**《**Todo系统Node **》说明文档**

1. **学号：**
2. **姓名：**

#### 技术框架 Vue+Express+Mysql

Vue前端部分:使用Vue + VantUi +ElementUi

在测试期间调用了Mock，实现前端自检查

Mock的单元测试

// mock.js

// 引入mockjs

const Mock = require('mockjs');

// 获取 mock.Random 对象

const Random = Mock.Random;

const getPersonTodo = () => {

let todoList = [];

for (let i = 0; i < 100; i++) {

let todoObject = {

id: i + 1,

name: Random.cname(),

tel: Random.natural(10000),

task: Random.csentence(3, 5),

isFinish: Random.boolean(),

}

todoList.push(todoObject)

}

return todoList;

}

const getImg = (val) => {

let imgList = [];

for (let i = 0; i < 5; i++) {

imgList.push(Random.image('200x100', '#4A7BF7', 'Todo'))

}

return imgList

}

// Mock.mock('/api', getPersonTodo)

Mock.mock('/api/img', getImg)

用户之间的交互功能在前端列出，共计15个请求，其余是数据表之间的关系，数据表有8个左右

文件夹 PATH 列表

文件夹 PATH 列表

卷序列号为 C8E0-B843

C:.

├─build

├─config

├─dist

│ └─static

│ ├─css

│ ├─fonts

│ └─js

├─node\_modules

├─src

│ ├─assets

│ │ └─css

│ ├─components

│ ├─mock

│ ├─pages

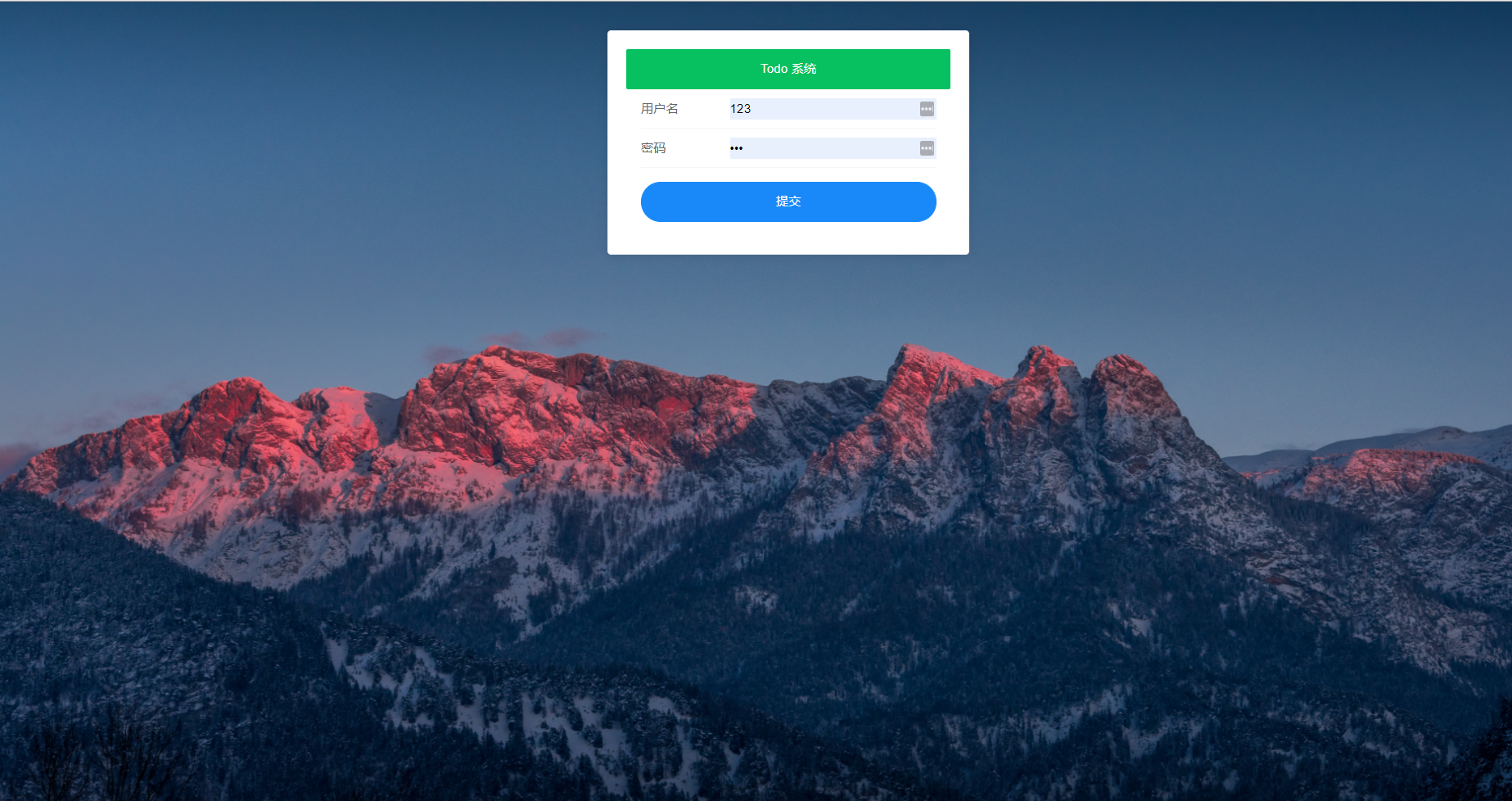
│ ├─router

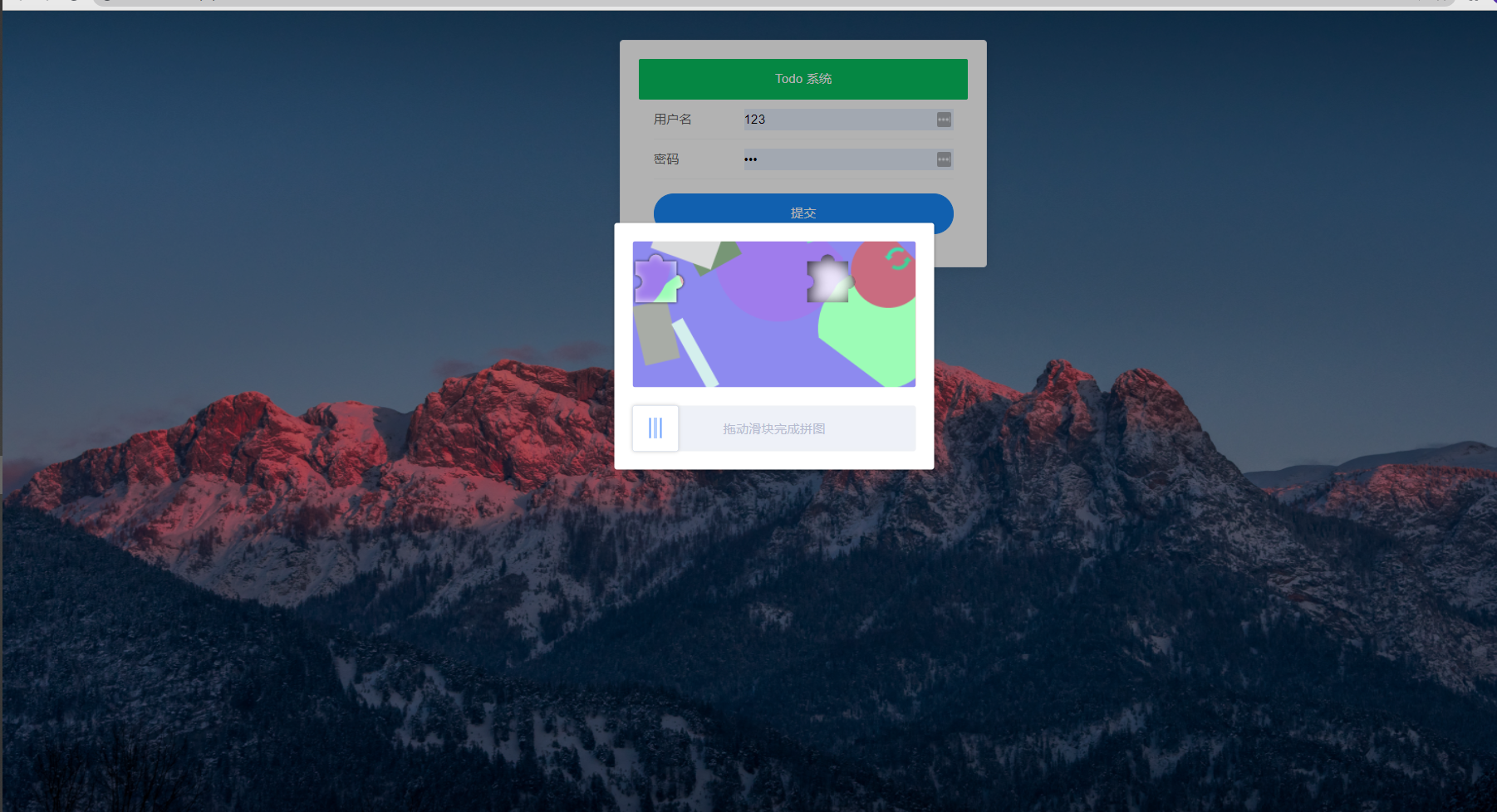
│ └─utils

└─static

### 界面功能：

登陆界面





关键代码

<template>

<div class="login-container-main">

<img :src="src" class="bg-login" />

<el-card class="box-card" >

<login-form @submit="handleToSubmit"/>

</el-card>

<Vcode

:show="active"

@success="handleConfirmSubmit(form)"

/>

</div>

</template>

<script>

import loginForm from "@/components/loginForm";

import Vcode from "vue-puzzle-vcode";

export default {

data() {

return {

active: false,

form:null,

src: 'https://cube.elemecdn.com/6/94/4d3ea53c084bad6931a56d5158a48jpeg.jpeg'

};

},

components: {

loginForm,

Vcode

},

methods: {

async handleToSubmit(form) {

this.active = true;

this.form = form;

// Notify({

// });

// await this.$router.push({path: "/main"});

},

async handleConfirmSubmit(form){

let params = {

user\_account: form.username,

user\_password: form.password

}

let res = await this.$http.postLogin(params)

console.log(res)

this.active = false

if (!res.token) {

return this.$message({

message: res.msg,

type: "danger"

})

}

this.$message({

message: res.msg,

type: "success"

})

localStorage.setItem("token", res.token);

await this.$router.push({path: "/main"})

}

},

};

</script>

<style scoped>

.login-container-main{

overflow: hidden;width: 100vw;

height: 100vh;

}

.box-card{

width: 400px;

position: absolute;

top: 2em;

left: 40%;

}

.bg-login{

width: 100vw;

height: 100vh;

overflow: hidden;

}

</style>

<template>

<div>

<swipper />

<todo-list :list="list" />

</div>

</template>

<script>

import todoList from './todoList';

import swipper from './swipper'

export default {

components: {

todoList,swipper

},

data() {

return {

chosenAddressId: '1',

list: [],

};

},

methods: {

async handleGetList() {

let token = localStorage.getItem("token");

let res = await this.$http.getList({token})

console.log(res)

this.list = res.list

}

},

mounted() {

this.handleGetList()

}

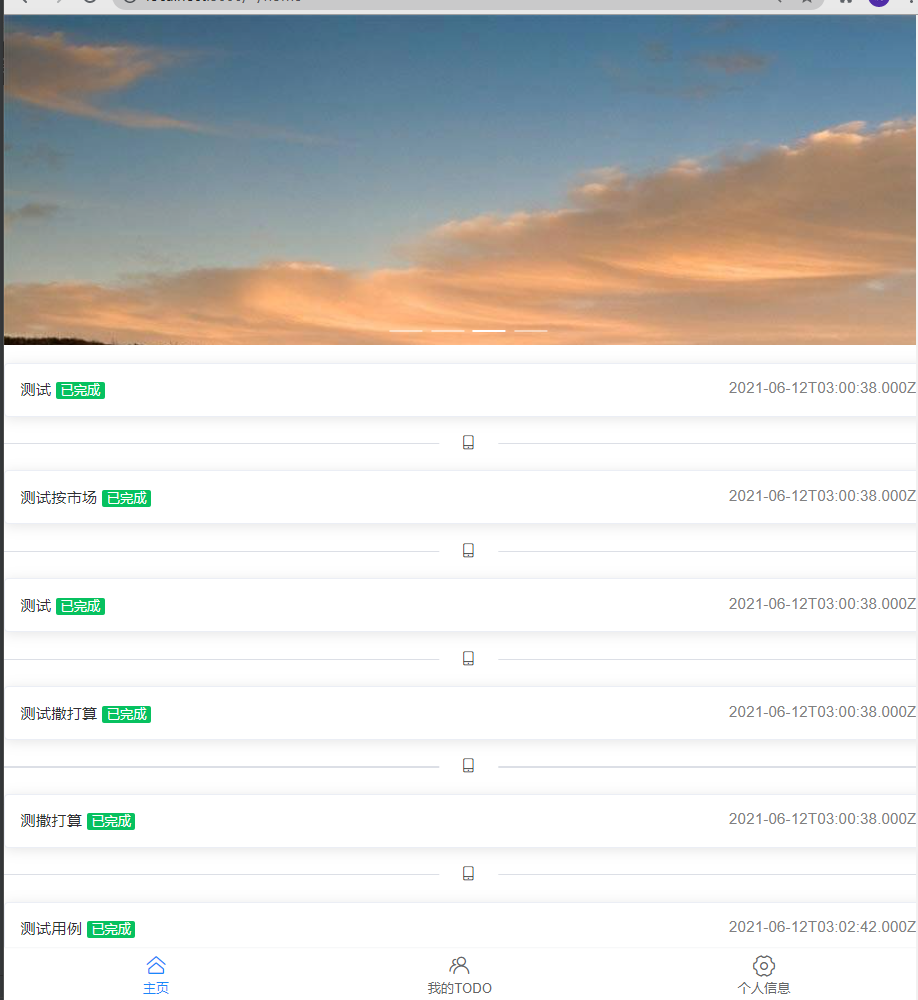
};

</script>

<style scoped>

</style>

### 所有人的分享详情



关键代码

<template>

<div>

<!-- <van-sticky>-->

<!-- <van-button type="primary" style="width: 100%;">Todo 首頁</van-button>-->

<!-- </van-sticky>-->

<ul class="todo-container">

<li v-for="(item, index) in list" :key="item + index">

<el-card :body-style="{ padding: '14px' }" style="margin-top: 1em">

<span style="font-size: 14px">{{ item.todo\_content }}</span>

<van-tag v-if="item.todo\_status" type="success">已完成</van-tag>

<el-tag size="mini" plain v-else type="warning">未完成</el-tag>

<!-- <el-badge :value="200" :max="99" class="item" style="float: right">-->

<!-- <el-button type="info" size="mini" plain> </el-button>-->

<!-- </el-badge>-->

<span style="font-size: 14px;color: gray;float: right">{{ item.todo\_start\_time }}</span>

</el-card>

<el-divider><i class="el-icon-mobile-phone"/></el-divider>

</li>

</ul>

</div>

</template>

<script>

export default {

props: {

list: {

default() {

return [];

},

},

},

};

</script>

<style scoped>

</style>

### 个人TODO界面

### 

此处采用获取所有的海报和Id，以及商户信息

<template>

<div>

<el-divider><i class="el-icon-mobile-phone"/>我的TODO</el-divider>

<keep-alive>

<ul class="todo-container">

<li v-for="(item, index) in list" :key="item + index">

<el-card class="box-card">

<el-checkbox

v-model="item.todo\_status"

@change="handleSetTodoStatus(item.todo\_status,item.todo\_id)"

/>

<span style="font-size: 14px">{{ item.todo\_content }}</span>

<van-tag v-if="item.todo\_status" type="success">已完成</van-tag>

<van-tag v-else> 未完成</van-tag>

<el-button type="success"

plain

size="small"

style="float: right"

@click="handleAddTodo"

v-show="index===list.length-1"

>新增

</el-button>

<el-button type="success"

style="float: right"

size="small"

plain

@click.prevent="handleSharedTodo(item.todo\_id)"

v-show="!item.todo\_shared\_status"

>分享

</el-button>

<el-button type="danger"

style="float: right"

size="small"

plain

@click.prevent="handleDelShare(item.todo\_id)"

v-show="item.todo\_shared\_status"

>取消分享

</el-button>

</el-card>

</li>

</ul>

</keep-alive>

<el-dialog

title="请输入"

:visible.sync="show"

>

<el-card class="box-card">

<el-form ref="form" label-width="80px">

<el-form-item label="TODO">

<el-input v-model="value" placeholder="请填入Todo"/>

</el-form-item>

</el-form>

</el-card>

<span slot="footer" class="dialog-footer">

<el-button @click="show = false">取 消</el-button>

<el-button type="primary" @click="handleFinishAdd(value)">确 定</el-button>

</span>

</el-dialog>

</div>

</template>

<script>

export default {

// props: {

// list: {

// default() {

// return [];

// },

// }

// },

data() {

return {

active: 0,

show: false,

value: null,

list: []

};

},

mounted() {

this.handleGetMyTodo()

},

methods: {

handleAddTodo() {

this.show = true

},

async handleGetMyTodo() {

const token = localStorage.getItem("token")

let res = await this.$http.getTodoList({token});

console.log(res)

res.list.forEach(val=>val.todo\_status=!!val.todo\_status)

this.list = res.list

},

async handleFinishAdd(value) {

if (!value) return this.$alert("请输入")

const token = localStorage.getItem("token");

let res = await this.$http.postTodo({todo\_content: value, token});

// console.log(res)

this.$router.go(0)

},

async handleSharedTodo(todo\_id) {

console.log(todo\_id)

const token = localStorage.getItem("token");

let res = await this.$http.sharedMyTodo({todo\_id, token})

console.log(res)

this.$router.go(0)

}, async handleDelShare(todo\_id) {

console.log(todo\_id)

const token = localStorage.getItem("token");

let res = await this.$http.delMyShare({todo\_id, token})

this.$router.go(0)

console.log(res)

},

async handleSetTodoStatus(value, todo\_id) {

const token = localStorage.getItem("token");

if (value) {

console.log({token, todo\_id})

return this.$http.start({token, todo\_id})

}

return this.$http.end({token, todo\_id})

}

},

};

</script>

<style>

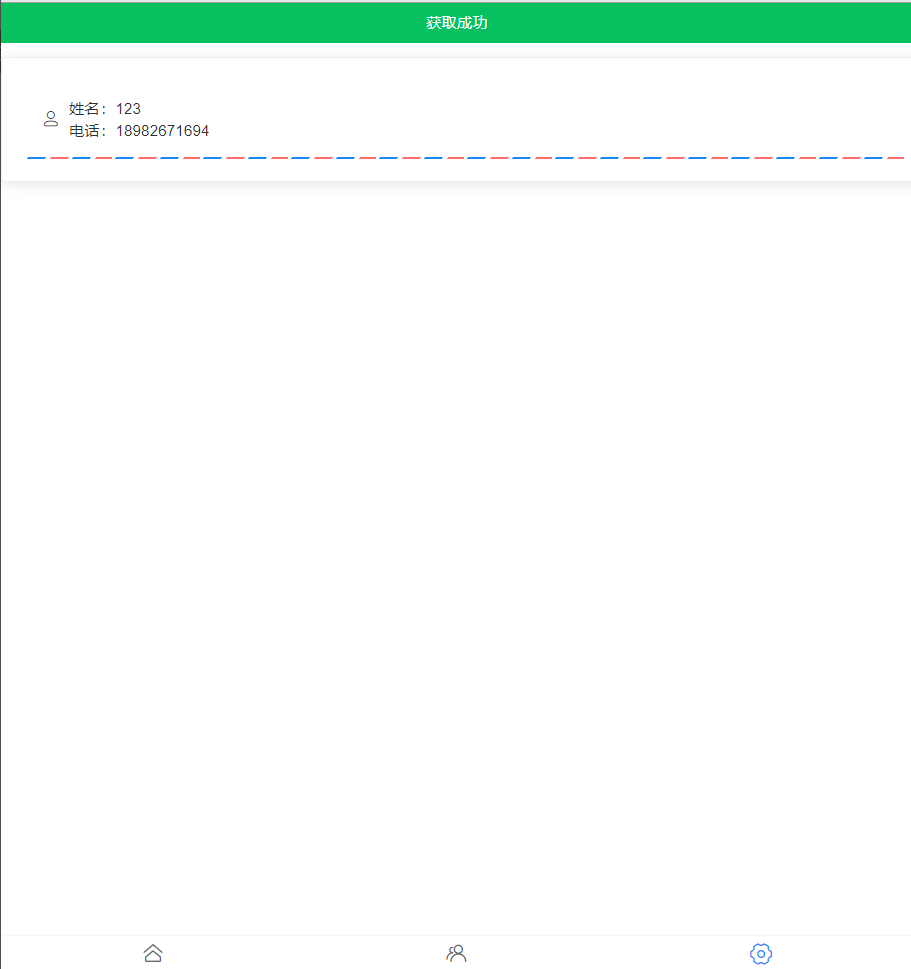
.van-checkbox\_\_label {

display: block;

width: 100%;

}

</style>



### 关键代码

<template>

<div>

<el-divider><i class="el-icon-mobile-phone"/>我的信息</el-divider>

<el-card class="box-card">

<van-contact-card

type="edit"

:tel="myInfo.user\_mobile"

:name="myInfo.username"

:editable="false"/>

</el-card>

</div>

</template>

<script>

export default {

data() {

return {

active: 0,

myInfo: {

user\_id: 1,

username: "张三",

user\_mobile: "13000000000",

}

}

},

async mounted() {

const token = localStorage.getItem("token")

let res = await this.$http.getUserInfo({token})

console.log(res)

this.myInfo = res

localStorage.setItem("userInfo",res)

}

};

</script>

<style scoped>

.todo-container li {

border: solid 1px;

background-color: beige;

margin-top: 1em;

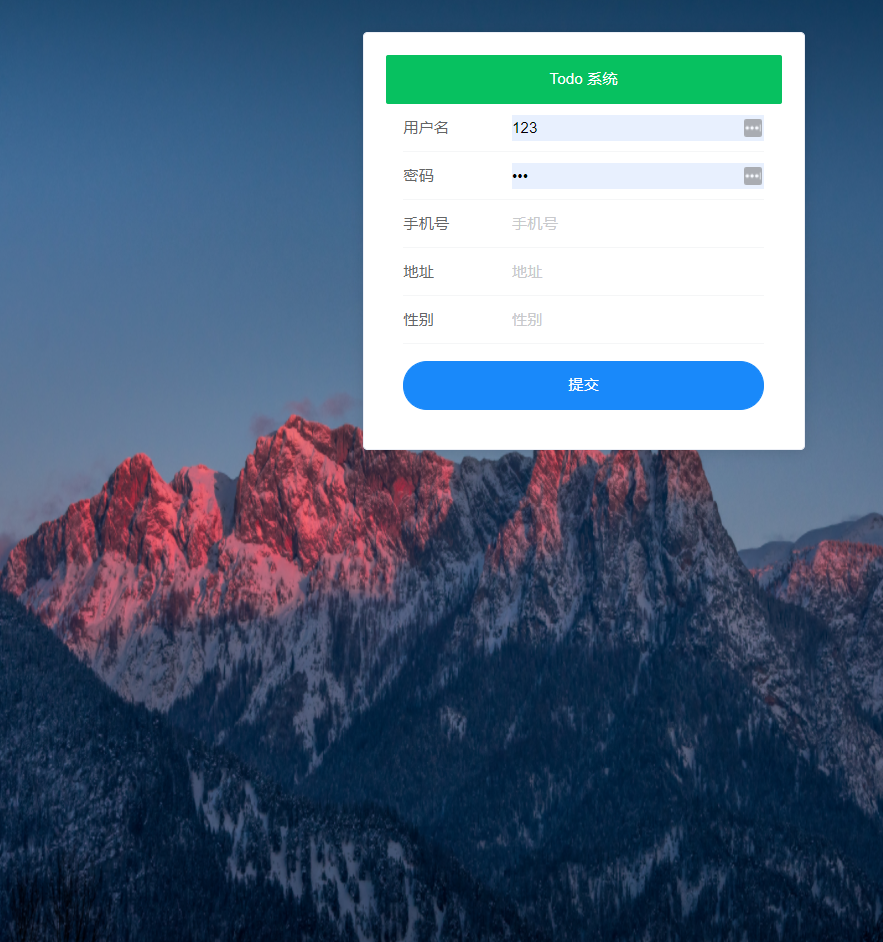
border-radius: 2em;

padding: 1em;

}

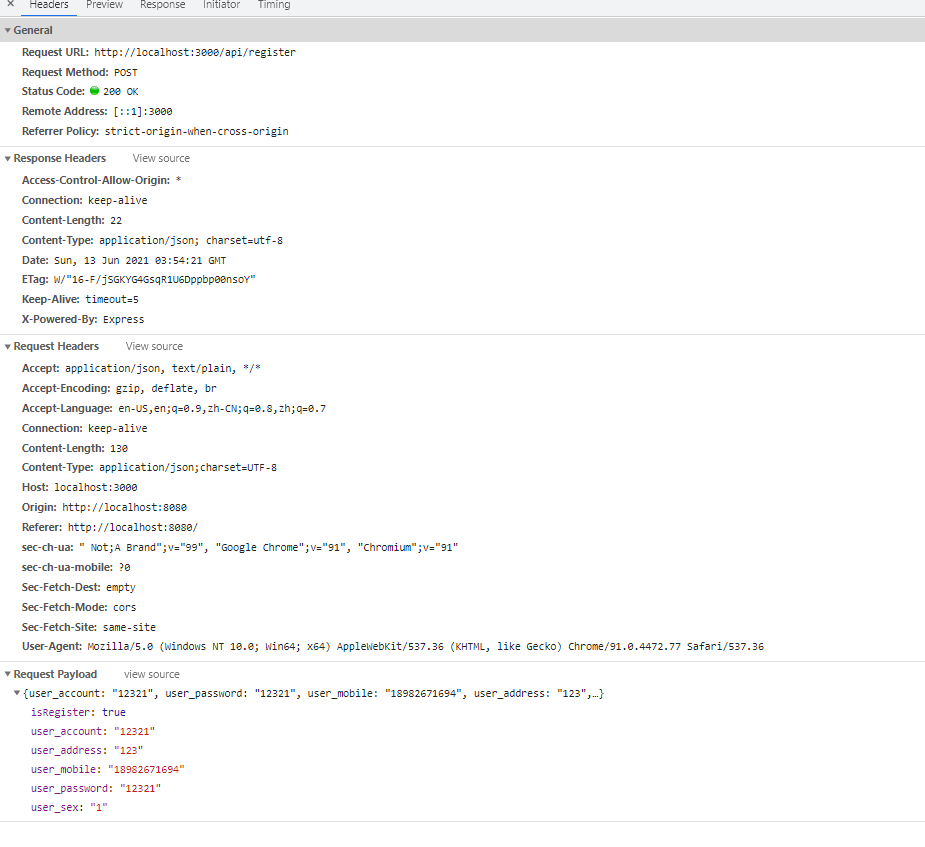
</style>

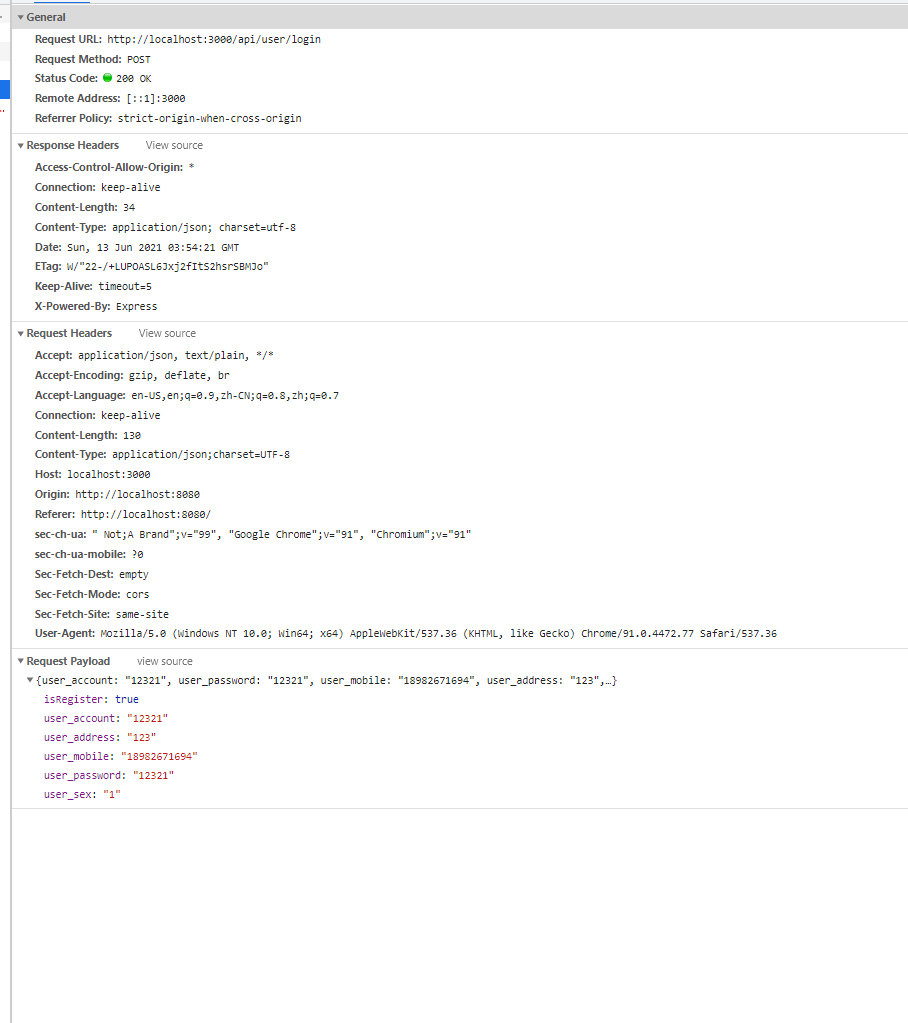
#### 注册界面

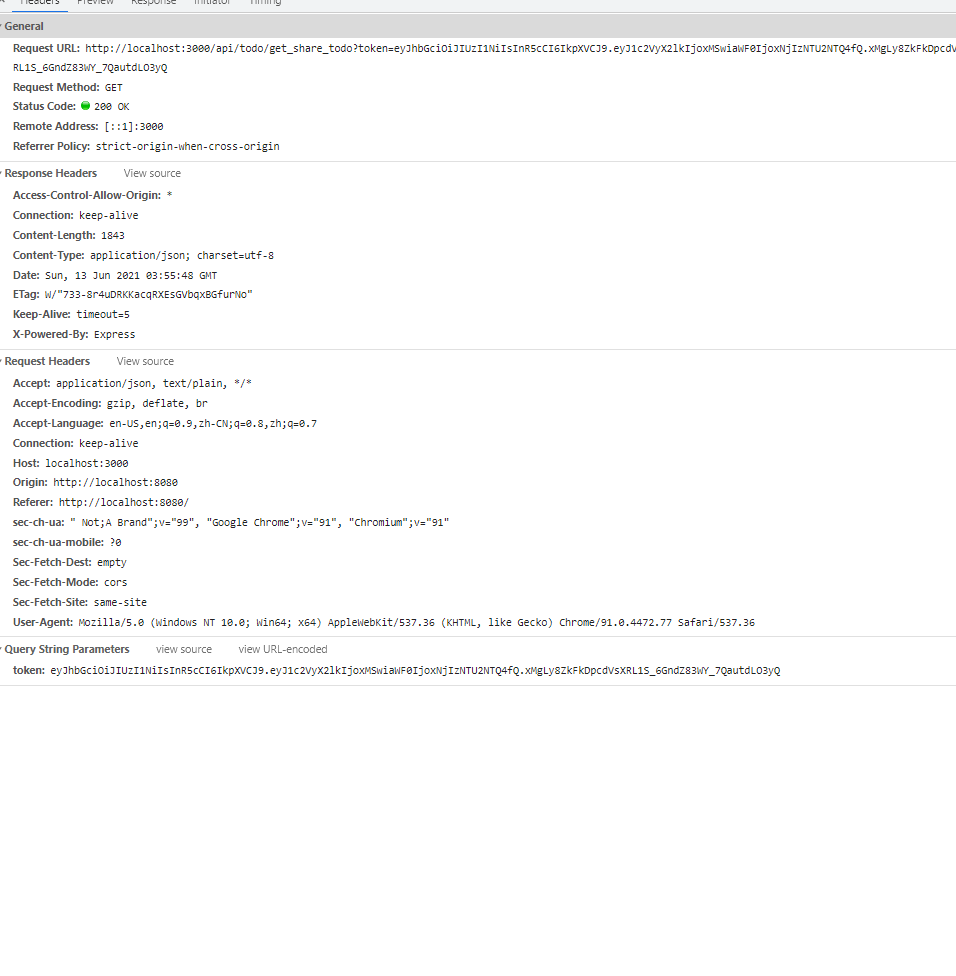


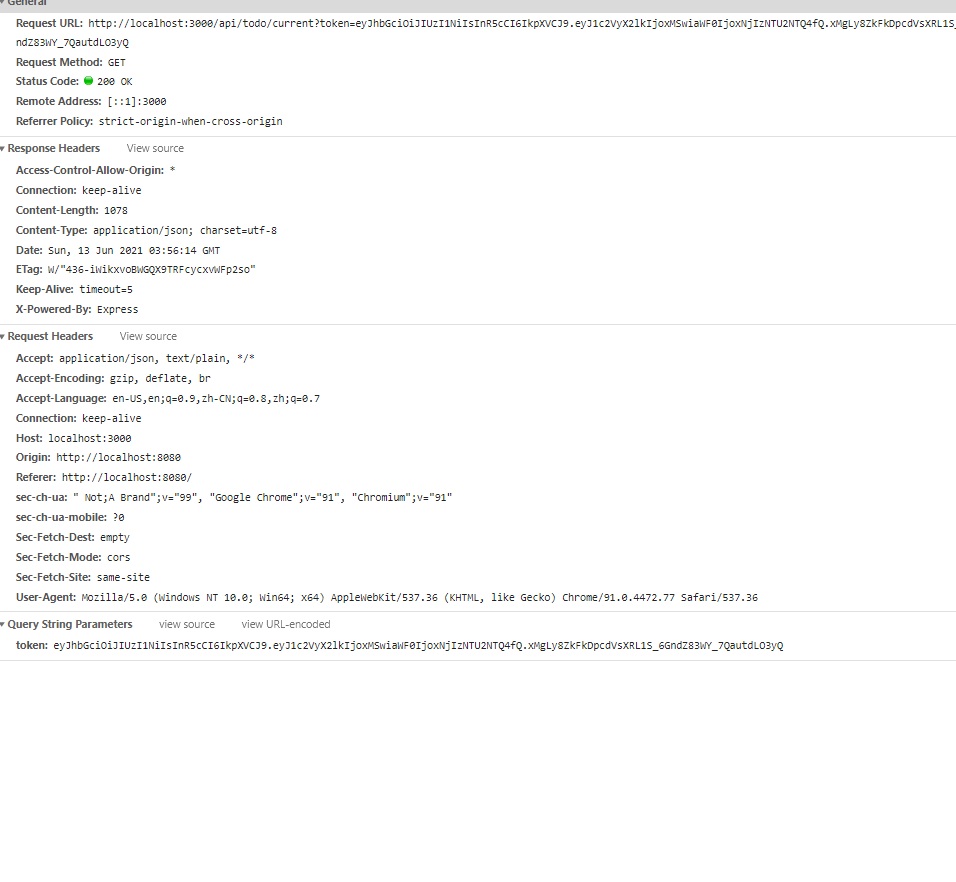
关键代码：

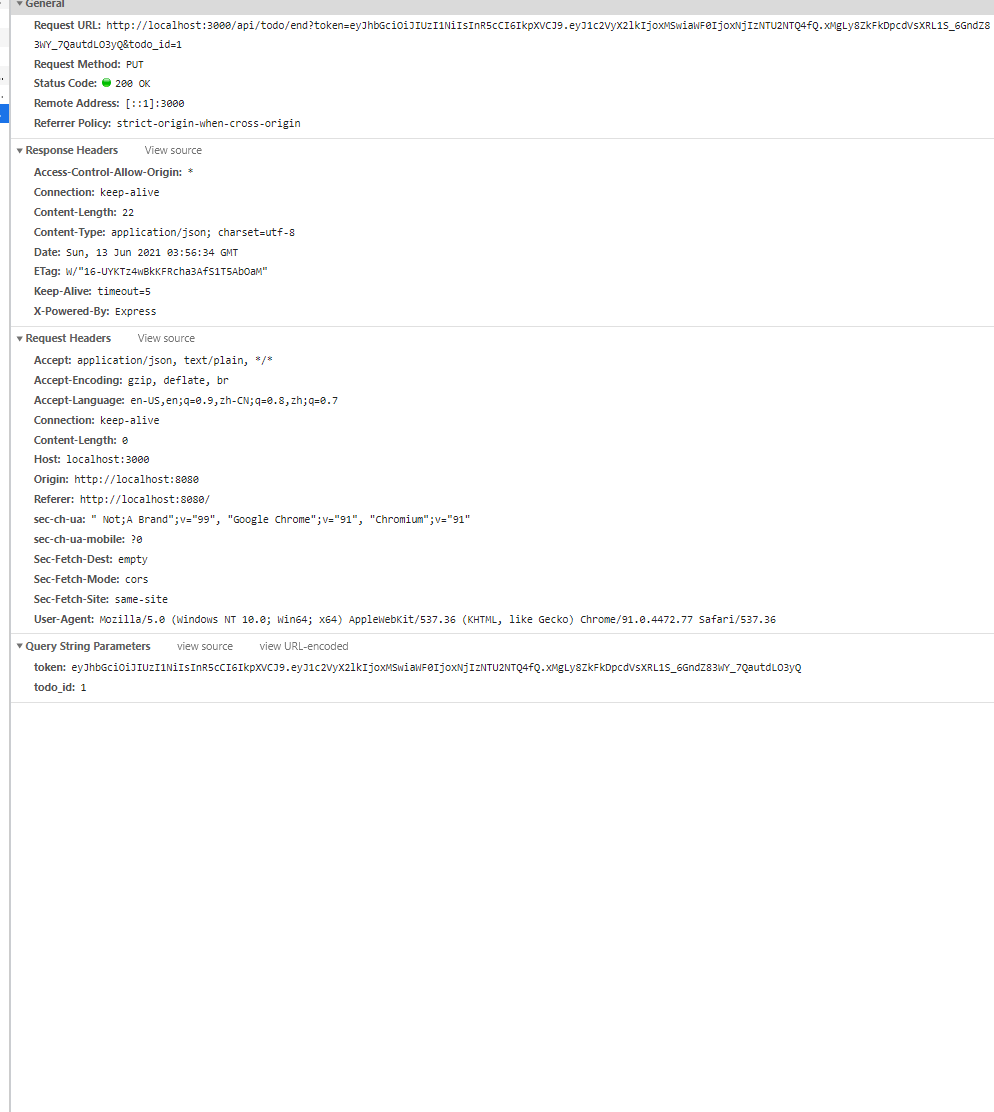
前端请求抓包

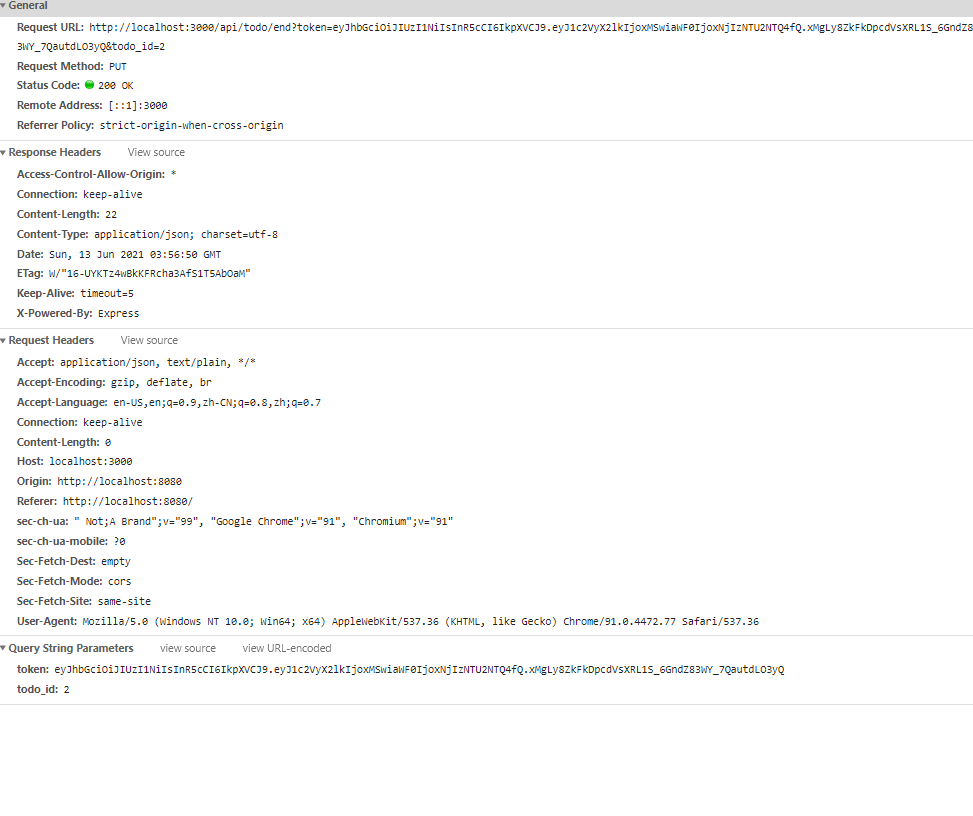


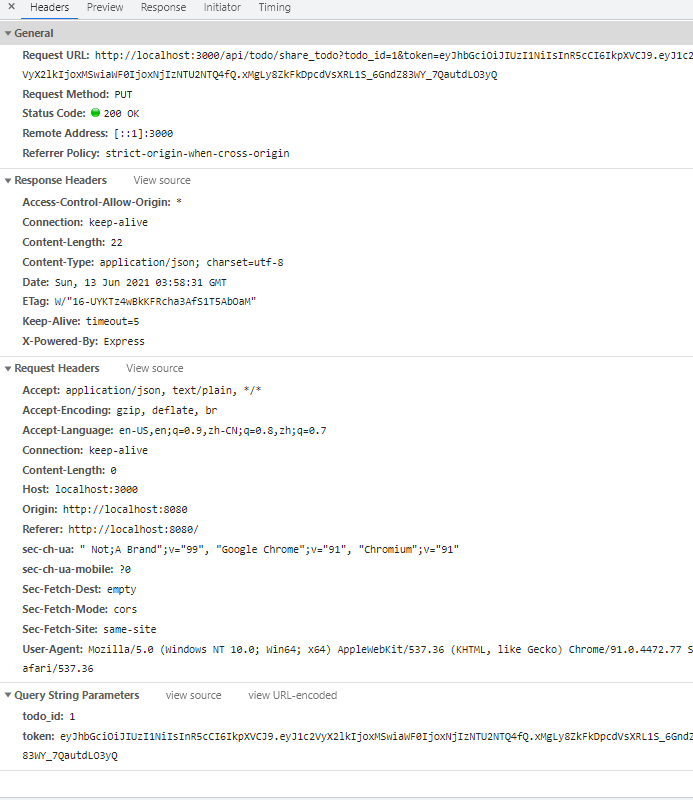


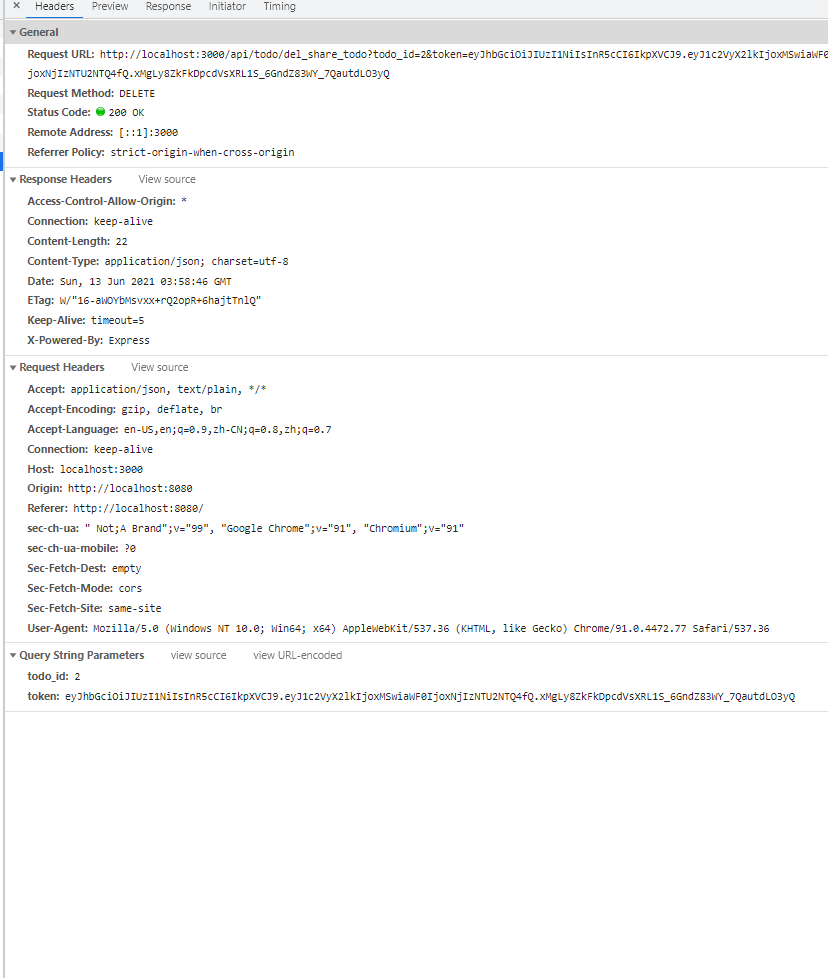


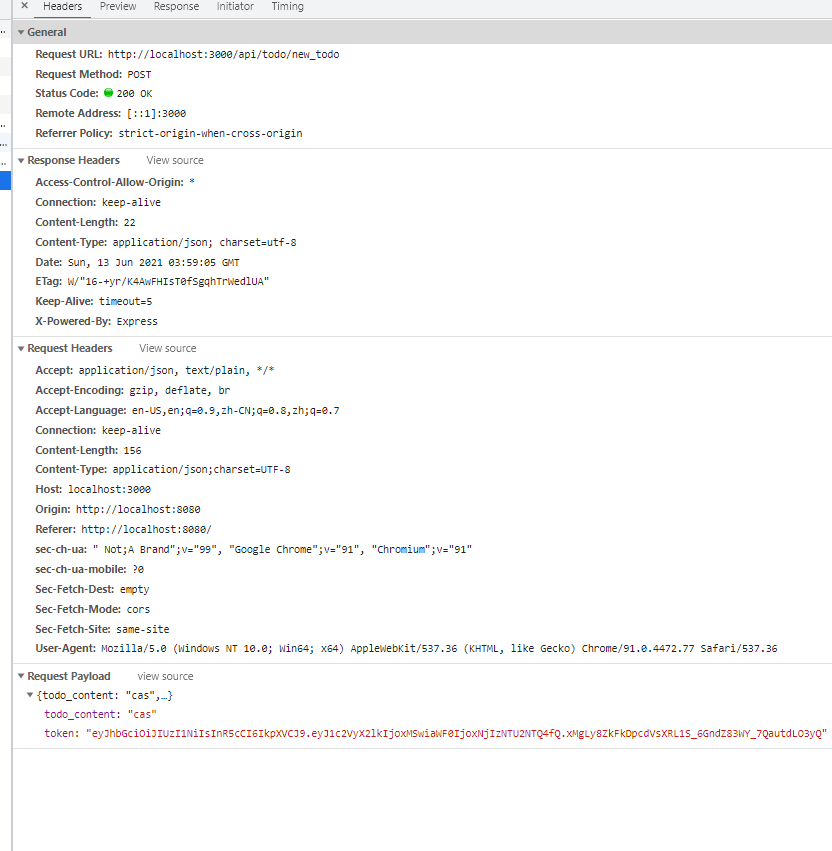


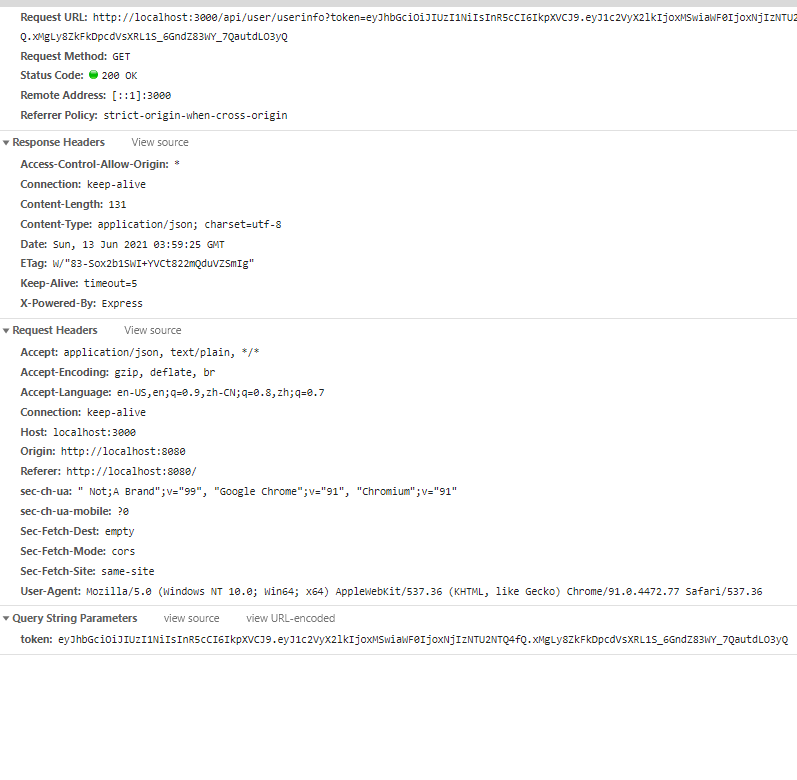




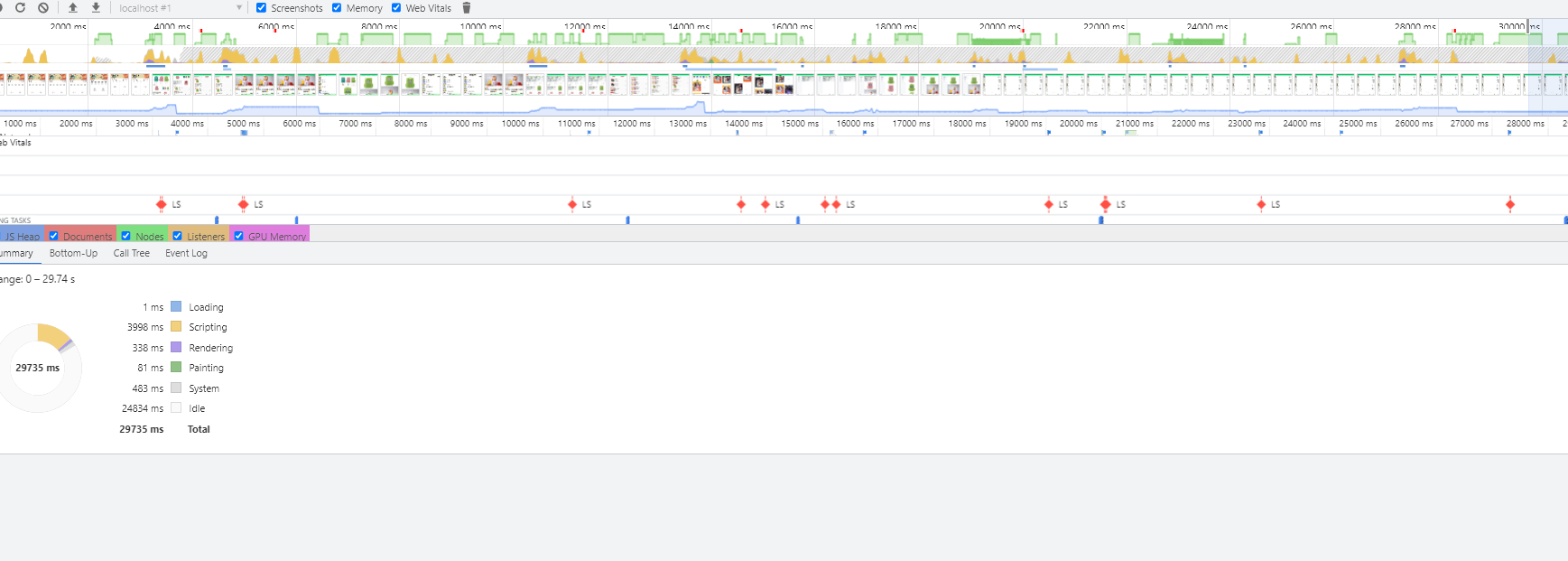




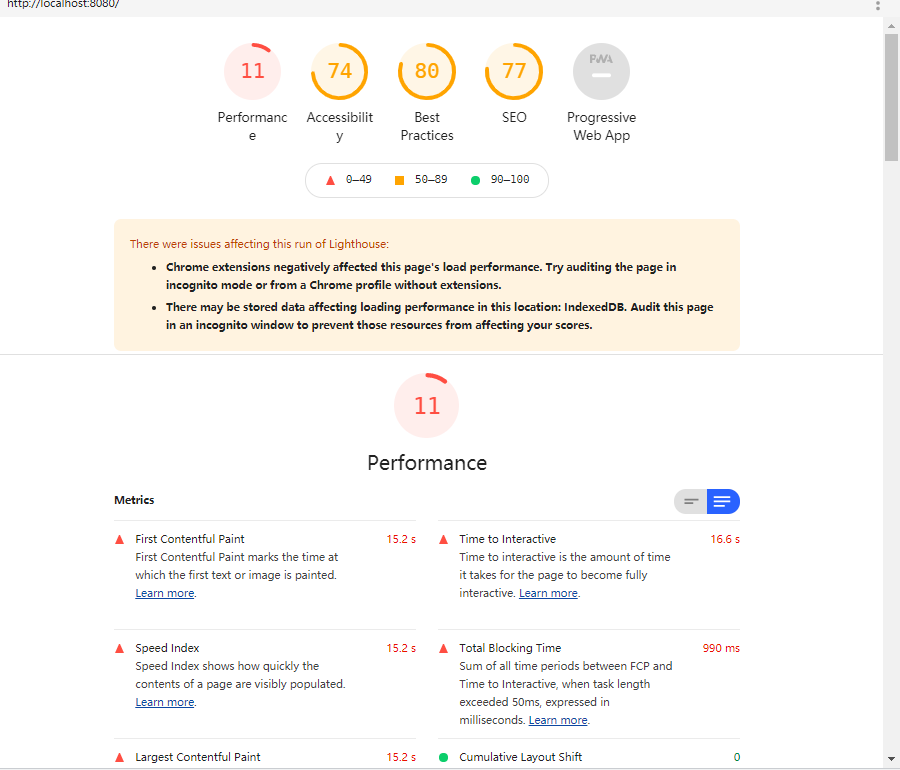


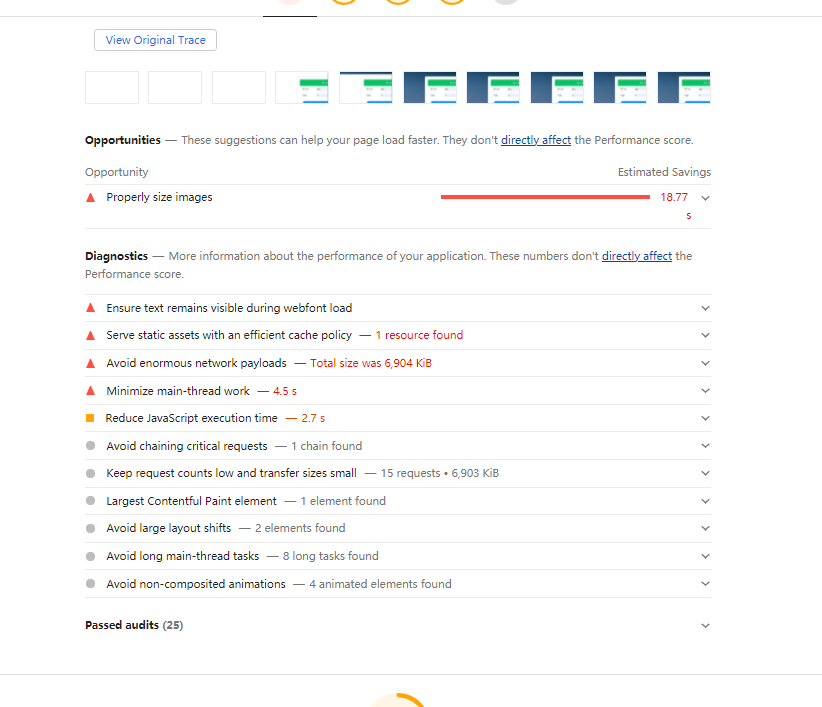


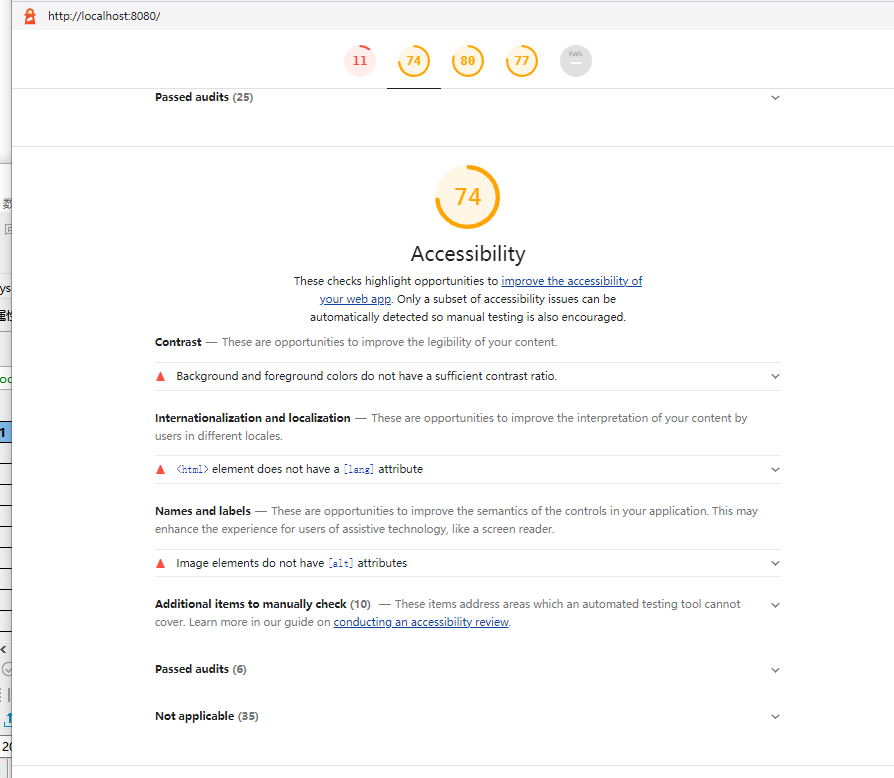
Performance性能分析



LightHouse性能分析









Express 后端部分

框架 采用Express Mysql

MySql 工具使用了navicat

文件目录：

├─.idea 这是编译器的配置信息

│ └─inspectionProfiles

├─bin 这是主要执行的点

├─controller 这里是对数据库的操作

├─dao 这里是连接数据库的池

├─middles 这里放着中间件

├─node\_modules 这是项目依赖

├─public 这是打包后的Vue（run build 后的）文件

│ └─static

│ ├─css

│ ├─fonts

│ └─js

│ ─index.html

├─routes 这里是路由层配置

└─services 这里是类似与orm，对sql进行的封装

Package.json中的项目依赖

{

"name": "shop\_system",

"version": "0.0.0",

"private": true,

"scripts": {

"start": "nodemon ./bin/www"

},

"dependencies": {

"cookie-parser": "~1.4.4",

"cores": "^0.8.5",

"debug": "~2.6.9",

"ejs": "~2.6.1",

"express": "~4.16.1",

"express-session": "^1.17.2",

"http-errors": "~1.6.3",

"mockjs": "^1.1.0",

"morgan": "~1.9.1",

"multer": "^1.4.2",

"mysql2": "^2.2.5",

"vue-puzzle-vcode": "^1.1.5"

},

"devDependencies": {

"cors": "^2.8.5",

"jsonwebtoken": "^8.5.1"

}

}

后端路由：

//获取当前未完成todo

router.get('/current', current);

// 发布新TODO

router.post('/new\_todo', new\_todo);

// 更改Todo状态

router.put('/end', end\_todo);

router.put('/start', start\_todo);

// 历史Todo

router.get('/history', history\_todo);

// 分享 Todo

router.put('/share\_todo', share\_todo);

// 获取所有的分享的Todo、

router.get('/get\_share\_todo', get\_share\_todo);

//取消分享 Todo

router.delete('/del\_share\_todo', del\_share\_todo);

//注册接口

router.post('/register', register);

//登陆接口

router.post('/login', login);

//注销接口

router.all("/logout", loginout);

//获取用户信息

router.get("/userinfo", userinfo)

// 用户修改信息

router.put("/modify", modify)

数据库的连接

//加载mysql数据库连接库

const mysql = require("mysql2");

const config = require("./config")

//调用mysql.createPool方法创建书库连接池

const pool = mysql.createPool(config);

// 中间信息查询确认,实际上线就干掉

pool.on('acquire', function (connection) {

console.log('Connection %d acquired', connection.threadId);

});

pool.on('connection', function (connection) {

connection.query('SET SESSION auto\_increment\_increment=1')

});

pool.on('enqueue', function () {

console.log('Waiting for available connection slot');

});

pool.on('release', function (connection) {

console.log('Connection %d released', connection.threadId);

});

module.exports = pool;

数据库操作部分，进行模块分离

用户的数据操作接口

//使用连接池

let pool = require('../dao/pool');

// 数据采用sha256加密

const crypto = require('crypto');

// 加入JWT ,做用户校验

const jwt = require('jsonwebtoken');

const secretKey = '123114655sad46aa';

const verify = (token, callback) => jwt.verify(token, secretKey, callback);

/\*\*

\* @param {string} user\_account

\* @param {string} user\_password

\* @param {function} callback

\* @return {void}

\* \*/

//根据用户名密码获取用户信息，供登录使用

function getUserByAccountAndPassword({user\_account, user\_password}, callback) {

let hash = crypto.createHash('sha256');

hash.update(user\_password)

pool.query(

"select user\_id from todo\_user where user\_account=? and user\_password=?",

[user\_account, hash.digest("hex")],

callback

);

}

/\*\*

\* @param {string} username

\* @param {string} user\_password

\* @param {string} user\_account

\* @param {string} user\_mobile

\* @param {string} user\_address

\* @param {number} user\_sex

\* @param {function} callback

\* @return {void}

\* \*/

//添加用户信息，供注册使用

function addUser({

username,

user\_password,

user\_account,

user\_mobile,

user\_address,

user\_sex

}, callback) {

// console.log(user)

// 判断优先判断是否存在该用户

let hash = crypto.createHash('sha256');

hash.update(user\_password)

pool.query(

`insert into todo\_user(

username,

user\_password,

user\_account,

user\_mobile,

user\_address,

user\_sex)

values(?,?,?,?,?,?)`,

[username,

hash.digest('hex'),

user\_account,

user\_mobile,

user\_address,

user\_sex,],

callback

);

}

/\*\*

\* @param {string} username

\* @param {string} user\_mobile

\* @param {string} user\_address

\* @param {string} token

\* @param {number} user\_sex

\* @param {function} callback

\* @return {void}

\* \*/

//修改用户基本信息，供修改用户的基本信息

function modifyUser({

username,

user\_mobile,

user\_address,

user\_sex,

token

}, callback) {

verify(

token,

(err, {user\_id}) => {

if (!err) {

pool.query(

`update todo\_user set

username=?,

user\_mobile=?,

user\_address=?,

user\_sex=?

where user\_id=?`,

[username,

user\_mobile,

user\_address,

user\_sex,

user\_id

],

callback

);

}

}

)

}

//根据Token 获取用户的信息

/\*\*

\* @param {string} token

\* @param {function} callback

\* @return {void}

\* \*/

function getUserInfoByToken({token}, callback) {

verify(token, (err, {user\_id}) => {

// console.log(data)

pool.query(

`select

username,user\_account,user\_mobile,user\_address,user\_sex

from todo\_user where user\_id=?`,

[user\_id],

callback

)

})

}

//指明可供其它模块使用的模块

module.exports = {

getUserByAccountAndPassword,

addUser,

getUserInfoByToken,

modifyUser

};

const pool = require('../dao/pool');

const jwt = require('jsonwebtoken');

const secretKey = '123114655sad46aa';

const verify = (token, callback) => {

jwt.verify(

token,

secretKey,

callback);

}

/\*\*

\* @param {number} userId

\* @param {function} callback

\* @return {void}

\* \*/

//查找指定用户所有的任务

function getAllTodo({token}, callback) {

verify(token, (err, {user\_id}) => {

pool.query(

`select

todo\_id,

todo\_content,

todo\_status,

todo\_created\_time,

todo\_set\_finish\_time

from todo\_log

where user\_id=?`,

[user\_id],

callback

);

})

}

/\*\*

\* @param {number} userId

\* @param {function} callback

\* @return {void}

\* \*/

//查找指定用户已完成任务

function getHistoryTodo({token}, callback) {

verify(token, (err, {user\_id}) => {

pool.query(

`select

todo\_id,

todo\_content,

todo\_status,

todo\_created\_time,

todo\_set\_finish\_time

from todo\_log

where status =1 and user\_id=?`,

[user\_id],

callback

);

})

}

/\*\*

\* @param {number} token

\* @param {function} callback

\* @return {void}

\* \*/

//查找指定用户进行中的任务

function getNowTask({token}, callback) {

verify(token, (err, {user\_id}) => {

pool.query(

`select

todo\_id,

todo\_content,

todo\_status,

todo\_created\_time,

todo\_set\_finish\_time,

todo\_shared\_status

from todo\_log

where user\_id=?`,

[user\_id],

callback

);

})

}

/\*\*

\* @param {number} todo\_id

\* @param {function} callback

\* @return {void}

\* \*/

//结束一个任务

function setEndTodo({todo\_id}, callback) {

console.log("end",todo\_id)

pool.query(

`update todo\_log set

todo\_status=0

where todo\_id=?`,

[ todo\_id],

callback

)

}

//开始一个任务

function setStartTodo({todo\_id}, callback) {

console.log("start",todo\_id)

pool.query(

`update todo\_log set

todo\_status=1

where todo\_id=?`,

[todo\_id],

callback

)

}

/\*\*

\* @param {string} todo\_content

\* @param {number} user\_id

\* @param {function} callback

\* \*/

//创建一个新任务

function createTodo({

todo\_content,

token,

}, callback) {

verify(token, (err, {user\_id}) => {

pool.query(

`insert into todo\_log(

user\_id,

todo\_content )

values(?,?)`,

[user\_id, todo\_content],

callback

);

})

}

function setShare({todo\_id}, callback) {

pool.query(

`update todo\_log set

todo\_shared\_status=1

where todo\_id=?`,

[todo\_id],

callback

);

}

function delShare({todo\_id}, callback) {

pool.query(

`update todo\_log set

todo\_shared\_status=0

where todo\_id=?`,

[todo\_id],

callback

);

}

function getShareTodo(callback) {

pool.query(

`select \* from todo\_log where todo\_shared\_status=1 `,

[],

callback

);

}

//提供给其它模块使用

module.exports = {

getHistoryTodo,

createTodo,

setEndTodo,

getNowTask,

getAllTodo,

getShareTodo,

setShare,

delShare,

setStartTodo

};

数据表分布

