }
}
```

- 查询文章信息

```
@Override
public ResponseResult esArticleSearch(UserSearchDto dto) {
 //第一页查询保存搜索记录
 if(dto.getFromIndex()==0){
     ResponseResult ret = getEntryId(dto);
     if(ret.getCode()!=AppHttpCodeEnum.SUCCESS.getCode()){
         return ret;
     }
     this.saveUserSearch((Integer) ret.getData(),
dto.getSearchWords());
 }
 //实例化SearchSourceBuilder对象
 SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
 searchSourceBuilder.query(QueryBuilders.matchQuery("title",
dto.getSearchWords()));
 searchSourceBuilder.size(dto.getPageSize());
 searchSourceBuilder.from(dto.getFromIndex());
 //利用SearchSourceBuilder对象构建index查询对象
 Search search = new Search.Builder(searchSourceBuilder.toString())
.addIndex(ESIndexConstants.ARTICLE_INDEX)
.addType(ESIndexConstants.DEFAULT_DOC).build();
 try {
     //获取jestClient实例并向对index执行查询语句
     JestResult result = jestClient.execute(search);
     String retStr = result.getJsonString();
     Map<String,Object> retMap = Maps.newHashMap();
     List<ApArticle> list =
result.getSourceAsObjectList(ApArticle.class);
     List<ApArticle> retList = Lists.newArrayList();
     for(ApArticle c : list){
         ApArticle article =
apArticleMapper.selectById(Long.valueOf(c.getId()));
         if(article==null)continue; retList.add(article);
     } //获取查询结果
     return ResponseResult.okResult(retList);
 } catch (Exception e) {
     e.printStackTrace();
 }
 return ResponseResult.errorResult(AppHttpCodeEnum.PARAM_INVALID);
}
```

4、前端详情开发

A、知识点（5）

- 【了解】搜索页和搜索结果页的需求
- 【了解】项目前端页面实现的功能范围和页面组件定义
- 【熟悉】通过迭代方式不断优化Request.js代码的过程
- 【熟悉】搜索页面的实现
- 【熟悉】搜索结果页面的实现

B、拓展知识点（2）

- 【熟悉】搜索页和结果页的组件开发

> 详见资料文件夹下《文章搜索页面组件开发说明.docx》文件

- 【熟练】前端页面的开发步骤

> 以下是建议的步骤，细心的同学可能发现这步骤和讲义不一样，是的，开发步骤对每个人都有所不同，而且同一个人随着工作年限增长，在每个时间段的开发习惯也有可能不同，所以开发步骤只是一种参考，以下的步骤对应初学者来说，可以方便页面效果调试，可供参考。
>
> - 需求分析
> - 页面拆分和组件定义
> - 创建主页面（内容可是测试测试）
> - Model定义
> - 配置路由
> - 组件开发
> - 组件测试（集成在主页面中测试）
> - 主页面VIEW开发
> - API开发
> - 主页面VM开发
> - 测试
> - 入口集成（在相应的页面，增加进入该页面的入口）

5、实名认证开发

A、知识点（4）

- 【熟练】创建后端微服务模块
- 【了解】实名认证的流程与需求，以及实现的范围
- 【熟练】实名认证的后端代码的开发
- 【熟悉】实名认证前端的代码开发

B、拓展知识点（1）

- 【熟悉】前端Admin项目的搭建

> 详见资料文件夹下《搭建前端Admin工程说明书.docx》文件

实名认证后端开发

9.1 需求分析

在系统中普通用户可以通过实名认证后可以成为自媒体用户， 在我们的后端管理系统中需要实现待审核用户列表、审核通过列表、审核失败列表、通过审核、驳回审核等功能，运营人员可通过此功能进行审核。

实名认证信息是通过APP进行申请提交的，信息主要包括有，身份证正反面、手持照、活体认证照片信息，这些信息提交后，由运营人员在后台管理系统中进行审核。注：此处实名认证仅实现后台系统的后端和前端开发。APP上的资料提交过程实则是照片上传的过程，就不进行演示。另外脸部识别的照片一般可通过三方活体认证的SDK进行获取。

9.2 接口定义

9.2.1 待审核列表

该接口直接从后端查询出待审核的数据即可， 查询时需要带上查询参数以及分页参数

参考标准	请参考通用接口规范
接口名称	/api/v1/admin/auth/authWaitList
请求DTO	com.heima.model.admin.dtos.AuthListDto
响应DTO	ResponseResult

CODE定义

PARAM_INVALID	PARAM_INVALID(501,"无效参数")

9.2.2 通过审核列表

参考标准	请参考通用接口规范
接口名称	/api/v1/admin/auth/authPassList
请求DTO	com.heima.model.admin.dtos.AuthListDto
响应DTO	ResponseResult

CODE定义

PARAM_INVALID	PARAM_INVALID(501,"无效参数"),

9.2.3 审核失败列表

参考标准	请参考通用接口规范
接口名称	/api/v1/admin/auth/authFailList
请求DTO	com.heima.model.admin.dtos.AuthListDto
响应DTO	ResponseResult

CODE定义

PARAM_INVALID	PARAM_INVALID(501,"无效参数"),

9.2.4 通过审核

参考标准	请参考通用接口规范
接口名称	/api/v1/admin/auth/authPass
请求参数	Integer id, String msg
响应DTO	ResponseResult

CODE定义

PARAM_INVALID	PARAM_INVALID(501,"无效参数"),

9.2.5 驳回审核

参考标准	请参考通用接口规范
接口名称	/api/v1/admin/auth/authFail
请求参数	Integer id , String msg
响应DTO	ResponseResult

CODE定义

PARAM_INVALID	PARAM_INVALID(501,"无效参数"),

9.3 后端管理模块开发

9.3.1 修改common模块

- 修改common模块resources/config下的application.properties, 添加一下配置

```
port.admin=9005
sn.admin=admin
```

- 修改common中修改MysqlConfig.java文件中的SqlSessionFactoryBean使其支持驼峰命名

```
@Bean
    public SqlSessionFactoryBean
mysqlCoreSqlSessionFactory(@Qualifier("mysqlCoreDataSource") DataSource
mysqlCoreDataSource) throws IOException {
        PathMatchingResourcePatternResolver resolver = new
PathMatchingResourcePatternResolver();
        SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean();
        sessionFactory.setDataSource(mysqlCoreDataSource);

 sessionFactory.setMapperLocations(resolver.getResources(this.getMapperFileP
ath()));
        sessionFactory.setTypeAliasesPackage(this.getAliasesPackage());
        //修改部分
        org.apache.ibatis.session.Configuration mybatisConf = new
org.apache.ibatis.session.Configuration();
        mybatisConf.setMapUnderscoreToCamelCase(true);
        sessionFactory.setConfiguration(mybatisConf);
        return sessionFactory;
    }
```

- 添加常量文件AdminConstans

```
package com.heima.admin.constans;

public class AdminConstans {
    public static final Short WAIT_AUTH = 1;
    public static final Short PASS_AUTH = 9;
    public static final Short FAIL_AUTH = 2;
}
```

9.3.2 添加接口到api模块

- 文件路径看包名

```
/**
 * 管理端认证管理控制器
 */
public interface AdminAuthControllerApi {

    /**
     * 获取待认证列表
     */
    ResponseResult loadAuthList(AuthListDto dto);

    /**
     * 获取审核通过列表
     * @param dto
     * @return
     */
    ResponseResult loadAuthPassList(@RequestBody AuthListDto dto);

    /**
     * 获取审核失败列表
     * @param dto
     * @return
     */
```

```
    ResponseResult loadAuthFailList(@RequestBody AuthListDto dto);

    /**
     * 审核通过
     * @param id
     * @param msg
     * @return
     */
    ResponseResult authPass(Integer id, String msg);


    /**
     * 审核失败
     * @param id
     * @param msg
     * @return
     */
    ResponseResult authFail(Integer id, String msg);
}
```

9.3.3 Admin模块编写

工程创建请参考article或者behavior模块，以下部分为特有部分

- pom文件

```
<dependencies>
    <!-- 引入依赖模块 -->
    <dependency>
        <groupId>com.heima</groupId>
        <artifactId>heima-leadnews-model</artifactId>
    </dependency>
    <dependency>
        <groupId>com.heima</groupId>
        <artifactId>heima-leadnews-common</artifactId>
    </dependency>
    <dependency>
        <groupId>com.heima</groupId>
        <artifactId>heima-leadnews-apis</artifactId>
    </dependency>
    <!-- Spring cloud starter -->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
    <!-- Spring boot starter -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <!-- 排除默认的logback日志，使用log4j-->
        <exclusions>
            <exclusion>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-logging</artifactId>
            </exclusion>
            <exclusion>
                <groupId>org.slf4j</groupId>
```

```
                <artifactId>slf4j-log4j12</artifactId>
            </exclusion>
            <exclusion>
                <groupId>ch.qos.logback</groupId>
                <artifactId>logback-access</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.dataformat</groupId>
        <artifactId>jackson-dataformat-cbor</artifactId>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.dataformat</groupId>
        <artifactId>jackson-dataformat-xml</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

- resources线面添加application.properties

```
server.port=${port.admin}
spring.application.name=${sn.admin}
```

- 控制器编写

```
@RestController
@RequestMapping("/api/v1/admin/auth")
public class AdminAuthController implements AdminAuthControllerApi {

    @Autowired
    private AdminAuthService adminAuthService;

    @GetMapping("/authWaitList")
    @Override
    public ResponseResult loadAuthList(@RequestBody AuthListDto dto) {
        return adminAuthService.load(AdminConstans.WAIT_AUTH, dto);
    }

    @GetMapping("/authPassList")
```

```
    @Override
    public ResponseResult loadAuthPassList(@RequestBody AuthListDto dto) {
        return adminAuthService.load(AdminConstans.PASS_AUTH, dto);
    }

    @GetMapping("/authFailList")
    @Override
    public ResponseResult loadAuthFailList(@RequestBody AuthListDto dto) {
        return adminAuthService.load(AdminConstans.FAIL_AUTH, dto);
    }

    @PostMapping("/authPass")
    @Override
    public ResponseResult authPass(Integer id, String msg) {
        return adminAuthService.updateStatusById(id, msg,
AdminConstans.PASS_AUTH);
    }

    @PostMapping("/authFail")
    @Override
    public ResponseResult authFail(Integer id, String msg) {
        return adminAuthService.updateStatusById(id, msg,
AdminConstans.FAIL_AUTH);
    }
}
```

- 服务层接口

```
public interface AdminAuthService {

    ResponseResult load(Short loadWaitAuthList, AuthListDto dto);

    ResponseResult updateStatusById(Integer id, String msg, Short passAuth);
}
```

- 服务层实现

```
@Service
public class AdminAuthServiceImpl implements AdminAuthService {

    @Autowired
    private ApUserRealnameMapper apUserRealnameMapper;

    @Autowired
    private WmUserMapper wmUserMapper;

    @Autowired
    private ApUserMapper apUserMapper;

    @Override
    public ResponseResult load(Short status, AuthListDto dto) {
        //参数为空
        if (dto == null) {
            return ResponseResult.errorResult(AppHttpCodeEnum.PARAM_INVALID);
        }
        //状态错误
```

```
        if (statusCheck(status)) {
            status = AdminConstans.WAIT_AUTH;
        }
        if (dto.getPage() == null || dto.getPage() < 0) {
            dto.setPage(1);
        }

        if (dto.getSize() == null || dto.getSize() < 0) {
            dto.setSize(10);
        }

        if (dto.getParams() == null) {
            dto.setParams(new HashMap<>());
        }
        Map<String, Object> params = dto.getParams();
        params.put("status", status);
        params.put("from", (dto.getPage() - 1) * dto.getSize());

        List<ApUserRealname> datas =
apUserRealnameMapper.loadListByStatusAndParam(dto);
        int total = apUserRealnameMapper.countAuthListByStatusAndParam(dto);
        PageResponseResult responseResult = new
PageResponseResult(dto.getPage(), dto.getSize(), total);
        responseResult.setData(datas);
        return responseResult;
    }

    @Override
    public ResponseResult updateStatusById(Integer id, String msg, Short status)
{
        if (id == null) {
            return ResponseResult.errorResult(AppHttpCodeEnum.PARAM_INVALID);
        }
        if (statusCheck(status)) {
            return ResponseResult.errorResult(AppHttpCodeEnum.PARAM_INVALID);
        }
        ApUserRealname apUserRealname = new ApUserRealname();
        apUserRealname.setId(id);
        apUserRealname.setStatus(status);
        if (msg == null)
            apUserRealname.setReason(msg);
        apUserRealnameMapper.updateByPrimaryKeySelective(apUserRealname);
        //认证通过添加自媒体账号, 修改ap_user身份标记
        if (status.equals(AdminConstans.PASS_AUTH)) {
            //添加自媒体账号, 查询ap_user信息封装到wmuser中
            WmUser wmUser = apUserMapper.findWmUserInfoByAuthId(id);
            if (wmUser == null) {
                return
ResponseResult.errorResult(AppHttpCodeEnum.PARAM_INVALID);
            }
            int apUserId = wmUser.getId();
            wmUser.setId(null);
            wmUser.setCreatedTime(new Date());
            wmUserMapper.insertSelective(wmUser);
            //修改ap_user标记
            ApUser apUser = new ApUser();
            apUser.setId(apUserId);
            apUser.setFlag(1);
```

```
            apUserMapper.updateByPrimaryKeySelective(apUser);
        }
        return ResponseResult.okResult(AppHttpCodeEnum.SUCCESS);
    }

    private boolean statusCheck(Short status) {
        if (status == null
                || (!status.equals(AdminConstans.WAIT_AUTH)
                && !status.equals(AdminConstans.FAIL_AUTH)
                && !status.equals(AdminConstans.PASS_AUTH))) {
            return true;
        }
        return false;
    }
}
```

9.4 Dao层开发

9.4.1 Mapper文件修改

- app包下面的ApUserMapper.xml

```
<select id="findWmUserInfoByAuthId"
resultType="com.heima.article.mysql.core.model.pojos.wemedia.WmUser">
      SELECT id, name, password, salt, image, phone, 9 status, 0 type
      FROM ap_user
      where id = (SELECT user_id from ap_user_realname where id = #{id})
  </select>
```

- app包下面的ApUserRealnameMapper.xml

```
<select id="loadListByStatusAndParam"
resultType="com.heima.article.mysql.core.model.pojos.app.ApUserRealname">
    select
    <include refid="Base_Column_List" />
    from ap_user_realname
    <where>
        <if test="dto.params.userId != null" >
            and user_id = #{dto.params.userId}
        </if>
        <if test="dto.params.name != null" >
            and name = #{dto.params.name}
        </if>
        <if test="dto.params.idno != null" >
            and idno = #{dto.params.idno}
        </if>
        <if test="dto.params.status != null" >
            and status = #{dto.params.status}
        </if>
        <if test="dto.params.reason != null" >
            and reason = #{dto.params.reason}
        </if>
    </where>
    limit ${dto.params.from}, ${dto.size}
</select>
<select id="countAuthListByStatusAndParam" resultType="java.lang.Integer">
```

```
    select
    count(0)
    from ap_user_realname
    <where>
        <if test="dto.params.userId != null" >
            and user_id = #{dto.params.userId}
        </if>
        <if test="dto.params.name != null" >
            and name = #{dto.params.name}
        </if>
        <if test="dto.params.idno != null" >
            and idno = #{dto.params.idno}
        </if>
        <if test="dto.params.status != null" >
            and status = #{dto.params.status}
        </if>
        <if test="dto.params.reason != null" >
            and reason = #{dto.params.reason}
        </if>
    </where>
    limit ${dto.params.from}, ${dto.size}
</select>
```

10 实名认证前端开发

10.1 接口定义

10.1.1 审核列表

- 在src/constants/api.js中定义常量映射到后端请求地址

```
export const  API_AUTH_LIST = '/api/v1/admin/auth/list' //审核列表
```

- 在src/api/auth.js中定义请求方法（此处省略了引入刚才定义的常量，此后所有导入省略请自行导入需要的常量及方法）

```
export function findAuthList(data) {
  return new Request({
    url: API_AUTH_LIST,
    method: 'post',
    data
  })
}
```

10.1.2 通过审核

- 在src/constants/api.js中定义常量映射到后端请求地址

```
export const  API_AUTH_PASS = '/api/v1/admin/auth/authPass' //通过审核
```

- 在src/api/auth.js中定义请求方法（此处省略了引入刚才定义的常量，此后所有导入省略请自行导入需要的常量及方法）

```
export function authPass(data) {
  return new Request({
    url: API_AUTH_PASS,
    method: 'post',
    data
  })
}
```

10.1.3 驳回审核

- 在src/constants/api.js中定义常量映射到后端请求地址

```
export const  API_AUTH_FAIL = '/api/v1/admin/auth/authFail'
```

- 在src/api/auth.js中定义请求方法（此处省略了引入刚才定义的常量，此后所有导入省略请自行导入需要的常量及方法）

```
export function authFail(data) {
  return new Request({
    url: API_AUTH_FAIL,
    method: 'post',
    data
  })
}
```

10.2 路由及菜单调整

10.2.1 路由调整

在src/router.js中asyncRouterMap对象的children数组中增加以下改动，以满足全局自动记录路由的功能：

```
{
    path:'/auth/index',
    component: () => import('./views/auth/index.vue'),
}
```

10.2.2 菜单调整

在src/constants/menus.js的MenuData中添加一下内容

```
{
       title:'用户管理',path : '/auth/index' ,icon:'el-icon-user',
       children:[
           { title:'用户列表' , path : '/users/index'},
           { title:'用户审核' , path : '/auth/index'}
       ]
   },
```

10.3 开发实现

10.3.1 搜索工具组件定义

在src/views/auth/components/SearchTool.vue中进行创建

```
<template>
  <section class="filter">
    <div class="filter-container">
      <el-form ref="form">
        <el-form-item label="审核状态：" label-width="110px">
          <a
            v-for='item in stateList'
            :key="item.value"
            href="javascript:;"
            :class="['state_label',(item.value === selectState.value) ? 'active'
: '']"
            @click="changeState(item)">{{item.label}}</a>
        </el-form-item>
      </el-form>
    </div>
  </section>
</template>

<script>
  export default {
    props:["changeParam"],
    data() {
      return {
        stateList:[
          {label:'待审核',value:1},
          {label:'审核通过',value:9},
          {label:'审核失败',value:2},
        ],
        date:null,
        selectState:{
          //选择的筛选状态
          label:'待审核',value: 1
        },
        userName: ''
      }
    },

    methods:{
      queryData() {
        let params = {
          status: this.selectState.value
        }
        this.changeParam(params)
      },
      //切换状态
      changeState (state) {
        this.selectState = state //设置状态
        this.queryData() //查询数据
      }
    }
  }
</script>


<style rel="stylesheet/scss" lang="scss" scoped>
  .filter {
    background-color: #ffffff;
```

```
    text-align: left;
    border: 1px solid #e7e7e9;
    header {
      border-bottom: 1px dashed #cccccc;
      margin: 0 5px;
      padding: 0 10px;
      font-size: 14px;
      height: 55px;
      line-height: 55px;
      color: #323745;
    }
    .filter-container {
      overflow: hidden;
      .el-form {
        padding: 20px 20px 0;
        overflow: hidden;
        .el-form-item {
          margin: 20px 0
        }
      }
      .date-filter {
        padding: 25px 20px 20px;
        overflow: hidden;
        span {
          font-size: 14px;
          margin-right: 20px;
          height: 40px;
          line-height: 40px;
          float: left;
          cursor: pointer;
          &.active, &:hover {
            color: #3296fa;
          }
        }
        .time-container {
          float: left;
          position: relative;
        }
      }
    }
  }
  .state_label {
    float:left;
    padding-right:25px;
    font-size:14px;
    color:#333;
  }
  .active{
    color:#3296fa
  }
</style>
```

10.3.2 搜索结果组件定义

在src/views/auth/components/SearchResult.vue中实现

```vue
<template>
  <section class="filter">
    <div class="filter-container">
      <el-form ref="form">
        <el-form-item label="审核状态：" label-width="110px">
          <a
            v-for='item in stateList'
            :key="item.value"
            href="javascript:;"
            :class="['state_label',(item.value === selectState.value) ? 'active'
: '']"
            @click="changeState(item)">{{item.label}}</a>
        </el-form-item>
      </el-form>
    </div>
  </section>
</template>

<script>
  export default {
    props:["changeParam"],
    data() {
      return {
        stateList:[
          {label:'待审核',value:1},
          {label:'审核通过',value:9},
          {label:'审核失败',value:2},
        ],
        date:null,
        selectState:{
          //选择的筛选状态
          label:'待审核',value: 1
        },
        userName: ''
      }
    },

    methods:{
      queryData() {
        let params = {
          status: this.selectState.value
        }
        this.changeParam(params)
      },
      //切换状态
      changeState (state) {
        this.selectState = state //设置状态
        this.queryData() //查询数据
      }
    }
  }
</script>


<style rel="stylesheet/scss" lang="scss" scoped>
  .filter {
    background-color: #ffffff;
    text-align: left;
```

```
    border: 1px solid #e7e7e9;
    header {
      border-bottom: 1px dashed #cccccc;
      margin: 0 5px;
      padding: 0 10px;
      font-size: 14px;
      height: 55px;
      line-height: 55px;
      color: #323745;
    }
    .filter-container {
      overflow: hidden;
      .el-form {
        padding: 20px 20px 0;
        overflow: hidden;
        .el-form-item {
          margin: 20px 0
        }
      }
      .date-filter {
        padding: 25px 20px 20px;
        overflow: hidden;
        span {
          font-size: 14px;
          margin-right: 20px;
          height: 40px;
          line-height: 40px;
          float: left;
          cursor: pointer;
          &.active, &:hover {
            color: #3296fa;
          }
        }
        .time-container {
          float: left;
          position: relative;
        }
      }
    }
  }
  .state_label {
    float:left;
    padding-right:25px;
    font-size:14px;
    color:#333;
  }
  .active{
    color:#3296fa
  }
</style>
```

10.3.3 集成实现

```
<template>
  <div>
    <search-tool
```

```
      :changeParam="searchAuthList"
    />
    <search-result
      ref='mySearchResult'
      :authList="authList"
      :host="host"
      :total="total"
      :changePage="searchAuthList"
      :pageSize="params.size"
      :authPassRealName="authPassRealName"
      :authFailRealName="authFailRealName"
    />
  </div>
</template>

<script>
  import SearchTool from './components/SearchTool.vue'
  import SearchResult from './components/SearchResult.vue'
  import {findAuthList, authPass, authFail} from '@/api/auth'
  export default {
    name: "AuthManage",
    data() {
      return {
        params:{
          page:1,
          size:10
        },
        total:0,
        host:'',
        authList:[]
      }
    },
    created() {
      this.searchAuthList();
    },
    components: {
      SearchTool,
      SearchResult
    },
    methods: {
      async searchAuthList(newParams) {
        let res = await findAuthList({...this.params, ...newParams});
        if (res.code == 200) {
          this.authList = res.data
          this.host = res.host
          this.total = res.total //总记录数
        } else {
          this.$message({type: 'error', message: res.error_message})
        }
      },
      async authPassRealName(params) {
        let res = await authPass(params)
        if (res.code == 200)
          this.$message({type: 'success', message: '操作成功'})
        else
          this.$message({type: 'success', message: res.error_message})
        this.searchAuthList();
      },
```

```
      async authFailRealName(params) {
        let res = await authFail(params)
        this.searchAuthList();
      },
    }
  }
</script>
<style scoped>
</style>
```