**程序结构设计**

一、整体结构设计

程序按照功能模块设计，主要分为以上五个模块：(以下只是简单说明，可自行扩展)

登录及个人信息管理模块：主要实现了用户的登录，注册，修改密码等操作。并在个人信息编辑的界面提供了个人信息的填写修改以及头像上传功能。

首页管理模块：通过动态标签页，实现了视频的分类卡片式展示，并提供了各个操作的入口。

视频播放和发布模块： 该模块主要实现了视频及对应信息的上传功能，并通过h5的video视频播放标签，实现了视频的读取，播放以及播放控制等功能。

视频评论模块：实现了用户对各个视频的评论功能，并提供评论区展示其他用户的评论。

用户交友模块：设计聊天框，实现了用户与用户之间聊天沟通，实现了交友功能。

二、详细模块设计

**1.登录模块设计**

登录模块主要包含了账号管理和用户个人信息管理两个模块，其对应了数据库的两张表：账号信息表sys\_user和宠物信息表t\_pet\_info， 两者通过宠物信息表的user\_id外键字段连接起来，即一个账号关联一个宠物信息。

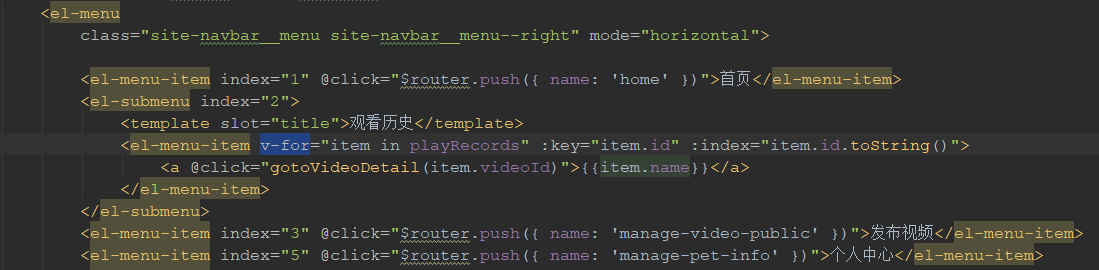
在账号管理中，主要实现了登录，注册以及密码修改功能。

在登录模块中，当用户输入账号密码并点击登录时，会将对应的账号信息发送到后端，后端进行数据校验判断正确性，失败则提示密码错误的错误信息，由前端进行渲染。当成功时，则返回成功信息，前端会跳转到网站首页，并关联显示当前的用户信息。

而注册和密码修改模块也是由前端收集账号信息，通过ajax发送到后台，实现了账号信息的新增和修改操作。而注册操作中，不仅新增账号信息表的信息，同时，还会在宠物信息表新增一条关联数据，方便后续操作。

在网站中，还提供了个人信息的编辑界面。(此处添加界面截图) 通过element-ui的<el-form>标签，实现了用户信息的填写和手机，通过<el-upload>标签实现了用户头像的上传，并保存后台返回的url路径。在用户点击提交后，通过http发送消息包体，到后端进行更新操作。

**2．首页模块设计**

1. 首页顶栏采用<el-menu>菜单标签页，设计排列的个人信息，视频发布，观看历史等入口，并通过router.push的方式进行路由跳转，跳转到指定的html页面。

2. 而首页采用<el-tabs>的标签页，通过vue的v-for循环遍历的方式，循环遍历从后台获取的视频分类列表，实现了动态标签页的渲染。而针对视频，同样采用v-for的方式，遍历视频列表，通过<el-card>设计卡片样式，展示视频的截图和部分信息，供用户浏览。同样的，采用router.push的方式，当用户点击相应卡片时，会带着视频的id参数，跳转到制定的视频页面，获取对应id的视频信息进行渲染。

<el-tab-pane  
 v-for="item in tabList"  
 :key="item.dictCode"  
 :label="item.dictName"  
 :name="item.dictCode.toString()">  
 <el-row>  
 <el-col :span="3" v-for="(o, index) in videoList" :key="o.id" :offset="index > 0 ? 1 : 0">  
 <el-card :body-style="{ padding: '0px' }">  
 <img class="image" :src="$http.adornUrl(o.filePic)" @click="gotoVideoDetail(o.id)">  
 <div style="padding: 14px;">  
 <span>{{o.name}}</span>  
 <div class="bottom clearfix">  
 <time class="time">{{ o.createTime }}</time>  
 </div>  
 </div>  
 </el-card>  
 </el-col>  
 </el-row>  
</el-tab-pane>

**3. 视频发布和播放模块设计**

1. 视频发布使用<el-form>表单标签，提供用户填写视频相关信息，并使用<el-upload>文件上传标签，通过设定accept参数，限制上传的格式为视频格式，实现了视频文件的上传功能，并通过handleVideoSuccess的回调接口，获取保存视频的url路径。

当用户单击提交时，发送http包到后台进行处理保存。

<el-card class="box-card" style="width: 50%;left: 100px;position: relative">  
 <el-upload  
 ref="uploadVideo"  
 :action="uploadVideo()"  
 :show-file-list="true"  
 :auto-upload="true"  
 :on-success="handleVideoSuccess"  
 :limit="1"  
 :on-exceed="handleVideoOver"  
 accept=".mp4,.avi,.rmvb,.rm,.mpeg,.3gp,.wmv,.flv">  
 <el-button slot="trigger" size="small" type="primary">选取文件</el-button>  
 </el-upload>  
</el-card>

2. 在视频播放中，使用的是h5封装好的视频播放标签<video>，只需要设定对应的参数：视频url地址src，视频的封面poster和视频的控制按钮controls即可实现视频的获取，展示以及播放功能。

<video v-if="videoSrc" :src="videoSrc" :poster="headSrc" controls="controls" class="vd-video">  
</video>

但是由于该项目是前后端分离，会出现静态文件获取不到的情况，因此需要在后端的webconfig的文件的addResourceHandlers接口中，将包含upload和petfile的http请求转换成请求本地对应路径的文件。

public void addResourceHandlers(ResourceHandlerRegistry registry) {  
 registry.addResourceHandler("/upload/\*\*").addResourceLocations("file:" + "/upload/");  
 registry.addResourceHandler("/petfile/\*\*").addResourceLocations("file:/petfile/");  
 WebMvcConfigurer.super.addResourceHandlers(registry);  
}

**4. 视频评论模块设计**

(添加页面截图)

视频评论模块关联的数据库表主要是t\_video\_comment视频评论表，表除了评论需要的评论人和评论内容等信息，还关联了视频的video\_id，实现了评论的归属。

当用户进入视频页面后，下方会出现评论列表和评论框。评论列表的渲染采用v-for的方式，当视频页面载入时，同时会请求获取视频评论的列表commentList,并通过v-for的方式，在<el-row>的行布局样式中填充评论内容，按行渲染出来。

<el-row style="margin-top: 20px" v-for="item in commentList" :key="item.id" class="vd-comment-detail">  
 <!--头像区-->  
 <el-col :span="2">  
 <div>  
 <img class="vd-comment-head" v-if="item.commentPic" :src="$http.adornUrl(item.commentPic)">  
 <img class="vd-comment-head" v-else src="~@/assets/img/avatar.png">  
 </div>  
 <div>  
 {{item.commentName}}  
 </div>  
 </el-col>  
 <el-col :span="22" style="padding-top: 10px; padding-left: 10px">{{item.content}}</el-col>  
</el-row>

而评论提交框使用的是textarea样式的input框，实现用户评论的多行填写功能。在点击提交后，会在t\_video\_comment视频评论表插入一条对应的评论记录。

<el-input type="textarea" :row="4" class="vp-textarea" v-model="commentContent"></el-input>

**5. 用户交友模块**

用户交友模块的设计包含两张数据库表，分别是聊天关系表t\_pet\_chat和聊天记录表t\_chat\_record。 聊天关系表主要包含了聊天双方的信息，建立聊天关系。而聊天记录表则外键聊天关系表的id主键，实现了针对聊天双方的聊天记录的保存，这样聊天双方就都可以看到聊天记录了。

(聊天框的图，包含一些聊天信息)

在聊天框设计中，通过查询当前用户的聊天关系表，渲染成左侧的聊天关系列表，用户可以点击对应的聊天窗口，与指定的用户进行聊天，实现了用户的多对象聊天。

<el-menu unique-opened @select="handleChatSelected">  
 <el-menu-item v-for="(item,index) in chatList"  
 :index="index.toString()" :key="index">  
 <span v-if="item.onePet === userInfo.id">{{item.secondName}}</span>  
 <span v-else>{{item.oneName}}</span>  
 </el-menu-item>  
</el-menu>

在指定聊天对象后，会请求后台对应聊天组的聊天记录，并在聊天窗口中显示出来。而用户可以在下面添加对应的聊天记录进行发送，系统会将对应的信息保存到后台数据库中，并更新前台的页面，刷新刚发送的聊天信息，实现了双方的聊天功能。

6. 历史记录模块(这个我不知道需不需要另起一个模块，还是跟上面的某个合并，看你需求)

在用户首次点击进入视频详情时，会在历史记录表t\_play\_record中插入一条播放记录，记录用户信息和该视频的信息。

在用户登录后，会请求用户最新的5条观看记录，在顶端菜单栏通过el-submenu标签，实现下拉显示播放记录的功能。并将对应的条目通过<a>标签渲染成可点击的链接标签，当用户点击后，会跳转到对应的视频详情页。

<el-submenu index="2">  
 <template slot="title">观看历史</template>  
 <el-menu-item v-for="item in playRecords" :key="item.id" :index="item.id.toString()">  
 <a @click="gotoVideoDetail(item.videoId)">{{item.name}}</a>  
 </el-menu-item>  
</el-submenu>

**后端层级设计**：

(具体可以针对每一层进行扩展，百度“java controller service dao作用”)

controller层：用来接收前端的请求，并调用对应的service层接口

service层：进行逻辑处理，并调用dao层接口，获取需要的值进行返回。

dao层：定义sql接口，其中sql在对应的xml中实现，主要用来编写sql语句，跟数据库进行通信。

entity： 数据库表对应实体类的定义。

vue，element\_ui的介绍，也写一些。

(vue的主要特性，双向绑定，常用v-bind, v-model, v-for, v-if等)