未命名

**本科生毕业论文（设计）**

题目：游戏玩家在线交流平台设计实现

学生姓名： 白凯发

指导教师： 王勇

学 院： 软件学院

专 业： H5

年 级： 2016

完成日期： 2020 年 月 日

摘 要

目的:在对游戏玩家在线交流平台设计实现快速搭建技术的探究过程中，为了达到快速高效的开发，以较低的技术成本，较高的效率以及为了达到能够方便玩家交流，方便官方发放游戏公告的目的。

方法:此次探究了前端两大大框架，通过对比论证的手法，来彰显出使用某种技术的优势，这会在主题部分进行详细阐述，在页面展示上，通过多种不同多样化的内容，丰富加深用户的使用感受。

结果与结论:通过参考其它交流网站，主要通过设置精美背景来增加网站的美观度，通过轮播，新闻等等方式丰富网站的内容，通过比对vue与reacat在**开发效率**上的异同，以及后端关于**nodejs**应当使用什么相关技术的一些辩论。通过比较各自的优缺点最终选择了**react**框架，配合的数据存储框架是**mobx**，后端则是选择nodejs以缩小开发的技术成本。最终实现了一个针对游戏玩家的在线交流平台，通过这次实践发现技术选型非常决定开发的效率，这在企业级开发中非常重要，**高效**的开发能够使我们尽可能快的完成任务，抢占市场先机。

**Abstract**

Objective: in the process of exploring the design and implementation of the online communication platform for gamers, in order to achieve the purpose of rapid development and facilitate the exchange of gamers.

Methods: this time, we explored the two front-end frameworks, and demonstrated the advantages of using a certain technology by means of comparative argumentation. This will be elaborated in the theme part, and on the page display, we will enrich and deepen the user's use experience through diversified content.

Results & CONCLUSION: by referring to other communication websites, we mainly increase the beauty of the website by setting beautiful background, enrich the content of the website by means of rotation, news, etc., compare the similarities and differences between Vue and reacat in **development efficiency**, and some debates on what relevant technologies should be used by **nodejs** in the back end. By comparing their advantages and disadvantages, we choose react framework, **mobx** as the data storage framework, and nodejs as the backend to reduce the development cost. Finally, an online communication platform for game players is realized. Through this practice, it is found that technology selection determines the efficiency of development, which is very important in enterprise level development. **Efficient development** can enable us to complete the task as soon as possible and seize the market opportunity.

目录

[第1章 绪论 1](#_Toc9503)

[1.1课题存在的背景以及意义 1](#_Toc17470)

[1.2市场存在的平台现状 1](#_Toc14683)

[第2章 网站需求分析 1](#_Toc17737)

[第3章 后端开发 3](#_Toc30624)

[3.1 技术框架设计 3](#_Toc22121)

[3.1.1 数据库设计 4](#_Toc12306)

[3.1.2 静态资源文件存储 5](#_Toc31422)

[3.1.3 后端路由规划 5](#_Toc9907)

[3.1.4 前台内容解析 6](#_Toc1563)

[3.1.5 实时通信 6](#_Toc26746)

[3.1.6 用户校验（1） 6](#_Toc69)

[3.1.7用户校验（2） 6](#_Toc31265)

[3.2技术选型的原因： 6](#_Toc3850)

[第4章 前端UI设计 7](#_Toc13897)

[第5章 前端程序开发 8](#_Toc1613)

[5.1技术选型 8](#_Toc7660)

[5.2前端开发 9](#_Toc30011)

[5.2.1技术框架设计 9](#_Toc13623)

[5.2.2技术选型的原因 10](#_Toc571)

[5.2.3注册模块 10](#_Toc6198)

[5.2.4个人中心 11](#_Toc13151)

[5.2.5热点轮播 12](#_Toc3589)

[5.2.6热门咨询 13](#_Toc28511)

[第六章 总结 14](#_Toc22111)

[6.1后端总结 14](#_Toc9059)

[6.1.1后端技术总结 14](#_Toc3262)

[6.1.2后端设计总结 14](#_Toc7207)

[6.2前端总结 15](#_Toc5486)

[6.2.1前端技术总结 15](#_Toc21827)

[6.2.2前端设计总结 16](#_Toc27709)

[参考文献 17](#_Toc22419)

[附 录 18](#_Toc10674)

[致 谢 20](#_Toc18463)

1. 绪论

1.1课题存在的背景以及意义

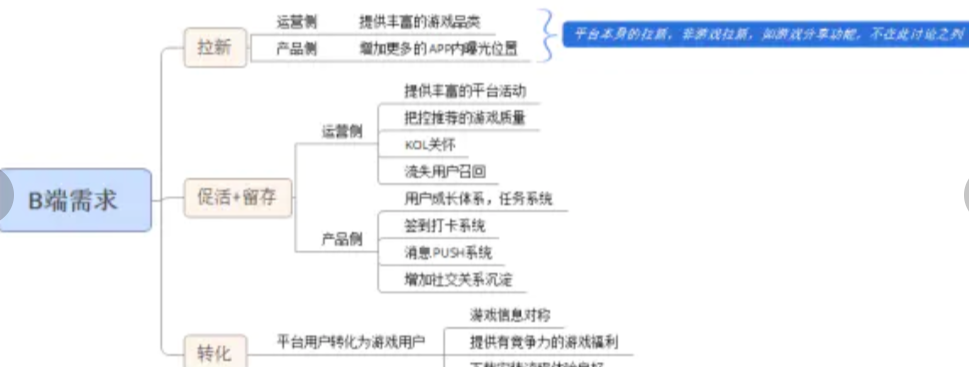
21世纪是一个电子竞技的新时代，我们通过电竞愉悦身心，交流感情，在工作之余作为一种休憩的手段，体验者新的技术给我们带来的种种好处，在这其中，平台起到了不可或缺的作用，通过平台，我们总是能第一手的获取到最新的游戏资讯，通过平台，官方总是能及时的对出现的各种状况问题第一时间反馈给玩家，通过平台，玩家们能很好地交流彼此的新的，这给游戏带来了巨大的流量，同时也增强了游戏的趣味性，我们使用着如此便利的内容，然而我们却很少去思考平台的背后是什么内容在支撑着，一个好的平台应该是怎样的，一个好的平台应该有怎样的技术支持呢？从电子竞技诞生至今，从前端以及后端诞生至今，有哪些好的内容值得我们去关注，我们应该怎么利用这些内容区搭建一个好的在线交流平台，这将会是本文着重讨论的内容，本文将从前端内容以及相关的各种技术起手，来实现在线交流平台。

1.2市场存在的平台现状

我们以腾讯的wegame平台为例，wegame平台不仅仅是一个玩家交流平台，更准确的称呼，wegame是一个综合性的平台，如果做一个集合的话，wegame>玩家游戏交流平台，但是虽然如此，wegame仍然在玩家交流这一方面做的可圈可点。首先就是它的咨询，轮播各种实施热点，再配合一些图片，能够较为清晰地表达出想表达的内容，比起其它传统的只有文字的表现形式，自然要更加吸引人，下面就是一些文字性的新闻内容以及一些近期的赛事，热门视频，搞笑视频，解说视频等等。翻开下一页，我们可以看到有商店，在商店里面，我们可以对我们玩的某款游戏做一个评价，或者去体验新的游戏，不足之处在于宣传内容和实际内容并不是完全符合，这就让人感觉大相径庭。总体来说目前的已存在的平台做的已经比较完善，我们可以借鉴做的好的地方，对于这些虚假宣传的内容我们要尽量抵制，尽量给玩家还原真实的环境。

第2章 网站需求分析

首先我们从一张图中看出用户的需求



根据马斯洛需求层次理论， 玩家的需求大致可以分为一下几类，基础需求，进阶需求，高级需求。其中基础需求是所有类型的玩家都必须满足的需求。因为如果没有基础需求其他的需求便都不存在，基础需求是进阶需求和高级需求的基础，但是进阶需求和高级需求比较重要，为了保证了游戏平台功能够持续的并且累积的活跃，形成一个完整生态闭环进阶需求和高级需求是关键。但是进阶需求和高级需求得视用户群体而定，如果平台游戏用户依旧为普通玩家，产品迭代重心应该还是在完善基础需求上面。

我们从玩家需要打发玩游戏时间这一需求上来分析



对应路径是这样的：

随便看看的类型：游戏平台不仅仅可以增加各类内容排行榜，并且可以增加各种个性化推荐（我们举个例子来说，比如基于地理位置推荐，更比如周围人在玩的游戏），游戏详情页，游戏分类等等这都是写好的思路。

查看最近新鲜的平台福利活动：福利活动是非常重要的一个环节，玩家们很有可能根据福利活动决定多玩或少玩游戏，对游戏的粘性有很大关系，首页增加福利模块，消息系统push重大福利活动

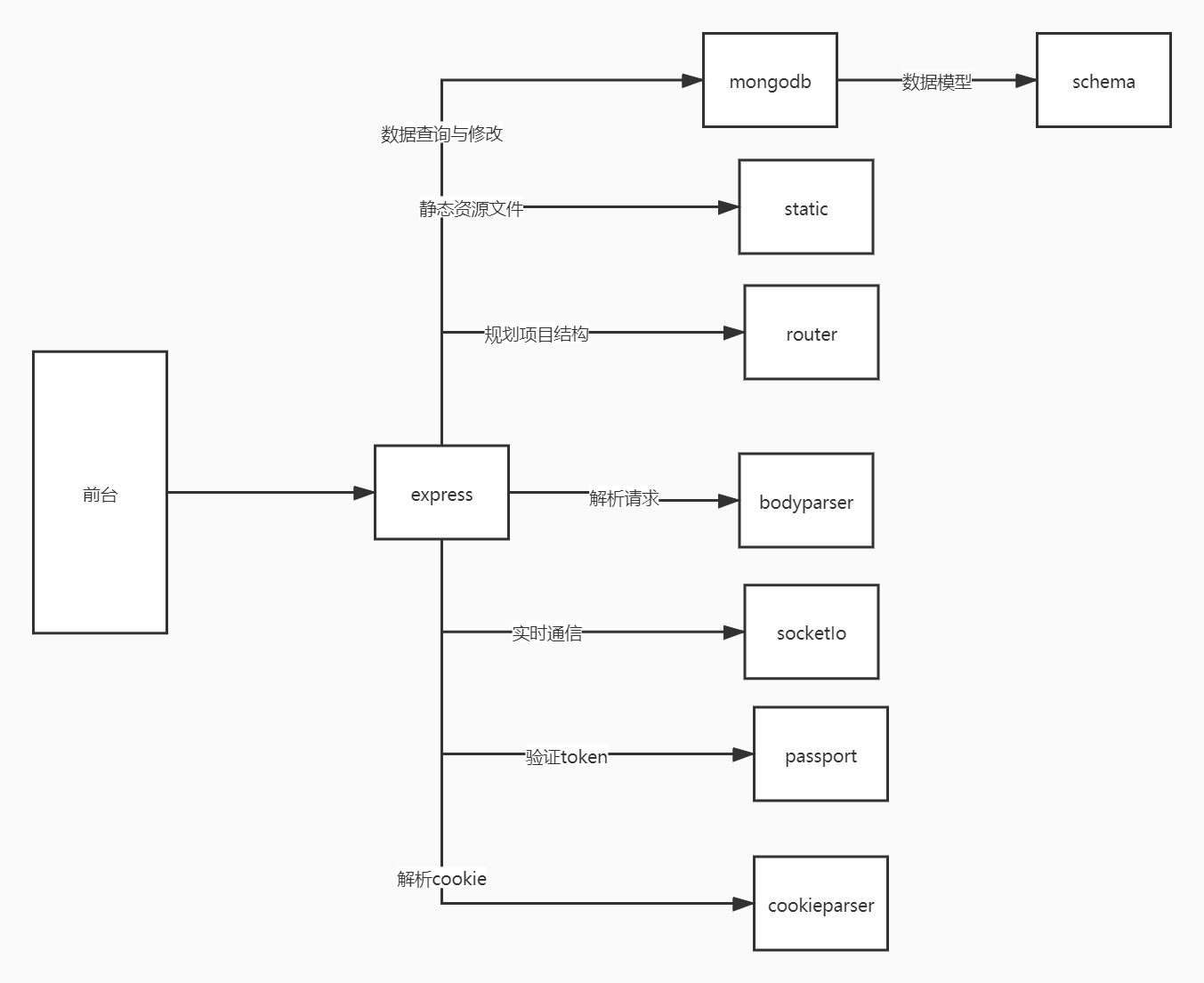
查看发布的帖子：比如说新增论坛，同时也需要支持查看历史回帖等消息

和其他的线上团体成员之间的交流：增加例如社群功能，支持即时通讯和短消息通讯等等

第3章 后端开发

3.1 技术框架设计

以下是整体的架构设计图



前台的内容在此图中不作介绍，后端开发，首先是架构设计，架构上来说，首先要确定是使用express还是koa，鉴于对nodejs掌握更好，对koa接触较少，虽然在小程序开发过程中接触过koa，但是从整体上来看，国内社区更偏向于express，而且express由于其历史悠久，因此各种中间件非常完善，这大大降低了我们的开发难度，基于以上几点考虑因此选择nodejs，系统架构上后端一定要覆盖住所有的内容。

3.1.1 数据库设计

鉴于前端至少会实现登录注册，因此会涉及数据查询，配合mongodb，mongodb由c++编写，分布式文件存储数据库，鉴于其拥有超强的扩展性，因此使用起来非常方便。Mongodb通过schema来存储对象，相对于关系型数据库，这种存储的好处是更接近原生的数据类型，容易操作，效率不逊色与关系型数据库。在实际使用的过程中，我会将用户信息存储为一个schema，例如

const mongoose  = require('mongoose');

const Schema = mongoose.Schema;

//Create Schema

const UserSchema = new Schema({

    name: {

        type: String,

    },

    email: {

        type: String,

    },

    password: {

        type: String,

    },

    avatar: {

        type: String,

    },

    identity: {

        type: String,

    },

    date: {

        type: Date,

        default: Date.now

    }

})

module.exports = User = mongoose.model("users", UserSchema)

可以看到在这里我对用户设计了头像，用户可以通过上传图片作为自己的头像，日期作为用户的一个回忆事件，记录了用户第一次来到这里，用户看到这里会联想到自己在这里的点点滴滴，利用用户的回忆增强对平台的感情。

3.1.2 静态资源文件存储

Static是我们存储静态资源的位置，用户会想要存储自己收藏的一些图片资源，一些好的工具文件等等，我们所有的图片资源文件等等都会存储到这里

3.1.3 后端路由规划

router仅仅是对后端的目录的一个优化，主要为了方便开发人员，对路由进行分类，方便开发人员快速找到相应的内容，快速的对需要更改的项目进行修改。

3.1.4 前台内容解析

用于解析客户端请求的body中的内容,内部使用JSON编码处理,url编码处理以及对于文件的上传处理。

3.1.5 实时通信

在玩家交流平台中，加入实时通信功能，会给用户带来更好的体验，媒体很重要的一点就是实时性非常重要，这会极大地增加客户的粘性，对于游戏玩家交流平台，这是必不可少的一部分内容，在评论这里，我们会加入实时讨论这一功能，这需要socketio的支持，

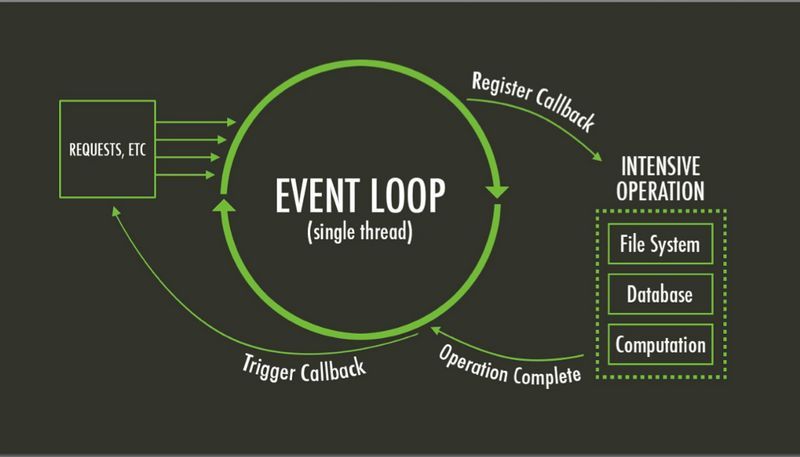
3.1.6 用户校验（1）

passport相对应的就是Jwt-oken,这是由于http协议是无状态的，客户端每次访问都是新的请求。每次请求都需要进行身份验证，传统方式是用session+cookie上面介绍过，而JWT就是更安全方便的方式。它的特点简洁，紧凑，不占空间，传输速度快,有利于多端分离，接口的交互，JWT是一种Token规范，主要面向的还是登录、验证和授权方向，同时可以用只来传递信息。

3.1.7用户校验（2）

cookie的作用主要是我们发送post请求到后端，后端验证合法，返回响应。

3.2技术选型的原因：

由于之前在老师的帮助下，已经进行过实战项目的开发，因此毕业设计课题定为游戏玩家在线交流平台设计实现,后台，我则是选择了nodejs，选择nodejs的原因主要是Nodejs区别于传统应用1. 它是异步事件驱动2.单线程，3.非堵塞I/O，4.性能出众。Nodejs的事件机制是这样的

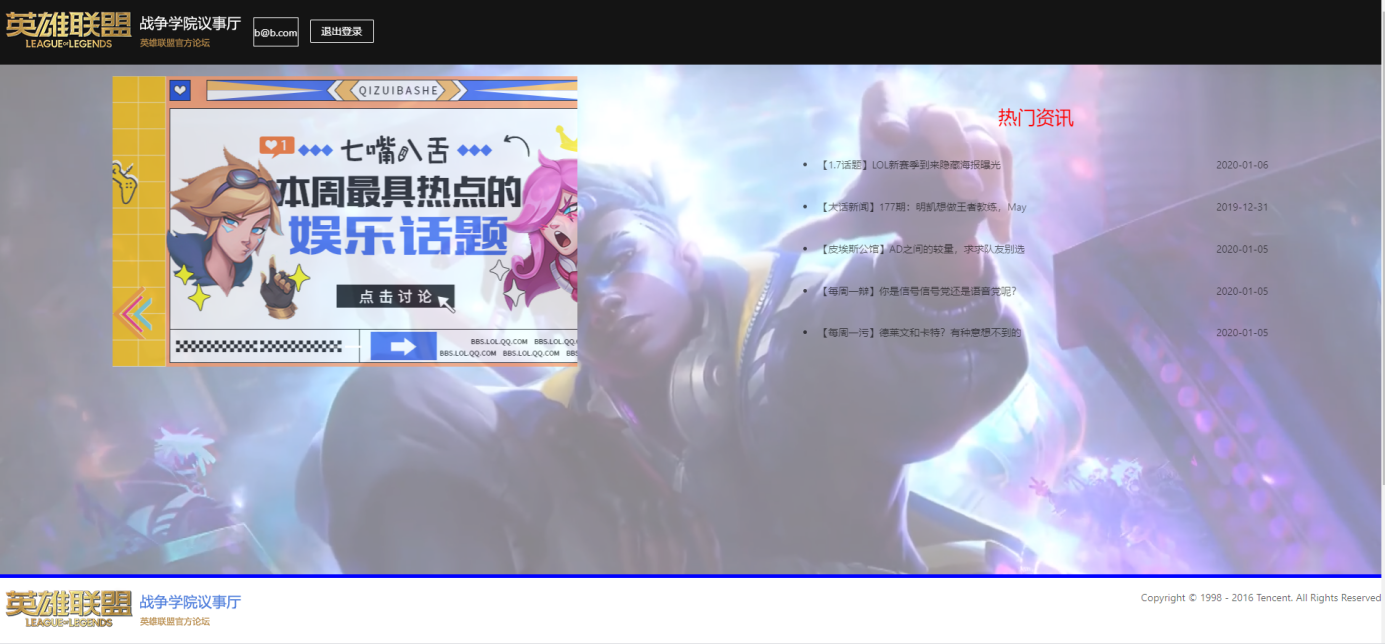
我们的场景类似于在线多人聊天，然而通过查阅文档我们知道，nodejs

比较适合I/O密集型的应用，例如多人在线小游戏，**在线多人聊天**，实时新闻，博客，微博等等之类的。我们的项目属于在线多人聊天，自然选择nodejs有着得天独厚的优势，再者是因为第三方的支持非常多。

第4章 前端UI设计

4.1设计ui

优美的界面非常重要，再开始写我们的项目之前，首先要设计好我们的ui，参考已有的游戏论坛平台，我设计了三分的结构



通过图片我们看到，三分的结构，好处主要是用户能非常方便的登入登出，通过顶层的navigation快速定位到自己想要查找或者操作的内容，同时，顶头的邮箱可以显示出用户的登入登出状态。

为了将最新的实时热点显示出来，我才用了轮播图，轮播图相比于单纯的文字展示，好处是通过图片，用户能够更清晰的知道此篇文章的内容是什么，同时提高了我们网站的浏览占比，充实了网站的内容，浏览的完整度变高，给用户提供了足够的优质信息。

但是由于轮播占的篇幅较大，并不能承载足够多的信息，于是我们在右侧增加tab来极大的丰富我们的内容，这是一种对称的网站布局

国人讲究对称美，这对于吸引我们的用户是大有益处的。

最后是所有官方网站都有的内容，网站的授权，以及网站的logo及网站的主要作用的标注。

第5章 前端程序开发

5.1技术选型

5.1.1框架选取

首先前端三框架中我选用的是react。

它的优点主要有

1、React速度很快：不直接操作DOM，而是引入了虚拟DOM，性能非常好。

2、跨浏览器兼容：虚拟DOM帮助我们解决了跨浏览器的问题，提供了标准化的API，以至于在IE8中都可以正常运行。

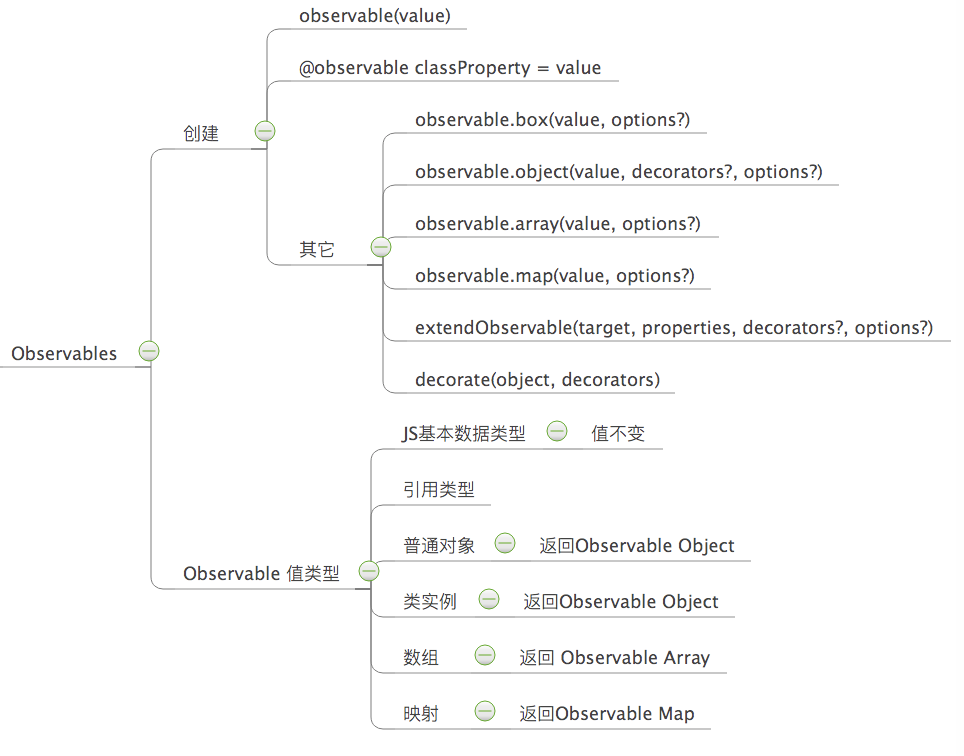
3、一切都是组件：模块化的代码，重用代码容易，可维护性比较高。

对于数据框架我选择了mobx，Mobx设计则更多偏向于面向对象编程和响应式编程，通常是将状态包装成可观察的对象，于是我们拥有了可观察对象的所有能力，一旦状态对象变更，就能自动获得更新。Mobx则通常按模块将应用状态划分，在多个独立的store中管理。这样做的好处就是使用非常方便，随用随存随取。Mobx中可以监听可观察对象，当其变更时将自动触发监听。使用起来非常舒服。

5.2前端开发

5.2.1技术框架设计

下面为我们的框架设计图



想要写好一个项目，一个好的框架必然不可少，为了使用上react hooks，因此采用mobx-react-lite来兼容之前的书写格式。这里描述下具体的兼容问题在哪里，我没有采用使用上下文的方式来引入mobx,因为上下文的方式首先增加了更多的api，其次，上下文的这种方式并不便于理解与掌握整体的架构，再者上下文的方式使得我们回到了过去，仓库之前没有分离开来，非常不便与我们对某些文件下某些具体数据的操作，冗余度高。因此采用mobx-react-lite是至关重要的，mobx-react-lite是通过useContext，createContext，useStores，useObserver这些api来保证我们在组件中可以方便的使用，对比之前的@inject('')与@observer,完美的配合在了一起，无论想使用类组件亦或者是函数组件，都能轻松自如，游刃有余的使用我们的公有数据。

5.2.2技术选型的原因

项目依赖antd和装饰器语法，因此采用customize-cra工具来简化这一操作。

customize-cra的优缺点主要有以下几点

优点:1.不需要npm run eject打开webpack配置来进行操作，减少了配置时间

1. 一次配置，多次使用，每次更换项目只需要移动config-overrides.js文件即可，方便快捷

缺点:主要缺点就是不打开webpack配置，无法进行一些精细化的操作，有些配置不存在于customize-cra中，当我们需要对webpack进行精细化配置的时候，customize-cra就显得捉襟见肘了。

5.2.3注册模块

1. 选取视频作为页面背景，此举是为了增加画面的优美程度，浏览过许多漂亮优美的网站，无非是动态，绚丽的界面能增加网站的体验，使用视频背景是最省时但是效果非常好的做法
2. 新建register.js，引用antd的form表单

import loadable from 'utils/loadable'//组件懒加载

import React, { useState, useEffect } from 'react';

import { withRouter } from 'react-router';

import { observer } from 'mobx-react-lite';

const BackgroundVideo = loadable(() => import('components/BackgroundVideo/BackgroundVideo'));

const NormalLoginForm = loadable(() => import('./components/NormalLoginForm/NormalLoginForm'));

export default observer(withRouter(function Login(props) {

    useEffect(() => {

    })

    return (

        <React.Fragment>

            <BackgroundVideo />

            <NormalLoginForm />

        </React.Fragment>

    )

}))

1. 发送请求到后端

export const register = (data) => {

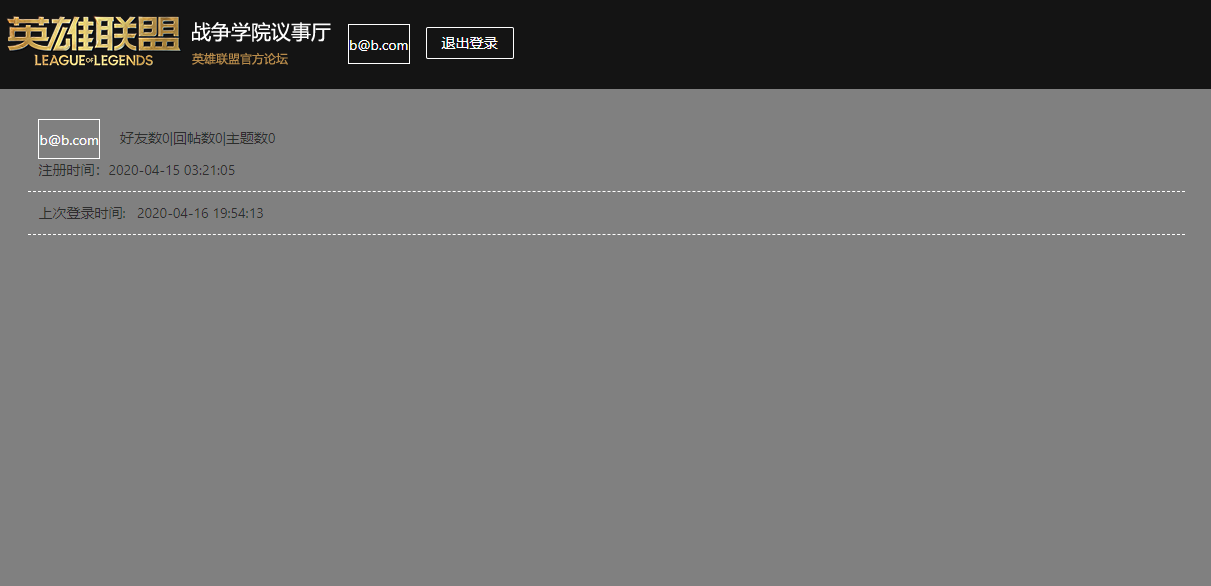
    return axios.post('/api/users/register', data);

}

1. 后端接受请求并查看邮箱是否已经存在于数据库中，不存在则创建用户，存在则返回错误信息
2. 注册成功

成功

5.2.4个人中心



个人中心主要展示用户的信息，其内容有用户名，好友数量，回帖数量，主题数量，用户的注册时间以及用户的上次登录时间。在设计登录时间上遇到一些小的问题，一开始我在思考这个功能应该从何做起，首先用户登录必然会伴随着一个时间点，于是在用户登录后我自动的调用api存入当前的时间点，等到下次用户登录的时候，我再从数据库中自动查询，于是便完成了了这一功能，通过这个过程，让我明白了，遇到问题要一步步分析，不要因为刚开始不知所措而放弃，要慢慢探究整个过程，逐步分析出其逻辑。

5.2.5热点轮播

在设计轮播的时候，进入轮播页面后如何返回这个问题困扰了很久，直接点击浏览器的返回按钮这种方式似乎显得不是那么优雅，有可能我们输入的一些内容没有及时保存下来，通过对比其他交流平台的做法，我选择了点击我们的logo回到主页看到热点轮播的这种做法，热点轮播较为清晰地描述了当下的实时热点，见下图。

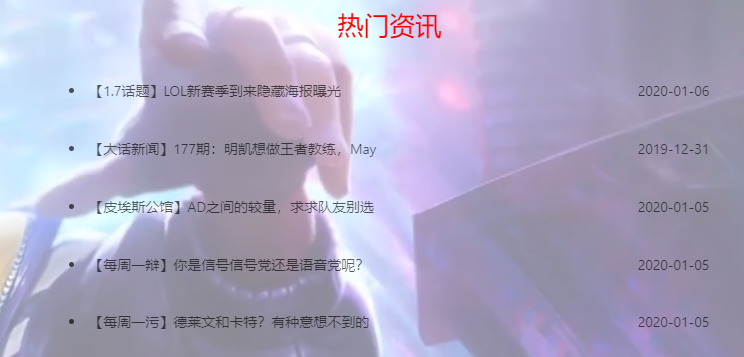


点击热点轮播进入详情，详情页主要是玩家们对有关主题的讨论，玩家们分享自己的所见所闻所感，并且可以配上相应的图片，更好的表达玩家遇到此事或者此时的心情。见下图



5.2.6热门咨询

由于轮播图所表现出的内容有限，考虑到对称布局，因此在右侧增加了热门资讯板块，用来较为多的展示当下的实时热点，实时热点自然需要有实时性，因此对应的每一行都加上了咨询的时间，点击咨询我们能看到资讯里面的详细内容，主要是以文章为主，能够让玩家们方便的获取游戏知识，提升游戏技巧，使得玩家获得更好的游戏体验。



第六章 总结

6.1后端总结

6.1.1后端技术总结

从后端的技术使用上来说，主要是使用了以express为基础的技术架构，由于nodejs整体是通过各种中间件来完成各种各样的功能，因此我们的使用上是非常的灵活的，我们可以通过设置各种各样的中间件来响应HTTP请求，除此之外，我们还可以通过定义路由表的方式，模块化，分层的执行不同的HTTP请求动作，当然express的能力不仅仅是如此，再加上express有着悠久的历史，完善的文档，丰富的资料，综上所述，express的选择是顺利完成这一切的基调，由于其以上的特点，因此在开发过程中比较顺利，再加上老师的一些指点密境，因此较快的完成了本次开发。

Express是奠基石，mongodb作为数据库自然是不可分割的一部分，对于数据的各种curd操作，都是依赖mongodb来完成的，例如用户名的存储，页面数据的存储，基本上所有有关post的请求都要与mongodb数据库有关。

提到post请求，必然与bodyparser密不可分，本次使用bodyparser是为了简化方便处理前端请求的数据，bodyparser能够对我们传入的请求体进行解析，bodyparser能够解析我们前端传过来的用户名，密码，帖子内容等等，除此之外，还能解析其它几种格式

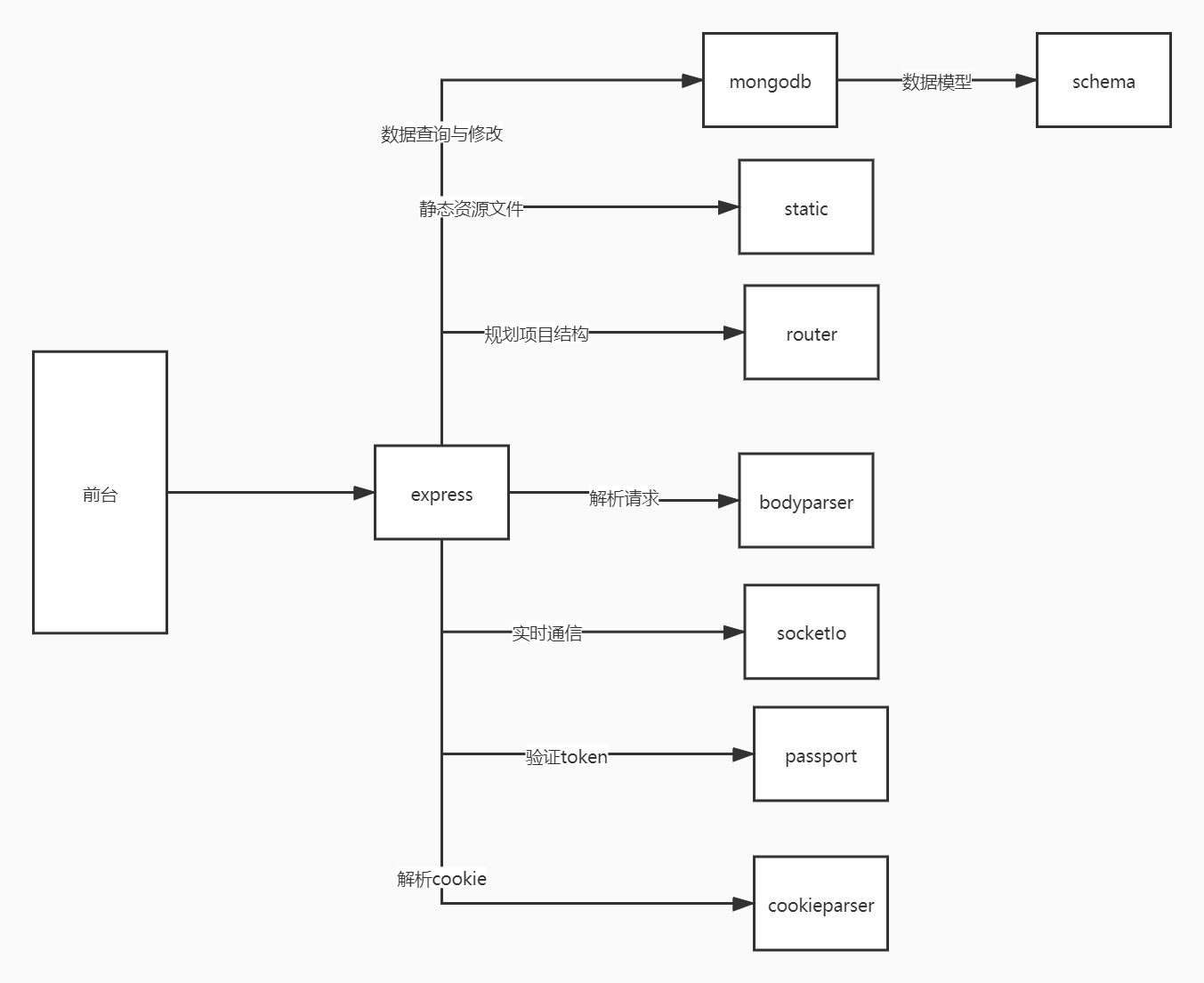
[JSON body parser](https://github.com/expressjs/body-parser" \l "bodyparserjsonoptions" \t "https://www.jianshu.com/p/_blank)  
[Raw body parser](https://github.com/expressjs/body-parser" \l "bodyparserrawoptions" \t "https://www.jianshu.com/p/_blank)  
[Text body parser](https://github.com/expressjs/body-parser" \l "bodyparsertextoptions" \t "https://www.jianshu.com/p/_blank)  
[URL-encoded form body parser](https://github.com/expressjs/body-parser" \l "bodyparserurlencodedoptions" \t "https://www.jianshu.com/p/_blank)

除此之外，在后端中，还加入了passport以增强我们前端后端交互的安全性，每一次的get或者post请求，都要携带我们的token，以防止被他人非法窃取用户资料。

以上就是后端主要用到的技术总结，通过这几项技术便实现了一个基本的能够前后端交互的程序。

6.1.2后端设计总结

后端主要设计了登录注册功能，以及热门帖子的内容存放等等功能，这依赖于下面的架构。

后端架构图是这样的

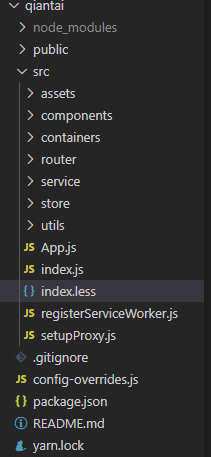
通过这张图我们可以清楚地看到，前端发送请求，通过bodyparser对url进行解析，然后开始验证前端请求的token是否正确，错误直接拒绝请求，成功则会查询数据库，然后通过express框架的操作传递数据给到前端，以上描述的是一个http请求的内容，当然我们对后端的管理还有路由的规划，这种架构设计主要是为了方便开发人员能够快速定位到需要修改的位置，完成指定功能。

6.2前端总结

6.2.1前端技术总结

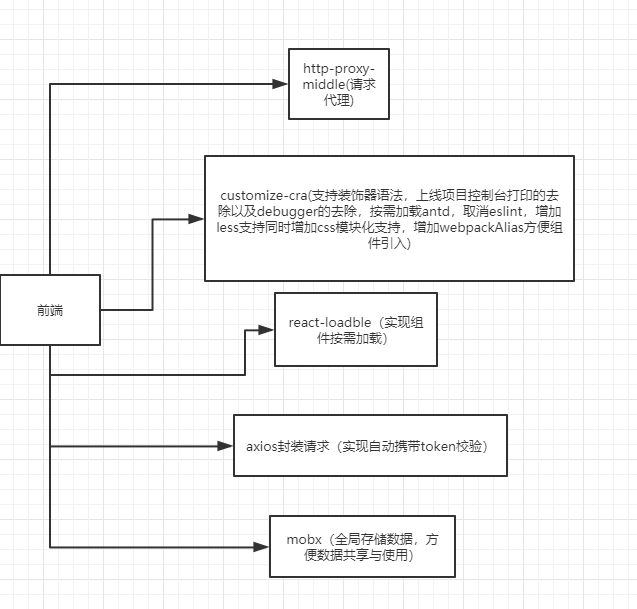
从前端的技术上来说，选择react的主要原因是对于react的掌握能力远远大于其它框架，并且react对比与其它框架有着非常不错的优势，前端依赖http-proxy-middleware

实现了请求代理，通过customize-cra支持了装饰器语法，按需引入antd，取消掉正式上线的打印以及debugger功能，取消掉了eslint功能，eslint会经常提示格式问题，虽然优化了代码结构，但是在效率上大大折扣，并且取消eslint并不会让程序崩溃，因此暂时没有选择使用eslint，另外增加了css的模块化，防止格式之间出现冲突，并且使用了webpackAlias，可以方便的引入各个文件下的不同内容，提升了开发效率，另外对axios进行了简单的封装，可以在每次请求之前在请求头中传入我们的token，避免了每次手动输入的麻烦。除此之外还增加了按需加载，防止项目越来越大首屏加载过慢，其次对我们的目录结构进行了调整，如下图



6.2.2前端设计总结

前端ui上主要有登录注册，个人信息查看，轮播热门，热点资讯，详情等等。这主要依赖于以下的技术框架设计



通过http-proxy-middleware，可以将本地的所有请求统一代理到5000端口，由于后端的端口便是5000，因此实现了前后端的连同。通过customize-cra,即webpack里面的功能，使得项目支持了装饰器语法，不仅仅mobx需要装饰器的语法，其它路由等等也可以通过这种方式方便的注入其中，随取随用非常方便。通过react-loadble，每一个组件，例如前端的轮播图等等，都是在需要展示的时候再次发起http请求，防止由于所有组件都挂载在首页上，首页内容越来越庞大而加载缓慢，通过对axios的封装，每次请求都可以自动携带请求头token消息，因此不必每次都关心token问题，mobx是用来存储页面上的所有的数据，能够方便的在各个组件之间共享这些数据，便利了模块化开发。

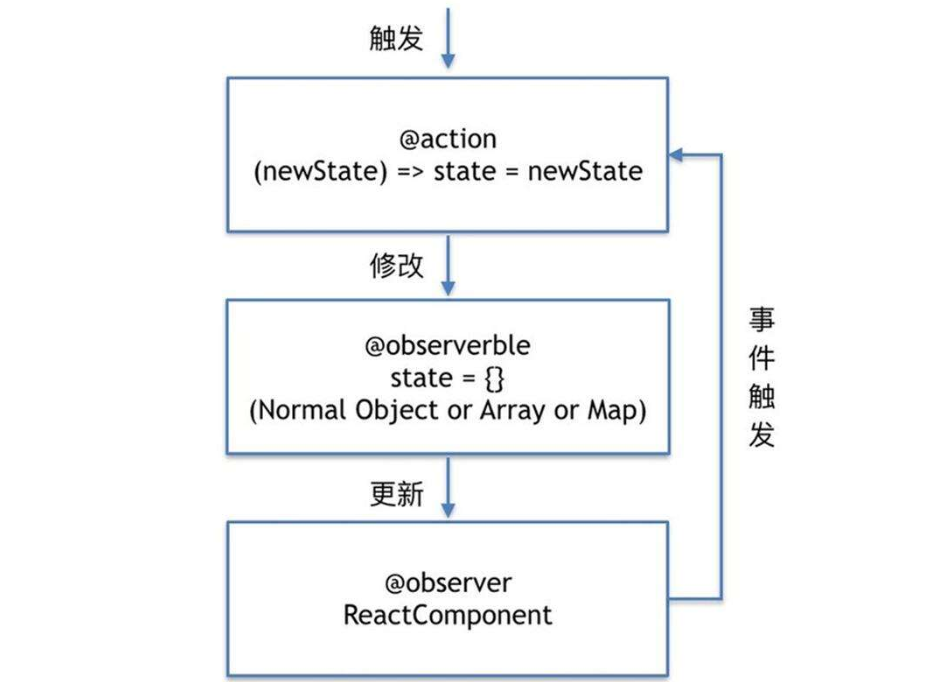
参考文献

[1] <https://cn.mobx.js.org/> mobx官方中文文档.

[2]https://www.baidu.com/link?url=Fh5y1h00OYUkk-X6NjYFBywduiKyBZssrpL1Q\_IJGw\_kc-SgIrCWh9GmLBkYp5Fo&wd=&eqid=8d54df07000a7dc4000000035e5ba15b mongodb官方文档.

[3]http://nodejs.cn/ nodejs官方中文文档

附 录



致 谢

历史两个月终于完成了这篇论文，在这个充满艰辛的过程中，从过去的一脸懵逼什么是论文到逐渐意识到论文是一样什么样的东西，再到想尽各种办法查阅资料去翻看应该怎么写论文，论文有哪些内容，需要注意哪些格式，在此过程中，前两次由于没有经验都出现了重大错误，章节不够，只是单纯的写了一些教程，不具有论文的性质，没有体现出自己的设计方案，而且格式也不太对，大部分时间在说各种工具与概念，这偏离了我的主题，应当写我在设计系统上的方案，架构，或者是遇到的问题，我是怎样解决这些问题的，通过对这些内容的仔细对比，最终经历多个版本的修改，终于完成，在此感谢本论文所引用的各位学者的著作，也要感谢国内良好的社区环境提供了大量的资料支持，同时感谢指导老师的谆谆教诲，督促我及时高效的完成任务，由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师和同学批评指正，感谢。