# 基于H5的美食分享类移动web应用

摘要：

关键词：

## H5-based gourmet sharing mobile web application

**Abstract:**

**Keywords:**

## 绪论

## 项目背景和意义：

“王者以民为天，而民以食为天”，吃对于每个人来说都是一件非常重要的事情，也是美好生活的重要组成部分。中国饮食文化博大精深，各色美食无不胜数，然而我们却时常为吃什么，美食怎么制作而感到困惑。

近几年，移动互联网高速发展，各类移动端应用层出不穷。android、ios作为原生应用，依托操作系统占据应用市场大部分份额，而web app凭借操作系统无关性、无需安装、高度可迭代等特点开始蓬勃发展。浏览器的性能不断提高给予了web应用更可靠的运行环境；类如angular.js 、react.js、vue.js等前端开发框架的提出简化了web应用的开发流程，提高了开发效率；HTML5规范不断发展，PWA(渐进式web应用)的提出更是让web应用有了与原生应用非常相似的操作体验。

结合美食的主题与web应用的独特优点，打造一款美食分享类web应用将对传播饮食文化，提高饮食品质具有深刻意义。

## 需求分析：

总体需求：

美食的主题与web应用的独特优点，设计并实现一款移动端的美食分享类应用。

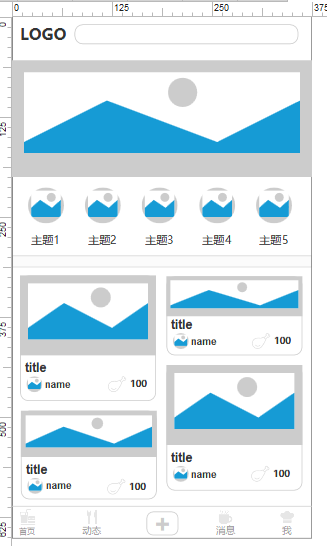
详细需求：

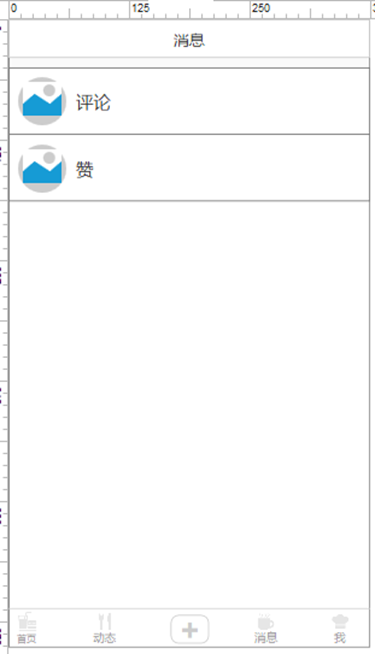
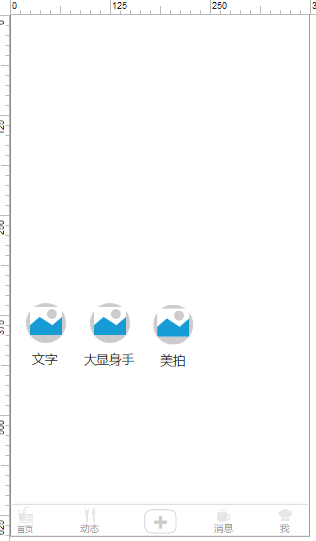
|  |  |  |
| --- | --- | --- |
| **功能模块** | **详细** | **备注** |
| 用户注册 | 提供用户名密码字段，并在前端确认密码以及数据完整性 |  |
| 用户登录 | 提供用户名和密码，在前端确认数据完整性 |  |
| 内容搜索 | 首页提供搜索框，用于针对用户名和动态内容的模糊搜索 |  |
| 首页内容动态推送 | 由后台完成内容的编辑 |  |
| 首页优质内容的推荐 | 根据点赞数及评论数量进行优质内容的推送 |  |
| 动态内容的获取 | 根据时间获取最新动态，依据时间进行排序 |  |
| 动态的发布 | 提供文字图片的发布 |  |
| 用户间的相互关注 | 用户间可进行相互关注及取消关注 |  |

## 概要设计：

### 1交互原型图设计

本次毕业设计使用axure rp进行快速原型设计。axure rp作为一款专业的快速原型设计工具，能够快速高效地创建项目原型图，使整个项目交互流程更加明朗，模块之间的关系更加清晰，更加清楚地展示项目需求。此次项目原型设计如下图：







### 2项目架构

本次毕业设计的定位为移动端web应用的设计与实现，因此主要注重项目在移动端的运行与UI风格。

整个项目采用前后端分离的基础架构方案，前端通过ajax程序向后端发送网络请求，后端提供RESTful风格的API接口作为前端请求的响应接口。

前端采用Vue.js+ Vuex + VueRouter + Axios的技术栈。其中vue作为一款轻便并且强大的前端应用开发框架，其具有社区活跃度高，工具完善，文档完善等特点，是现代前端开发框架的不二选择；Vuex 是一款基于Vue的状态管理器，它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化；VueRouter是Vue官方提供的路由管理器，它和 Vue.js 的核心深度集成，让构建单页面应用变得易如反掌。Axios是Vue官方推荐的http请求库，同时支持浏览器和node端的使用。

后端采用 node + express + mongodb + mongoose的技术选型。其中node作为javascript服务端的运行环境，它具有事件驱动，支持高并发和非阻塞IO等特点；express是一款基于 Node.js 平台的极简、灵活的 web 应用开发框架，它提供一整套清晰的服务端开发模式，并且对于RESTful API的支持非常友好；mongodb是一款文档型数据库，作为一款非关系型数据库，支持数据的横向扩展；mongoose提供对数据库清晰的模型定义，提供强大的数据库操作接口，使数据坤查询更加方便。

整个项目前后端皆以javascript作为主要开发语言，因此项目开发阶段node为主要运行环境为主的开发运行环境，npm 作为node的官方包管理器提供应用包管理服务。

项目架构图如下：

HTTP

pm2

Axios

expresss

VueRouter

Vuex

mongoose

mongodb

node

Vue

### 3模块划分

为了前后端之间交互清晰，项目便于管理，前后端采用一致的模块划分。均分为：

1. 用户模块——支持用户的注册、登录认证、获取以及修改用户信息、用户之间的关注功能
2. 动态模块——用于用户编辑和获取动态信息，添加评论和点赞等操作
3. 主题模块——用于首页推荐主题内容的操作
4. 搜索模块——负责全网用户以及动态内容的模糊搜索功能
5. 上传模块——负责静态文件的上传的处理功能

### 4前后端项目目录结构划分

合理的目录机构划分将有利于模块之间的划分以及项目的开发迭代。本次设计采用前后端完全分离的目录结构划分，响应各自项目有自己独立的目录结构。

###### 前端项目主要目录结构

├─assets

├─components

├─constants

├─factories

├─router

├─store

├─utils

└─views

1. assets: 用于存放前端所需的静态资源文件，例如css文件、js文件、图片资源等
2. components: 负责项目所用的一些通用组件
3. constants: 负责配置一些全局静态变量
4. factories: 负责前端所需的http请求操作
5. router: 负责应用的路由操作
6. store: 全局项目数据管理
7. utils: 相对复杂的数据处理
8. views: 项目主要视图页面

###### 后端项目主要目录结构

├─common

├─configs

├─controllers

├─middlewares

├─models

├─services

├─test

│ └─controllers

└─uploads

└─pictures

1. commons: 负责存放项目所用到的通用工具
2. configs: 负责配置全局静态变量
3. controllers: 后端处理请求的控制器
4. middlewares: 存放中间件程序
5. models: 负责定义数据模型的程序
6. services: 负责较为复杂的数据库操作及逻辑处理
7. test: 测试文件

### 5接口设计

搜索操作

* URL: /search
* Method: POST

获取搜索高频关键字

* URL: /search
* Method: GET

获取推荐内容

* URL: /recommendation/content
* Method: GET

获取推荐动态

* URL: /recommendation/dynamics
* Method: GET

获取动态列表

* URL: /dynamics
* Method: GET

发布动态

* URL: /dynamic
* Method: POST

获取动态详情

* URL: /dynamics/:article\_id
* Method: GET

修改动态内容

* URL: /dynamics/:article\_id
* Method: PUT

删除动态

* URL: /dynamics/:article\_id
* Method: DELETE

点赞操作

* URL: /dynamics/:article\_id/approve
* Method: PUT

获取评论

* URL: /dynamics/:article\_id/comment
* Method: GET

添加评论

* URL: /dynamics/:article\_id/comment
* Method: POST

删除评论

* URL: /dynamics/:article\_id/comment
* Method: DELETE

获取个人动态列表

* URL: /dynamics/user/:user\_id
* Method: DELETE

获取粉丝列表

* URL: /fans/:user\_id
* Method: GET

获取关注列表

* URL: /follow/:user\_id
* Method: GET

添加、取消关注

* URL: /follow/:user\_id
* Method: PUT

## 详细设计：

### 1应用UI设计

### 

### 1数据库模型设计

User 用户模型

{

\_id: String,

username: String,

password: String,

picture\_url: String,

phone\_number: String

introduction: String,

fans: Array

follow: Array

}

Article 文章模型

{

\_id: String,

type: Number,

title: String,

title\_img: String,

content: String,

create\_at: Date,

update\_at: Date,

comment\_count: Number,

approve\_count: Number,

tag: Array

}

Comment 评论模型

{

username: String,

content: String,

article: { type: ObjectId, ref: ‘Article’ },

create\_at: Date,

root\_id: { type: ObjectId, ref: ‘Comment’ },

reply\_to: { type: ObjectId, ref: ‘Comment’ },

replies: { type: ObjectId, ref: ‘Comment’ }

}

Tag 标签模型

{

\_id: String

name: String,

create\_at: Date

}

Recommendation 推荐内容模型

{

\_id: String,

img: String,

title: Sting,

page\_url: String

}

KeyWord 搜索关键字模型

{

content: String,

search\_count: Number

}

FeedBack 用户反馈模型

{

username: String,

user\_id: String,

content: String,

browser\_type: String

}

### 2模块详细设计

**用户模块：**

用户认证：用户登录时，程序根据用户名及密钥生成一端长字符串返回给客户端，用户执行需要特殊权限的操作时需在请求头Authorization字段上带上这一段字符串。

认证代码如下：

auth.verify = (req, res, next) => {

const authHeader = req.header('Authorization');

const auth = req.auth = {isAuth: false, message: '用户未登录'};

if(!authHeader){

return next();

}

const token = authHeader.split(' ')[1];

if(authHeader.split(' ')[0] != 'Bearer' || !token){

auth.message = '验证信息格式错误';

return next();

}

//校验token

jwt.verify(token).then(obj => {

//token已校验

auth.isAuth = true;

delete auth.message;

if(auth.isMember = obj.user\_type === 'member'){

auth.username = obj.username;

auth.\_id = obj.id;

}

next();

}).catch(err => {

Object.assign(auth, err);

return next();

});

}

用户是否已被关注处理：在前端获取用户信息时，需要展示是否该用户课被关注，如果已被关注就可以执行取消关注操作，否则可被关注，如果该用户是当前用户本身则不可进行任何操作。服务端通过service层对用户模型的isFollow字段进行响应修改，再将处理后的数据交给控制层用于返回给前端。处理程序如下：

userService.addIsFollow = (userId, users) => {

return User.findUserById(userId).then(user => {

const follows = user.follow;

for (let i = 0; i < users.length; i++) {

if (users[i].\_id.toString() === userId) {

users[i].is\_follow = 2;

}

for (let j = 0; j < follows.length; j++) {

if (users[i].\_id.toString() === follows[j].\_id.toString()) {

users[i].is\_follow = 1;

}

}

}

return users;

})

}

**图片上传模块**

图片上传处理：上传的服务端的图片文件需要唯一的文件名以便在加载文件是没有二义性，上传的图片文件通过哈希算法生成一个唯一的文件名重新赋给该图片，再通过io处理将图片存储在服务端静态文件系统中。其中生成唯一文件名的方法如下：

function getNewNameByHash() {

let string = Date.now().toString().substring(5, 13);

return crypto.createHash('md5').update(string).digest('hex');

}

**模糊搜索模块**

用户名和动态内容的模糊查询：应用首页具有搜索功能设置，用户可通过关键字对用户或者全网的动态内容进行搜索查询。服务端通过dynamics集合和user集合的联动查询的方式进行搜索。其控制层的联动查询代码如下：

searchController.search = (req, res, next) => {

const query = req.body.query;

const userId = req.body.userId;

const obj = {};

User.fuzzyQueryByUserName(query).then(users => {

return userService.addIsFollow(userId, users);

}).then(users => {

obj.users = users;

return Dynamics.fuzzyQuerByTitleAndContent(query);

}).then(dynamics => {

obj.dynamics = dynamics;

res.success(obj);

}).catch(next);

}

## 编码开发：

### 1环境准备

编码环境为windows 10 + Visual Studio + MINGW, 其中MINGW是集成了git与shell运行环境的命令行工具。

另需工具

"node": ">= 6.0.0",

"npm": ">= 3.0.0",

"mongodb": ">= 3.2.0",

此次开发的版本控制工具选择git，远程代码仓库选择github。其中git是一款开源的分布式版本控制工具，它可以刻将你的代码恢复到任一提交的时刻。github是一个免费的代码托管平台，它可以清楚地看到通过git每一次代码提交的记录。

## 项目测试：

**1测试框架**

**2测试模块**

## 软件交付：

**1运行环境介绍**

**2运行方法**