**七牛云项目架构设计文档**

**一 项目介绍**

* 后端基于gin框架
* 前端使用vue框架

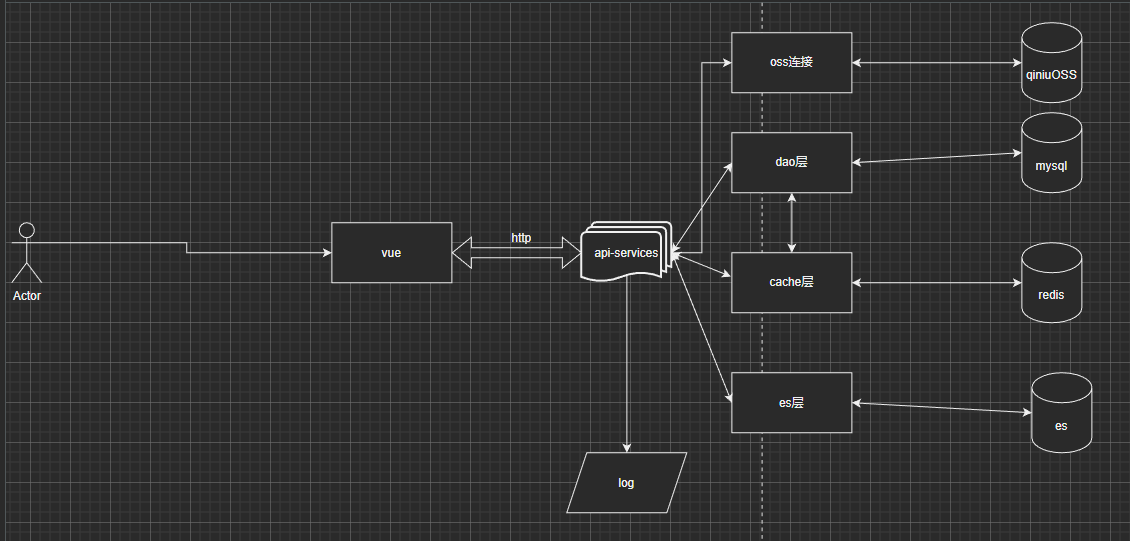
**二 项目分工**

|  |  |
| --- | --- |
| **成员** | **分工** |
| 阮航 | 项目选型+主要接口 |
| 王瑾 | 剩余接口+接口测试+结构优化 |
| 王怀强 | 前端实现 |

**三 项目实现**

**1.架构设计**

**1.1业务架构**



**1.2后端代码结构**

|  |
| --- |
| XML . ├─api 数据传输对外接口 ├─cmd 项目启动文件 ├─config 配置文件 ├─logs 日志输出 ├─middleware 中间件 ├─pkg 错误定义与工具包 │ ├─e │ └─util ├─repository 仓库 │ ├─db │ │ ├─dao 数据持久化层 │ │ └─model 模型定义 │ └─oss 对象存储 | └─es es  | └─cache 缓存  ├─routes 路由层 ├─service 业务层 ├─types 请求与响应规范层 └─ main.go 启动文件 |

**2 技术选型**

考虑到项目比较小，后端选择使用单体架构

· 后端：gin+gorm

前端： vue

**项目优点**

* 使用前后端分离
* 清晰的项目架构
* 高可读，易维护
* 方便结合docker部署
* 日志输出和错误定义与工具包，有助于更好地监控和解决问题

**项目结果**

* 基础功能实现
* 高级功能实现
* 通过七牛云添加水印
* 添加弹幕功能，增加互动性
* 允许用户添加好友
* 用户可投稿

**3 项目实际实现**

* 使用swag编写并生成接口文档

|  |
| --- |
| XML // @Summary 加入收藏夹 // @Description 通过表单提交将视频加入收藏夹 // @ID FavlistAdd // @Accept x-www-form-urlencoded // @Produce json // @Param request body types.FavlistAddReq true "想要添加的视频id和文件夹id" // @Header 200 {string} Token "我的token" // @Success 200 {object} Response // @Failure 400 {object} ErrorResponse // @Router /interaction/fav [put] func FavlistAdd(ctx \*gin.Context) {  var req \*types.FavlistAddReq  //ctx.ShouldBind(&req) 获取前端输入的表单信息  if err := ctx.ShouldBind(&req); err != nil {  ctx.JSON(http.StatusBadRequest, types.ErrorResponse(err))  //打日志  util.LogrusObj.Error(err)  return   }  // 获取userSrv对象  srv := service.GetInterSrv()   resp, err := srv.FavlistAdd(ctx.Request.Context(), req, util.GetUidInToken(ctx))  if err != nil {  util.LogrusObj.Error(err)  ctx.JSON(http.StatusOK, types.ErrorResponse(err))  return  }  //返回给前端相应信息  ctx.JSON(http.StatusOK, types.RespSuccess(ctx, resp, http.StatusOK)) } |



* 使用 golang.org/x/crypto/bcrypt包对密码进行加密

|  |
| --- |
| XML // SetPassword 设置密码加密 func (u \*User) SetPassword(password string) error {  bytes, err := bcrypt.GenerateFromPassword([]byte(password), PassWordCost)  if err != nil {  return err  }  u.PasswordDigest = string(bytes)  return nil } |

* 使用github.com/h2non/filetype包对用户输入进行内容检索

|  |
| --- |
| XML // 使用文件类型检测库来判断文件类型 kind, \_ := filetype.Match(buffer)  // 如果文件类型是图片类型，返回true if kind.MIME.Type == "video" {  return true, nil }  // 否则，不是图片文件 return false, nil |

* 对用户关注之类高频读写信息添加缓存，达到快速响应

|  |
| --- |
| XML // AddFollow 关注关系缓存 func AddFollow(ctx context.Context, uId, followId uint) error {  return RedisClient.  Set(ctx, GenFollowUserCacheKey(uId, followId), 1, 7\*24\*time.Hour).  Err() } |

* 推荐功能使用redis进行用户画像维护，记录用户点赞数据，以达到视频推荐

|  |
| --- |
| XML // 维护用户画像的函数 func UpdateUserProfile(ctx context.Context, userKey string, tagID int64, score float64) error {  // 设置用户画像中的标签分数  z := redis.Z{  Score: score,  Member: tagID,  }  \_, err := RedisClient.ZAdd(ctx, "user:"+userKey, z).Result()   if err != nil {   return err  }   return nil } |

* 使用redis维护热门视频榜单，并设定定时任务，每24小时刷新

|  |
| --- |
| XML // AddPopularVideo 热门视频 func AddPopularVideo(ctx context.Context, videoID int64, score float64, createTime time.Time) error {  CleanUpOldVideos(ctx)  // 将视频ID和创建时间添加到一个Hash中  RedisClient.HSet(ctx, "video\_create\_time", fmt.Sprintf("%d", videoID), fmt.Sprintf("%d", createTime.Unix()))   // 使用 ZINCRBY 命令来增加或更新视频的分数  intCmd := RedisClient.ZIncrBy(ctx, "popular\_videos", score, fmt.Sprintf("%d", videoID))  if intCmd.Err() != nil {  fmt.Printf("更新视频热度失败: %v\n", intCmd.Err())  }   return nil } |

* 使用es进行视频与用户搜索

|  |
| --- |
| XML // VideoTitleRetrieve 页码，页显示条数，视频标题 func VideoTitleRetrieve(page int, limit int, name string) ([]int64, error) {  if limit == 0 {  limit = 15  }  if page <= 0 {  page = 1  }  from := limit \* (page - 1)  query := elastic.NewMatchQuery(`nick\_video\_title`, name)  util.LogrusObj.Printf("Retrieving video titles: page=%d, limit=%d, name=%s\n", page, limit, name)  var res, err = EsClient.Search(models.VideoModel{}.VideoIndex()).Query(query).From(from).Size(limit).Do(context.Background())  if err != nil {  return nil, err  }  var ids []int64  for \_, hit := range res.Hits.Hits {  var source map[string]interface{}  err = json.Unmarshal(hit.Source, &source)  if err != nil {  return nil, err  }  if id, ok := source["video\_id"].(float64); ok {  ids = append(ids, int64(id))  } else {  util.LogrusObj.Println("Warning: could not convert video\_id to float64")  }  }  return ids, nil } |

* 错误定义，本项目有完善的错误定义(位于server/pkg/e)

|  |
| --- |
| XML SUCCESS = 200 UpdatePasswordSuccess = 201 NotExistInentifier = 202 ERROR = 500 InvalidParams = 400 ERRORNULL = 410  //成员错误 ErrorExistNick = 10001 ErrorExistUser = 10002 ErrorNotExistUser = 10003 ErrorNotCompare = 10004 ErrorNotComparePassword = 10005 ErrorFailEncryption = 10006 ErrorUserNotFound = 10010 ... |

|  |
| --- |
| XML ...  ErrorUserNotFound: "用户不存在",   ErrorUploadFile: "上传失败",  ErrorAdminFindUser: "管理员查询用户失败",   ErrorDatabase: "数据库操作出错,请重试",   ErrorOss: "OSS配置错误", }  // GetMsg 获取状态码对应信息 func GetMsg(code int) string {  msg, ok := MsgFlags[code]  if ok {  return msg  }  return MsgFlags[ERROR] } |

* 检测用户是否上传封面，没上传则使用七牛云视频截帧当封面

|  |
| --- |
| XML if imageF != nil {  b, err = s.IsImageFile(imageF)  if b != true || err != nil {  return  }  iFile, err = FileHeaderToBytes(imageF)  if err != nil {  return  }  coverURL, err = oss.AddImage(userId, req.Title, iFile)  if err != nil {  return  } } else {  coverURL = videoUrl + "?vframe/jpg/offset/1" } |

* 通过七牛云事件实现转码与添加水印

