《课程备忘录》

程序清单



小 组： G10

小组成员 吴登钻31901236（组长）

赵晟浩 31901244

钟朱楠 31901245

**修 订 记 录**

**类别：**A – 增加 M – 修改 D – 删除

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本** | **日期** | **制定** | **类别** | **批准** | **描述（注明修改的条款或页）** |
| 0.1.0 | 2021-12-14 | 钟朱楠 | A | 吴登钻 | 确定文档目录结构，学生端代码清单 |
| 0.1.1 | 2021-12-21 | 吴登钻 | A | 钟朱楠 | 管理员，教师端代码清单 |
| 0.1.2 | 2021-12-26 | 钟朱楠 | M | 吴登钻 | 更新部分代码清单 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[1.学生端 4](#_Toc91067761)

[1.1云函数 4](#_Toc91067762)

[1.1.1添加课程 4](#_Toc91067763)

[1.1.2.添加个人任务 5](#_Toc91067764)

[1.1.3 添加闹钟 5](#_Toc91067765)

[1.1.4课程是否存在 6](#_Toc91067766)

[1.1.5删除闹钟 7](#_Toc91067767)

[1.1.6编辑个人邮箱 7](#_Toc91067768)

[1.1.7编辑个人电话 8](#_Toc91067769)

[1.1.8完成个人任务 8](#_Toc91067770)

[1.1.9完成课程任务 9](#_Toc91067771)

[1.1.10获取闹钟详情 9](#_Toc91067772)

[1.1.11获取课程任务详情 10](#_Toc91067773)

[1.1.12加载闹钟列表 10](#_Toc91067774)

[1.1.13加载课程界面 10](#_Toc91067775)

[1.1.14加载课程任务列表 11](#_Toc91067776)

[1.1.15登录 12](#_Toc91067777)

[1.1.16查找课程信息 13](#_Toc91067778)

[1.1.17利用时间地点查找课程 13](#_Toc91067779)

[1.1.18设置闹钟 14](#_Toc91067780)

[1.1.19修改闹钟 14](#_Toc91067781)

[1.2组件 15](#_Toc91067782)

[1.2.1 时间 15](#_Toc91067783)

[1.2.2 课程表 25](#_Toc91067784)

[1.3页面 31](#_Toc91067785)

[1.3.1添加闹钟界面 31](#_Toc91067786)

[1.3.2闹钟列表界面 34](#_Toc91067787)

[1.3.3闹钟详情界面 36](#_Toc91067788)

[1.3.4课表界面 40](#_Toc91067789)

[1.3.5课程任务列表 44](#_Toc91067790)

[1.3.6课程任务详情 45](#_Toc91067791)

[1.3.7添加课程界面 48](#_Toc91067792)

[1.3.8修改个人信息界面 58](#_Toc91067793)

[1.3.9添加个人任务界面 62](#_Toc91067794)

[1.3.10个人信息界面 64](#_Toc91067795)

[1.3.11登录界面 67](#_Toc91067796)

[1.3.12注册界面 73](#_Toc91067797)

[1.3.13个人任务列表 80](#_Toc91067798)

[1.3.14个人任务详情 85](#_Toc91067799)

[2教师端 88](#_Toc91067800)

[2.1云函数 88](#_Toc91067801)

[2.1.1添加课程 88](#_Toc91067802)

[2.1.2添加闹钟 89](#_Toc91067803)

[2.1.3课程是否存在 90](#_Toc91067804)

[2.1.4删除闹钟 90](#_Toc91067805)

[2.1.5编辑邮箱 91](#_Toc91067806)

[2.1.6编辑电话 91](#_Toc91067807)

[2.1.7课程任务是否存在 92](#_Toc91067808)

[2.1.8课程是否存在 92](#_Toc91067809)

[2.1.9获取完成率 94](#_Toc91067810)

[2.1.10获取教师课程任务列表 95](#_Toc91067811)

[2.1.2添加课程 96](#_Toc91067812)

[2.2组件 96](#_Toc91067813)

[2.2.1 进度 96](#_Toc91067814)

[2.2.2 进度圈 100](#_Toc91067815)

[2.2.3课程表 108](#_Toc91067816)

[2.3页面 114](#_Toc91067817)

[2.3.1添加闹钟 114](#_Toc91067818)

[2.3.2闹钟列表 117](#_Toc91067819)

[2.3.3闹钟详情 119](#_Toc91067820)

[2.3.4课程表 123](#_Toc91067821)

[2.3.5添加课程 126](#_Toc91067822)

[2.3.6添加课程任务 135](#_Toc91067823)

[2.3.7个人信息 143](#_Toc91067824)

[2.3.8登录 147](#_Toc91067825)

[2.3.9注册 153](#_Toc91067826)

[2.3.10课程任务进度详情 159](#_Toc91067827)

[管理员端 164](#_Toc91067828)

[云函数 164](#_Toc91067829)

[组件 164](#_Toc91067830)

[页面 164](#_Toc91067831)

# 1.学生端

## 1.1云函数

### 1.1.1添加课程

'use strict';

const db = uniCloud.database()

exports.main = async (event, context) => {

//event为客户端上传的参数

console.log('event : ', event)

/\* 最后都会在coourse\_stu表中填加一条记录，表明这个学生与课程有联系\*/

let which\_course = await db.collection('course').where({

/\* 获取刚添加的课程id号 \*/

course\_day: event.course\_day,

start\_time: event.start\_time,

end\_time: event.end\_time,

course\_add: event.course\_add,

}).get()

for (let i = 0; i < 1000; i++) {

if (i == 800) {

let result = await db.collection('course\_stu').add({

course\_id: which\_course.data[0].\_id,

course\_name: event.course\_name,

stu\_idnum: event.idnum,

course\_day: event.course\_day,

start\_time: event.start\_time,

end\_time: event.end\_time,

})

}

}

// let res = await db.collection('course\_stu').add({

// course\_id: which\_course.data[0].\_id,

// course\_name: event.course\_name,

// stu\_idnum: event.idnum,

// })

//返回数据给客户端

return {

courseid: which\_course.data[0].\_id,

msg: '添加课程学生的联系'

}

};

## 1.1.2.添加个人任务

'use strict';

const db = uniCloud.database() //对数据库的对象获取；

exports.main = async (event, context) => {

const collection = db.collection('personal\_task') //对holle数据库的获取；

//event为客户端上传的参数

console.log('event : ', event)

// 添加数据

let res = await collection.add({

idnum:event.idnum,

task\_course:event.task\_course,

task\_title: event.task\_title,

task\_content: event.task\_content,

task\_ddl: new Date(+new Date(event.task\_ddl) + 8 \* 3600 \*1000).toISOString().replace(/T/g,' ').replace(/\.[\d]{3}Z/,''),

iffinish:"未完成",

ing:0

})

return {

code: 200,

msg: '查询成功',

data: res

}

}

## 1.1.3 添加闹钟

'use strict';

const db = uniCloud.database();

const collection = db.collection("alarm");

exports.main = async (event, context) => {

//该闹钟是否存在

let res1 = await collection.where({

alarm\_title:event.alarm\_title,

alarm\_content:event.alarm\_content,

dateTime : new Date(+new Date(event.dateTime) + 8 \* 3600 \*1000).toISOString().replace(/T/g,' ').replace(/\.[\d]{3}Z/,''),

user\_id:event.user\_id

}).get()

console.log(res1)

if (res1.affectedDocs != 0) {

return "该闹钟已存在";

}

let res2 = await collection.add({

alarm\_title:event.alarm\_title,

alarm\_content:event.alarm\_content,

dateTime : new Date(+new Date(event.dateTime) + 8 \* 3600 \*1000).toISOString().replace(/T/g,' ').replace(/\.[\d]{3}Z/,''),

user\_id:event.user\_id

})

return res2;

};

## 1.1.4课程是否存在

'use strict';

const db = uniCloud.database()

exports.main = async (event, context) => {

//event为客户端上传的参数

console.log('event : ', event)

let course\_exist = await db.collection('course').where({

/\* 查看当前课程表中是否存在该课程 \*/

// idnum: event.idnum,

course\_name: event.course\_name,

course\_day: event.course\_day,

start\_time: event.start\_time,

end\_time: event.end\_time,

course\_add: event.course\_add,

}).count()

//返回数据给客户端

return {

code:200,

msg:'获取审核表成功',

coursenum: course\_exist.total

}

};

## 1.1.5删除闹钟

'use strict';

const db = uniCloud.database();

const collection = db.collection("alarm");

exports.main = async (event, context) => {

let res = await collection.where({

\_id:event.\_id

}).remove()

return res

};

## 1.1.6编辑个人邮箱

'use strict';

const db = uniCloud.database() //对数据库的对象获取；

exports.main = async (event, context) => {

let collection = db.collection('student')

let res = await collection.where({

idnum: event.idnum,

}).update({

email: event.email,

})

return {

code: 200,

msg: '邮箱修改成功',

data: res

}

}

## 1.1.7编辑个人电话

'use strict';

const db = uniCloud.database() //对数据库的对象获取；

exports.main = async (event, context) => {

let collection = db.collection('student')

let res = await collection.where({

idnum:event.idnum,

}).update({

tel: event.tel,

})

return {

code: 200,

msg: '查询成功',

data: res

}

}

## 1.1.8完成个人任务

'use strict';

const db = uniCloud.database() //对数据库的对象获取；

exports.main = async (event, context) => {

let collection = db.collection('personal\_task')

let res = await collection.where({\_id:event.task\_id})

.update({

ing:0,

iffinish: "已完成",

}).then((res) => {

uni.showToast({

title: '更新成功'

})

})

return {

code: 200,

msg: '查询成功',

data: res

}

}

### 1.1.9完成课程任务

'use strict';

const db = uniCloud.database();

const collection = db.collection("course\_task\_stu")

exports.main = async (event, context) => {

let res = await collection.where({

course\_task\_id:event.\_id,

stu\_id:event.stu\_id

}).update({

task\_state:"1"

})

return res

};

### 1.1.10获取闹钟详情

'use strict';

const db = uniCloud.database();

const collection = db.collection("alarm");

exports.main = async (event, context) => {

let res = await collection.where({

\_id:event.\_id

}).get()

//返回数据给客户端

return {

data:res.data

}

};

### 1.1.11获取课程任务详情

'use strict';

const db = uniCloud.database();

const collection = db.collection("course\_task")

exports.main = async (event, context) => {

let res= await collection.where({

\_id:event.\_id

}).get()

//返回数据给客户端

return {

data:res.data

}

};

### 1.1.12加载闹钟列表

'use strict';

const db = uniCloud.database();

const collection = db.collection("alarm")

exports.main = async (event, context) => {

let res = await collection.where({

user\_id:event.user\_id

}).get()

//console.log(res.data[0].dateTime)

//返回数据给客户端

return {

data:res.data

}

};

### 1.1.13加载课程界面

'use strict';

const db = uniCloud.database();

const collection = db.collection("course\_stu")

exports.main = async (event, context) => {

let res = await collection.where({

stu\_idnum:event.stu\_idnum

}).get()

return {

data:res.data

}

};

### 1.1.14加载课程任务列表

'use strict';

const db = uniCloud.database();

const collection = db.collection("course\_task")

const collection2 = db.collection("course\_task\_stu")

const dbcmd = db.command

// 课程任务列表

// 根据course\_id获取该课程的所有任务，用当前时间进行筛选，除去ddl已经到了的，获取到一个task的列表

// 再用stu\_id和task\_id查找该列表中finish = 0的,在根据task\_id显示数据

exports.main = async (event, context) => {

const $ = dbcmd.aggregate

let res = await collection.where({

course\_id:event.course\_id

// task\_ddl:dbcmd.gt(new Date())

},

dbcmd.expr(

$.gt(['$time',$.dateFromString({

task\_ddl: new Date('2022-02-02').toISOString()

})])

)

).get()

// console.log(res.data)

// let listTable = []

// //找这个学生没有完成的任务

// for (let i = 0; i < res.data.length; i++) {

// let res2 = await collection2.where({

// task\_id:res.data[i].\_id,

// stu\_idnum:event.stu\_id,

// iffinish: "0"

// }).get()

// listTable.push({

// task\_id:res2.data[0]

// })

// }

// let resTable = []

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

// let res3 = await collection.where({

// \_id:listTable[i].task\_id

// }).get()

// resTable.push(res3.data[0])

// }

return {

data: res.data

}

};

### 1.1.15登录

'use strict';

const db = uniCloud.database();

exports.main = async (event, context) => {

//返回数据给客户端

// return event

const student = db.collection('student');

let res = await student.where({

name:"wdz"

}).get()

//event为客户端上传的参数

console.log('event : ', event)

return{

// content:"成功"

code:200,

msg:"我的名字："+event.name+"我的年龄："+event.age,

context:context

}

};

### 1.1.16查找课程信息

'use strict';

const db = uniCloud.database();

const collection = db.collection("course")

exports.main = async (event, context) => {

let res = await collection.where({

\_id:event.course\_id

}).get()

return {

data:res.data

}

};

### 1.1.17利用时间地点查找课程

'use strict';

const db = uniCloud.database();

const collection = db.collection("course")

exports.main = async (event, context) => {

let res = await collection.where({

course\_day:event.course\_day,

start\_time:event.start\_time,

course\_add:event.course\_add

}).get()

console.log(res)

return {

data:res.data

}

};

### 1.1.18设置闹钟

'use strict';

const db = uniCloud.database();

const collection = db.collection("alarm")

exports.main = async (event, context) => {

//console.log("收到了")

//增加数据

//new Date(+new Date(date\_time)).toISOString().replace(/T/g,' ').replace(/\.[\d]{3}Z/,'')

//console.log(new Date(event.dateTime))

let res = await collection.add({

dateTime : new Date(+new Date(event.dateTime) + 8 \* 3600 \*1000).toISOString().replace(/T/g,' ').replace(/\.[\d]{3}Z/,'')

})

return res;

};

### 1.1.19修改闹钟

'use strict';

const db = uniCloud.database();

const collection = db.collection("alarm");

exports.main = async (event, context) => {

let res = await collection.where({

// alarm\_title:event.alarm\_title,

// alarm\_content:event.alarm\_content,

// dateTime : new Date(+new Date(event.dateTime) + 8 \* 3600 \*1000).toISOString().replace(/T/g,' ').replace(/\.[\d]{3}Z/,'')

\_id:event.\_id

}).update({

alarm\_title:event.alarm\_title,

alarm\_content:event.alarm\_content,

dateTime: new Date(+new Date(event.dateTime) + 8 \* 3600 \*1000).toISOString().replace(/T/g,' ').replace(/\.[\d]{3}Z/,'')

})

return res

};

## 1.2组件

### 1.2.1 时间

<template>

<view>

<view class="long-data">

<view class="long-data-title long-data-fl">提醒时间</view>

<!-- <view class="long-data-fr long-data-changeTimeIcon" @tap="openStatus">></view> -->

<view class="long-data-changeTime long-data-fr" @tap="tapIsShow">{{year}}年{{month}}月{{day}}日 {{week}} {{hour}}:{{minute}}{{ type == 'day' ? '(' + desc + ')' : ''}}

<view v-if="showCheck" class="long-data-check">{{checkStr}}</view>

</view>

<!-- <view class="long-data-check-triangle"></view> -->

</view>

<view class="long-data-picker" :style="'display:' + [isShow ? 'block' : 'none']">

<!-- 触发事件bindDateChange -->

<picker-view :indicator-style="itemHeight" :value="dateValues" @change="bindDateChange">

<picker-view-column>

<view class="long-datetime-item" v-for="(item,index) in dateObj.dates" :key="index">{{item}}</view>

</picker-view-column>

<picker-view-column>

<view class="long-datetime-item" v-for="(item,index) in dateObj.hours" :key="index">{{item}}时</view>

</picker-view-column>

<picker-view-column>

<view class="long-datetime-item" v-for="(item,index) in dateObj.minutes" :key="index">{{item}}分</view>

</picker-view-column>

</picker-view>

</view>

</view>

</template>

<script>

export default {

name: "long-date",

props: {//子组件接收父组件传来的值

openStatus: { //是否展开状态

type: Boolean,

default () {

return false //默认不展开

}

},

type: { //模式

type: String,

default () {

// return 'between' //根据开始时间和结束模式获取

return 'day' //以天数获取开始时间模式

}

},

getDayNum: { //获取天数（day模式可用）

type: Number,

default () {

return 14

}

},

chooesMaxDay: { //是否设置最大选择天数 （day模式可用）

type: Number,

default () {

return 0 // 0 默认不设置 >0代表天数

}

},

startTime: { //开始时间（between模式可用）

type: String,

default () {

return ''

}

},

endTime: { //结束时间（between模式可用）

type: String,

default () {

return ''

}

},

},

data() {

switch (this.type) {

case 'between':

//获取开始时间

var startDate = new Date(this.startTime + ' 00:00:00')

//获取结束时间

var endDate = new Date(this.endTime + ' 00:00:00');

//设置默认选择时间

var chooesDate = new Date(this.startTime + ' 00:00:00')

//获取多少天日期

var num = (endDate - startDate) / 86400000

break;

case 'day':

//获取当前时间

var startDate = new Date();

//设置默认选择时间

var chooesDate = new Date();

//获取多少天日期

var num = this.getDayNum

break;

}

//获取年份

let year = chooesDate.getFullYear()

//获取月份

let month = chooesDate.getMonth() + 1

//获取当前日(1-31)

let day = chooesDate.getDate()

//获取当前的小时 并格式化1-10数字

let h = chooesDate.getHours()

let hour = h < 10 ? '0' + h : h

//获取当前的分钟数 并格式化1-10数字

let m = chooesDate.getMinutes()

let minute = m < 10 ? '0' + m : m

//设置日期数组

let dates = []

//设置小时数数组

let hours = []

//设置分钟数数组

let minutes = []

for (let i = 0; i <= num; i++) { //获取包括的日期

if (i != 0) {

startDate.setDate(startDate.getDate() + 1)

}

let str\_year = startDate.getFullYear()

let str\_month = startDate.getMonth() \* 1 + 1

let str\_day = startDate.getDate()

if (str\_month < 10) {

str\_month = '0' + str\_month;

}

if (str\_day < 10) {

str\_day = '0' + str\_day;

}

dates.push(str\_year + '-' + str\_month + '-' + str\_day)

}

for (let i = 0; i < 24; i++) {

let str = i;

if (i < 10) {

str = '0' + str;

} else {

str = '' + str;

}

hours.push(str);

}

for (let i = 0; i < 60; i++) {

let str = i;

if (i < 10) {

str = '0' + str;

} else {

str = '' + str;

}

minutes.push(str);

}

let dateObj = {

dates,

hours,

minutes,

}

return {

chooesDate,

year,

month,

day,

hour,

minute,

// second,

dateObj,

checkStr: '时间不能小于当前时间',

showCheck: false,

isShow: this.openStatus,

selectRes: '',

itemHeight: `height: ${uni.upx2px(88)}px;`,

dateValues: [0, 0, 0]

};

},

computed: {

//获取几天后的描述

desc: function() {

let chooes\_time = new Date(this.chooesDate.toLocaleDateString())

// var chooes\_time = Date.parse(new Date('2019-10-30 10:00:00')) / 1000

let now\_time = new Date(new Date().toLocaleDateString())

//获取多少秒

let haveSecond = (chooes\_time - now\_time) / 86400000;

if (haveSecond == 0) {

return '今天'

} else if (haveSecond == 1) {

return '明天'

} else if (haveSecond == 2) {

return '后天'

} else if (haveSecond == 3) {

return '3天后'

} else if (haveSecond == 4) {

return '4天后'

} else if (haveSecond == 5) {

return '5天后'

} else if (haveSecond == 6) {

return '6天后'

} else if (haveSecond == 7) {

return '一周后'

}

},

//获取星期几

week: function() {

let week = this.chooesDate.getDay()

switch (week) {

case 1:

return '周一'

break;

case 2:

return '周二'

break;

case 3:

return '周三'

break;

case 4:

return '周四'

break;

case 5:

return '周五'

break;

case 6:

return '周六'

break;

case 0:

return '周日'

break;

}

},

},

mounted() {

this.initDate();

},

methods: {

//初始化时间

initDate() {

},

//切换日期选择显示状态

tapIsShow() {

this.isShow = !this.isShow

},

//滚动切换时间

bindDateChange(e) { //有效日期的滚动日期时间方法

let valueArr = e.detail.value;

let dateStr = this.dateObj.dates[valueArr[0]];

let hourStr = this.dateObj.hours[valueArr[1]];

let minuteStr = this.dateObj.minutes[valueArr[2]];

let chooes\_time = new Date(dateStr + ' ' + hourStr + ':' + minuteStr)

this.chooesDate = chooes\_time

this.year = chooes\_time.getFullYear()

this.month = chooes\_time.getMonth() + 1

this.day = chooes\_time.getDate()

let h = chooes\_time.getHours()

let m = chooes\_time.getMinutes()

this.hour = h < 10 ? '0' + h : h

this.minute = m < 10 ? '0' + m : m

let max\_day = this.chooesMaxDay;

if (max\_day > 0 && this.type == 'day') {

let haveSecond = chooes\_time - new Date();

if (haveSecond < 0) {

this.checkStr = '时间不能小于当前时间'

this.showCheck = true

} else {

if ((haveSecond / 86400000) > max\_day) {

this.checkStr = '时间不能大于' + max\_day + '天'

this.showCheck = true

} else {

this.checkStr = ''

this.showCheck = false

}

}

}

this.selectRes = this.year + '-' + this.month + '-' + this.day + ' ' + this.hour + ':' + this.minute;

//父类方法名，传递的值

// this.$emit("select", {

// time: this.selectRes

// });

this.$emit("select", this.selectRes);

}

},

}

</script>

<style>

.long-data {

margin-top: 30rpx;

height: 80rpx;

background: #FFFFFF;

line-height: 80rpx;

padding-left: 30rpx;

padding-right: 30rpx;

border-bottom: 1px solid #eee;

/\* position: relative; \*/

}

.long-data-check {

height: 40rpx;

width: 100%;

background: #e54d42;

position: absolute;

left: 0;

top: 18rpx;

color: #fff;

line-height: 45rpx;

border-radius: 0rpx;

padding: 0px 20rpx;

font-size: 20rpx;

text-align: center;

border-radius: 20rpx;

}

.long-data-check-triangle {

width: 0;

height: 0;

border-top: 12rpx solid transparent;

border-left: 10rpx solid #e54d42;

border-bottom: 12rpx solid transparent;

position: absolute;

right: 223rpx;

top: 26rpx;

}

.long-data-fl {

float: left;

}

.long-data-fr {

float: right;

}

.long-data-changeTime {

color: #888;

font-size: 25rpx;

position: relative;

text-align: right;

padding: 0rpx 20rpx;

}

.long-data-changeTimeIcon {

color: #888;

}

.long-data-picker {

width: 100%;

height: 250rpx;

overflow: hidden;

background: #fff;

transition: height 0.3s;

}

.long-datetime-item {

text-align: center;

width: 100%;

height: 88upx;

line-height: 88upx;

text-overflow: ellipsis;

white-space: nowrap;

font-size: 30upx;

}

.long-data-picker picker-view {

height: 100%;

}

</style>

### 1.2.2 课程表

<template>

<view class="timetable">

<view class="header"> <!-- 显示周几-->

<view class="header-item" v-for="(item,index) in week" :key="item" :style="{ color: todayWeekIndex === index ? '#4070FF' : 'unset' }">{{ item }}</view>

</view>

<view class="main">

<view class="row" v-for="(item,index) in timetableType" :key="index"><!-- 显示节数及每节课时间 -->

<view class="time-item" >

<view class="index">{{ item.index }}</view><!-- 显示节数 -->

<view class="time">{{ item.name }}</view><!-- 显示每节课的上课时间 -->

</view>

</view>

<view class="course-container">

<view class="week" v-for="(week, weekIndex) in courseData" :key="weekIndex">

<view class="courseList" v-for="(course, courseIndex) in week" :key="courseIndex">

<view @click="handleCourseClick(course, weekIndex, courseIndex)" class="course" :style="{ height: (course.length \* 79) + 'px', background: course.backgroundColor}" v-if="course.length > 0">{{ course.name }}</view>

<!-- <view @click="testCourseClick()"class="course" :style="{ height: (course.length \* 79) + 'px', background: course.backgroundColor}" v-if="course.length > 0">{{ course.name }}</view> -->

</view>

</view>

</view>

</view>

</view>

</template>

<script>

export default {

name: 'Timetable',

props: {

timetableType: {

type: Array,

default: () => {

return [

{ index: '1', name: '08:00\n08:45' },

{ index: '2', name: '08:50\n09:35' },

{ index: '3', name: '09:50\n10:35' },

{ index: '4', name: '10:40\n11:25' },

{ index: '5', name: '11:30\n12:15' },

{ index: '6', name: '13:30\n14:15' },

{ index: '7', name: '14:20\n15:05' },

{ index: '8', name: '15:20\n16:05' },

{ index: '9', name: '16:10\n16:55' },

{ index: '10', name: '18:30\n19:15' },

{ index: '11', name: '19:20\n20:05' },

{ index: '12', name: '20:10\n20:55' }

]

}

},

week: {

type: Array,

default: () => {

return ['一', '二', '三', '四', '五', '六', '日']

}

},

timetables: {

type: Array,

default: () => {

return []

}

},

palette: {

type: Array,

default: () => {

return []

}

}

},

data () {

return {

allPalette: [...this.palette, '#f05261', '#48a8e4', '#ffd061', '#52db9a', '#70d3e6', '#52db9a', '#3f51b5', '#f3d147', '#4adbc3', '#673ab7', '#f3db49', '#76bfcd', '#b495e1', '#ff9800', '#8bc34a'],

course\_id:""

}

},

computed: {

courseData () {

// 为数据标记背景颜色的函数

let paletteIndex = 0

const getBackgroundColor = () => {

const backgroundColor = this.allPalette[paletteIndex]

paletteIndex++

if (paletteIndex >= this.allPalette.length) {

paletteIndex = 0

}

return backgroundColor

}

// 合并

const listMerge = []

this.timetables.forEach(function (list, i) {

if (!listMerge[i]) {

listMerge[i] = []

}

list.forEach(function (item, index) {

if (!index) {

return listMerge[i].push({ name: item, length: 1, backgroundColor: item === '' ? 'none' : getBackgroundColor() })

}

if (item === (listMerge[i][index - 1] || {}).name && item) {

const sameIndex = (listMerge[i][index - 1] || {}).sameIndex

if (sameIndex || sameIndex === 0) {

listMerge[i][sameIndex].length++

return listMerge[i].push({ name: item, length: 0, sameIndex: sameIndex })

}

listMerge[i][index - 1].length++

return listMerge[i].push({ name: item, length: 0, sameIndex: index - 1 })

} else {

return listMerge[i].push({ name: item, length: 1, backgroundColor: item === '' ? 'none' : getBackgroundColor() })

}

})

})

return listMerge

},

todayWeekIndex () {

let weekIndex = new Date().getDay() - 1

if (weekIndex === -1) {

weekIndex = 6

}

return weekIndex

}

},

methods: {

handleCourseClick:function(course, weekIndex, courseIndex) {

const data = {

index: courseIndex + 1,

length: course.length,

week: this.week[weekIndex],

weekIndex: weekIndex,

name: course.name

}

console.log(`星期${data.week}; 第${data.index}节课; 课程名:${data.name}; 课节:${data.length}`)

console.log(data)

//根据周几第几节课和地点，查该课程的\_id

let addr=data.name.split('@')

let week = "周" + data.week

let index = data.index

let add = addr[1]

uniCloud.callFunction({

name:"searchCourseByTimeAddr",

data:{

course\_day:week,

start\_time:index+'',

//start\_time:"3",

course\_add:add

}

}).then((res)=>{

const{result} = res

console.log(result)

this.course\_id = result.data[0].\_id

console.log(this.course\_id)

uni.navigateTo({

url:'/pages/CourseTask/CourseTask?\_id=' + this.course\_id

// url:'/pages/course\_task/course\_task?\_id=61b6079bb804f80001909fcc'// + this.course\_id

})

})

this.$emit('courseClick', data)//子组件向父组件传值,,第一个值为父组件定义函数，第二个是需要传递的参数

},

// toDetail() {

// this.$emit("click",event);

// console.log("点击了！")

// console.log(this.course\_id)

// uni.navigateTo({

// url:'../course\_task/course\_task?\_id='+this.course\_id

// })

// },

testCourseClick:function(){

console.log("test")

},

toTask:function(){

}

}

}

</script>

<style scoped lang="scss">

.timetable{

background: white;

border: 1px solid #E4E7ED;

border-radius: 8rpx;

.header{

padding-left: 88rpx;

height: 56rpx;

display: flex;

align-items: center;

background: #F5F7FA;

.header-item{

flex: 1;

text-align: center;

}

}

.main{

position: relative;

.row{

height: 85px;

position: relative;

&:after{

content: '';

height: 0;

width: 100%;

position: absolute;

bottom: 0;

left: 0;

border-bottom: 1rpx dashed #E4E7ED;

}

.time-item{

height: 100%;

width: 88rpx;

text-align: center;

background: #F5F7FA;

.index{

color: #909399;

padding-bottom: 8rpx;

padding-top: 16rpx;

}

}

}

.course-container{

position: absolute;

top: 0;

left: 90rpx;

width: calc(100% - 280rpx);

// width: 88rpx;

height: 100%;

display: flex;

.week{

flex: 1;

width: 0;

flex-grow: 1;

display: flex;

flex-direction: column;

.courseList{

word-break: break-all;

color: white;

overflow: hidden;

.course{

width: 70rpx;

padding: 8rpx;

left:40rpx;

// border-radius: 16rpx;

border-radius: 16rpx;

text-align: center;

}

}

}

}

}

}

</style>

## 1.3页面

### 1.3.1添加闹钟界面

<template>

<view class = "all">

<view class="main" >

<view class="alarm" id="alarm\_title">

<input class = "alarm\_input" placeholder="闹钟标题" v-model="alarm\_title"></input>

<!-- <input v-model="inputSerialNum" style="padding: 10rpx;font-size: 30rpx;" class="uni-input" focus placeholder="请输入值" /> -->

</view>

<view class="alarm" id="alarm\_content">

<textarea class = "alarm\_input" placeholder="请输入闹钟内容" v-model="alarm\_content"></textarea>

</view>

<long-date

type="day"

getDayNum="10"

:openStatus="true"

startTime="2021-10-01"

endTime="2022-06-01"

chooesMaxDay="8"

@select="Time">

</long-date>

<view id="submit">

<button type="primary" @click="addAlarm()">添加</button>

</view>

</view>

</view>

</template>

<script>

//导入子组件

import longDate from "@/components/long-date/long-date.vue"

export default {

components: {

longDate

},

data() {

return {

alarm\_title:"",

alarm\_content:"",

dateTime : '',

user\_id:"31901245"

}

},

methods: {

Time(val) {

//console.log(1)

this.dateTime = val

//console.log(val)

},

addAlarm() {

var alarmMills = new Date(this.dateTime).getTime()

var nowMills = new Date().getTime()

console.log(alarmMills)

console.log(nowMills)

if (alarmMills - nowMills < 0) {

uni.showToast({

title: '闹钟时间不能小于当前时间',

icon: 'none',

duration:1000

})

return false;

}

uniCloud.callFunction({

name:"addAlarm",

data:{

alarm\_title:this.alarm\_title,

alarm\_content:this.alarm\_content,

dateTime:this.dateTime,

user\_id:this.user\_id

},

success(res) {

console.log(res)

},

fail(res) {

console.log(res)

}

})

uni.switchTab({

url:'../Alarm/Alarm'

})

}

}

}

</script>

<style>

.main{

background-color: #FFFFFF;

height: 1200rpx;

margin-top: 30px;

}

.alarm{

background-color: #F2F2F2;

height: 70rpx;

/\* border:#c9daf8; \*/

border-radius: 20rpx;

/\* border-style:groove;\*/

margin-top: 30rpx;

margin-left: 20rpx;

margin-right: 20rpx;

padding-left: 30rpx;

padding-right: 30rpx;

padding-top: 15rpx;

}

.alarm\_input {

font-size: 38rpx;

}

#alarm\_title {

}

#alarm\_content{

height: 200rpx;

}

#submit{

margin: 40px;

}

</style>

### 1.3.2闹钟列表界面

<template>

<view>

<view >

<!-- <view id="1">

<button type="primary" @click="loadAlarm()">加载</button>

</view> -->

<uni-list>

<uni-list >

<view v-for="item in list" @click="toDetail(item.\_id)">

<uni-list-item :title="item.dateTime" :note="item.alarm\_title" ></uni-list-item>

</view>

</uni-list>

</uni-list>

</view>

</view>

</template>

<script>

export default {

data() {

return {

user\_id:"",

list:[]

}

},

onLoad(){

this.user\_id = uni.getStorageSync("globalUser");

this.loadAlarm()

},

onShow(){

this.loadAlarm()

},

methods: {

toDetail(\_id){

//监听

this.$emit("click",event);

uni.navigateTo({

url:'../AlarmDetail/AlarmDetail?\_id=' + \_id

//url:'../alarm\_detail/alarm\_detail?alarm\_title=' + i + '&alarm\_content=' + c

//url:'../alarm\_detail/alarm\_detail?alarm\_title=' + list[index].alarm\_title + '&alarm\_content=' + list[index].alarm\_content

})

},

loadAlarm() {

uniCloud.callFunction({

name:"loadAlarm",

data:{

user\_id:this.user\_id

}

}).then((res)=>{

const {result} = res

this.list = result.data

})

}

},

onNavigationBarButtonTap:function(e){

// console.log(e.text);//提交

// console.log(e.fontSize);//16px

uni.navigateTo({

url:'../AddAlarm/AddAlarm'

})

}

}

</script>

<style>

</style>

### 1.3.3闹钟详情界面

<template>

<view class = "all">

<view class="main" >

<view class="Alarm" id="Alarm\_title">

<input class = "Alarm\_input" placeholder="闹钟标题" v-model="Alarm\_title"></input>

<!-- <input v-model="inputSerialNum" style="padding: 10rpx;font-size: 30rpx;" class="uni-input" focus placeholder="请输入值" /> -->

</view>

<view class="Alarm" id="Alarm\_content">

<textarea class = "Alarm\_input" placeholder="请输入闹钟内容" v-model="Alarm\_content"></textarea>

</view>

<long-date

type="day"

getDayNum="10"

:openStatus="true"

startTime="2021-10-01"

endTime="2022-06-01"

chooesMaxDay="8"

@select="Time">

</long-date>

<view id="submit">

<button type="primary" @click="updateAlarm()">修改</button>

</view>

<view id="delete">

<button type="primary" @click="deleteAlarm()">删除</button>

</view>

</view>

</view>

</template>

<script>

//导入子组件

import longDate from "@/components/long-date/long-date.vue"

export default {

components: {

longDate

},

data() {

return {

\_id:"",

Alarm\_title:"",

Alarm\_content:"",

dateTime : ''

}

},

onLoad(option) {

this.\_id = option.\_id

this.getAlarmDetail(option);

},

methods: {

getAlarmDetail(option) {

uniCloud.callFunction({

name:'getAlarmDetail',

data:{

\_id : this.\_id

}

}).then((res)=>{

const {result} = res

this.Alarm\_title = result.data[0].Alarm\_title

this.Alarm\_content = result.data[0].Alarm\_content

console.log(this.Alarm\_title)

})

},

Time(val) {

//console.log(1)

this.dateTime = val

//console.log(val)

},

updateAlarm() {

//console.log("执行了吗")

//console.log(this.dateTime)

uniCloud.callFunction({

name:"updateAlarm",

data:{

\_id:this.\_id,

Alarm\_title:this.Alarm\_title,

Alarm\_content:this.Alarm\_content,

dateTime:this.dateTime

},

success(res) {

},

fail(res) {

console.log(res)

}

})

uni.switchTab({

url:'../Alarm/Alarm'

})

},

deleteAlarm() {

uniCloud.callFunction({

name:"deleteAlarm",

data:{

\_id:this.\_id

},

success(res) {

console.log(res)

},

fail(res) {

console.log(res)

}

}),

uni.switchTab({

url:'../Alarm/Alarm'

})

}

}

}

</script>

<style>

.main{

background-color: #FFFFFF;

height: 1200rpx;

margin-top: 30px;

}

.Alarm{

background-color: #F2F2F2;

height: 70rpx;

/\* border:#c9daf8; \*/

border-radius: 20rpx;

/\* border-style:groove;\*/

margin-top: 30rpx;

margin-left: 20rpx;

margin-right: 20rpx;

padding-left: 30rpx;

padding-right: 30rpx;

padding-top: 15rpx;

}

.Alarm\_input {

font-size: 38rpx;

}

#Alarm\_title {

}

#Alarm\_content{

height: 200rpx;

}

#submit{

margin-left: 50px;

margin-right:50px;

margin-top: 30px;

}

#delete{

margin-left: 50px;

margin-right:50px;

margin-top: 20px;

}

</style>

### 1.3.4课表界面

<template>

<view>

<view>

<lpxtimetable :timetables="timetables"></lpxtimetable>

</view>

</view>

</template>

<script>

import lpxtimetable from '../../components/lpx-timetable/lpx-timetable.vue'

export default {

data() {

return {

timetables: [

['', '', '', '', '', '', '', '', '', '', '', ''],

['', '', '', '', '', '', '', '', '', '', '', ''],

['', '', '', '', '', '', '', '', '', '', '', ''],

['', '', '', '', '', '', '', '', '', '', '', ''],

['', '', '', '', '', '', '', '', '', '', '', '']

],

courseList: [

],

stu\_idnum: "",

course\_name: "",

course\_day: "",

start\_time: "",

end\_time: "",

course\_add: "",

theCourse: []

}

},

onLoad() {

// this.stu\_idnum = "31901238"

this.stu\_idnum = uni.getStorageSync("globalUser")

this.loadCourse()

},

onShow(){

this.loadCourse()

},

onPullDownRefresh() {

//监听下拉刷新动作的执行方法，每次手动下拉刷新都会执行一次

console.log('refresh');

this.loadCourse();

setTimeout(function() {

uni.stopPullDownRefresh(); //停止下拉刷新动画

}, 1000);

},

methods: {

loadCourse() {

uniCloud.callFunction({

name: "loadCourse",

data: {

stu\_idnum: this.stu\_idnum

}

}).then((res) => {

const {

result

} = res

this.courseList = result.data

this.updateTimeTables()

})

},

updateTimeTables() {

//for循环根据每一门课的\_id到course中查找具体的课程详细,并改变timetables

for (var i = 0; i < this.courseList.length; i++) {

//this.searchCourse(this.courseList[i].course\_id)

this.searchCourse(this.courseList[i].course\_id)

// for (var j = this.start\_time; j <= this.end\_time; j++) {

// // console.log(j)

// //console.log(this.timetables[this.course\_day][j])

// this.timetables[this.course\_day][j] = this.course\_name

// }

}

},

searchCourse(course\_id) {

uniCloud.callFunction({

name:'searchCourse',

data:{

course\_id:course\_id

}

}).then((res)=>{

const {result} = res

this.theCourse = result.data

//该课程的详细信息

this.course\_name = result.data[0].course\_name

if (result.data[0].course\_day == "周一") {

this.course\_day = 0

} else if (result.data[0].course\_day == "周二") {

this.course\_day = 1

} else if (result.data[0].course\_day == "周三") {

this.course\_day = 2

} else if (result.data[0].course\_day == "周四") {

this.course\_day = 3

} else if (result.data[0].course\_day == "周五") {

this.course\_day = 4

}

//console.log(this.course\_name)

//console.log(this.course\_day)

this.start\_time = result.data[0].start\_time

this.end\_time = result.data[0].end\_time

this.course\_add = result.data[0].course\_add

//console.log(this.start\_time + " " + this.end\_time)

for (var j = this.start\_time; j <= this.end\_time; j++) {

//this.timetables[this.course\_day][j - 1]= this.course\_name

this.$set(this.timetables[this.course\_day],j - 1,this.course\_name +"@"+this.course\_add)

}

//console.log("周三第二节课" + this.timetables[2][1])

//console.log("周三第三节课" + this.timetables[2][2])

// console.log(this.timetables)

// this.timetables[0][1] = "软件工程"

//this.$set(this.timetables[0],1,"软件工程")

//console.log(this.timetables[0][1])

})

}

},

components: {

lpxtimetable

},

onNavigationBarButtonTap: function(e) {

console.log(e.text); //提交

console.log(e.fontSize); //16px

uni.navigateTo({

url: '../EditCourse/EditCourse'

})

}

}

</script>

<style>

</style>

### 1.3.5课程任务列表

<template>

<view>

<uni-list>

<uni-list>

<view v-for="item in course\_task\_list" @click="loadCourseTaskDetail(item.\_id)" >

<uni-list-item :title = "'@'+item.course\_name+'：'+item.task\_title" :note="'截止时间：' + item.task\_ddl" show-arrow="true" ></uni-list-item>

</view>

</uni-list>

</uni-list>

</view>

</template>

<script>

export default {

data() {

return {

course\_id:"",

course\_task\_list:[]

}

},

onLoad(option){

this.course\_id = option.\_id

// this.course\_id = '61b6079bb804f80001909fcc'

this.getCourseTaskList();

},

onShow(){

// this.course\_id = option.\_id

this.getCourseTaskList();

},

methods: {

getCourseTaskList() {

console.log(this.course\_id)

console.log(uni.getStorageSync("globalUser"))

uniCloud.callFunction({

name:"loadCourseTaskList",

data:{

course\_id:this.course\_id,

stu\_id:uni.getStorageSync("globalUser")

}

}).then((res)=>{

const{result} = res

this.course\_task\_list = result.data

console.log("结果是啥")

// console.log(this.course\_task\_list[0])

console.log(this.course\_task\_list.length)

})

},

loadCourseTaskDetail(\_id) {

this.$emit("click",event)

uni.navigateTo({

url:'../CourseTaskDetail/CourseTaskDetail?\_id=' + \_id

})

}

}

}

</script>

<style>

</style>

### 1.3.6课程任务详情

<template>

<view>

<view class = "page\_name">

<p>任务详情</p>

</view>

<view class = "task\_title">

<p>{{task\_title}}</p>

</view>

<view class = "course\_name">

<p>{{course\_name}}</p>

</view>

<view class = "task\_content">

<p>{{task\_content}}</p>

</view>

<view class = "end\_time">

<p>截止时间：{{task\_ddl}}</p>

</view>

<view class = "finish">

<button class = "finish" type = "primary" @click="finished()">完成</button>

</view>

</view>

</template>

<script>

export default {

data() {

return {

\_id : "",

course\_name:"",

task\_title:"",

task\_content:"",

task\_ddl:""

}

},

onLoad(option){

this.\_id = option.\_id

console.log(this.\_id)

this.getCourseTaskDetail();

},

onShow() {

this.getCourseTaskDetail();

},

methods: {

getCourseTaskDetail() {

uniCloud.callFunction({

name:'getCourseTaskDetail',

data: {

\_id:this.\_id

}

}).then((res)=>{

const {result} = res

this.course\_name = result.data[0].course\_name

this.task\_title = result.data[0].task\_title

this.task\_ddl = result.data[0].task\_ddl

this.task\_content = result.data[0].task\_content

console.log(this.course\_name)

console.log(this.task\_title)

console.log(this.task\_ddl)

console.log(this.task\_content)

})

},

finished() {

console.log('任务完成')

uniCloud.callFunction({

name:"finishCourseTask",

data:{

//任务的\_id

\_id:this.\_id,

stu\_id:uni.getStorageSync("globalUser")

},

success(res) {

uni.showToast({

title:'任务完成',

icon: 'none',

duration:1000

})

}

})

}

}

}

</script>

<style>

.page\_name{

color: #808080;

height: 130rpx;

font-size: 40rpx;

font-weight: 550;

margin-left: 280rpx;

margin-top: 25rpx;

}

.task\_title{

margin-left: 30rpx;

font-weight: 600;

font-size: 40rpx;

}

.course\_name{

margin-left: 30rpx;

color: #18BC37;

margin-top: 5px;

}

.task\_content{

margin-top: 50rpx;

margin-left: 50rpx;

}

.end\_time {

margin-left: 300rpx;

margin-top: 30rpx;

}

.finish{

margin-top: 50px;

margin-left: 70rpx;

margin-right: 70rpx;

}

</style>

### 1.3.7添加课程界面

<template>

<view>

<view class="main">

<view class="task" id="task\_course">

<textarea placeholder="请输入课程名" v-model="course\_name" class="text"></textarea>

</view>

<view class="task">

<select-lay :zindex="1211" :value="course\_day" name="name" placeholder="请选择课程所在日" :options="datalist"

@selectitem="selectdayitem" id="select">

</select-lay>

</view>

<view class="start\_endtime" id="starttime">

<select-lay :zindex="1211" :value="start\_time" name="name" placeholder="开始节数" :options="timelist"

@selectitem="selectstarttimeitem" id="select">

</select-lay>

</view>

<view class="start\_endtime" id="endtime">

<select-lay :zindex="1211" :value="end\_time" name="name" placeholder="结束节数" :options="timelist"

@selectitem="selectendtimeitem" id="select">

</select-lay>

</view>

<view class="task" id="task\_title">

<textarea value="" placeholder="请输入课程地点" v-model="course\_add" class="text"></textarea>

</view>

<!-- <view>{{course\_name}}</view>

<view>{{course\_day}}</view>

<view>{{start\_time}}</view>

<view>{{end\_time}}</view>

<view>{{course\_add}}</view> -->

<!-- <view class="task" id="task\_content">

<textarea value="" placeholder="请输入任务内容"></textarea>

</view> -->

</view>

<!-- <view class="title">实时获取输入值：{{course\_name}},{{course\_day}},{{start\_time}},{{end\_time}},{{course\_add}}</view> -->

<view id="submit">

<button @click="add\_course()" class="btn">提交</button>

</view>

</view>

</template>

<script>

export default {

data() {

return {

datalist: [],

timelist: [],

//模拟初始数据

tval: "value2",

// arrays: ['---请选择---','教师','学生','管理员'],

index: 0,

course\_name: '',

course\_day: '',

start\_time: '',

end\_time: '',

course\_add: '',

idnum: '',

coursenum: '',

id: '学生'

}

},

methods: {

selectdayitem(index, item) {

// console.log(item)

if (index >= 0) {

// this.tval = item.value;

this.course\_day = item.value;

} else {

this.tval = ""

}

},

selectstarttimeitem(index, item) {

// console.log(item)

if (index >= 0) {

// this.tval = item.value;

this.start\_time = item.value;

} else {

this.tval = ""

}

},

selectendtimeitem(index, item) {

// console.log(item)

if (index >= 0) {

// this.tval = item.value;

this.end\_time = item.value;

} else {

this.tval = ""

}

},

add\_course: function() {

uniCloud.callFunction({

name: "course\_exist",

data: {

course\_name: this.course\_name,

course\_day: this.course\_day,

start\_time: this.start\_time,

end\_time: this.end\_time,

course\_add: this.course\_add,

idnum: this.idnum,

id: ''

},

}).then((res) => {

const {

result

} = res

// console.log(result.coursenum )

// this.info=result.data

this.coursenum = result.coursenum

console.log(this.coursenum)

})

for (let t = 0; t < 8000; t++) {

if (t == 90) {

if (this.coursenum == 0) {

uniCloud.callFunction({

name: "add\_course",

data: {

course\_name: this.course\_name,

course\_day: this.course\_day,

start\_time: this.start\_time,

end\_time: this.end\_time,

course\_add: this.course\_add,

idnum: this.idnum,

},

// 成功

}).then((res) => {

if (this.id == '学生') {

uniCloud.callFunction({

name: "add\_course\_stu",

data: {

course\_name: this.course\_name,

course\_day: this.course\_day,

start\_time: this.start\_time,

end\_time: this.end\_time,

course\_add: this.course\_add,

idnum: this.idnum,

},

// 成功

success(res) {

console.log("添加课程success");

uni.showToast({

title: '添加课程成功',

icon: 'none',

duration: 1000

});

},

})

}

if (this.id == '教师') {

uniCloud.callFunction({

name: "add\_course\_tea",

data: {

course\_name: this.course\_name,

course\_day: this.course\_day,

start\_time: this.start\_time,

end\_time: this.end\_time,

course\_add: this.course\_add,

idnum: this.idnum,

},

// 成功

success(res) {

console.log("添加课程success");

uni.showToast({

title: '添加课程成功',

icon: 'none',

duration: 1000

});

},

})

}

})

}

}

}

// let j = 0;

// for (let i = 0; i < 98000; i++) {

// if (i == 97900) {

// if (this.id == '学生') {

// uniCloud.callFunction({

// name: "add\_course\_stu",

// data: {

// course\_name: this.course\_name,

// course\_day: this.course\_day,

// start\_time: this.start\_time,

// end\_time: this.end\_time,

// course\_add: this.course\_add,

// idnum: this.idnum,

// },

// // 成功

// success(res) {

// console.log("添加课程success");

// uni.showToast({

// title: '添加课程成功',

// icon: 'none',

// duration: 1000

// });

// },

// })

// }

// if (this.id == '教师') {

// uniCloud.callFunction({

// name: "add\_course\_tea",

// data: {

// course\_name: this.course\_name,

// course\_day: this.course\_day,

// start\_time: this.start\_time,

// end\_time: this.end\_time,

// course\_add: this.course\_add,

// idnum: this.idnum,

// },

// // 成功

// success(res) {

// console.log("添加课程success");

// uni.showToast({

// title: '添加课程成功',

// icon: 'none',

// duration: 1000

// });

// },

// })

// }

// }

// }

// uniCloud.callFunction({

// name:"add\_course",

// data: {

// course\_name: this.course\_name,

// course\_day: this.course\_day,

// start\_time: this.start\_time,

// end\_time: this.end\_time,

// course\_add: this.course\_add,

// idnum:this.idnum,

// },

// // 成功

// success(res) {

// console.log("添加课程success");

// uni.showToast({

// title: '添加课程成功',

// icon: 'none',

// duration: 1000

// });

// },

// //失败

// fail(e) {

// console.log(e);

// }

// })

// uni.switchTab({

// url: '../course/course'

// });

}

},

// onBackPress() {

// uni.navigateTo({

// url: "../course/course",

// });

// return true

// },

onLoad(option) {

this.idnum = uni.getStorageSync("globalUser");

console.log(this.idnum)

// this.id = option.id

},

onReady() {

this.datalist = [{

label: "周一",

value: "周一"

},

{

label: "周二",

value: "周二"

},

{

label: "周三",

value: "周三"

},

{

label: "周四",

value: "周四"

},

{

label: "周五",

value: "周五"

},

{

label: "周六",

value: "周六"

},

{

label: "周日",

value: "周日"

}

];

this.timelist = [{

label: "1",

value: "1"

},

{

label: "2",

value: "2"

},

{

label: "3",

value: "3"

},

{

label: "4",

value: "4"

},

{

label: "5",

value: "5"

},

{

label: "6",

value: "6"

},

{

label: "7",

value: "7"

},

{

label: "8",

value: "8"

},

{

label: "9",

value: "9"

},

{

label: "10",

value: "10"

},

{

label: "11",

value: "11"

},

{

label: "12",

value: "12"

}

];

},

}

</script>

<style>

.main {

margin-top: 80rpx;

}

.task,

.start\_endtime {

height: 70rpx;

/\* border: #006400; \*/

background-color: #F2F2F2;

border-radius: 20rpx;

/\* border-style: groove; \*/

margin: 20rpx;

padding: 10rpx;

}

.start\_endtime {

width: 300rpx;

}

.text {

margin-top: 10rpx;

margin-left: 15rpx;

}

#starttime {

float: left;

}

#endtime {

float: right;

}

#task\_content {

height: 950rpx;

}

#submit {

margin: 40px;

}

.btn {

color: #FFFFFF;

background-color: #7CCD7C;

}

#task\_title {

margin-top: 200rpx;

}

</style>

### 1.3.8修改个人信息界面

<template>

<view>

<view class="main">

<view class="hint">手机号修改：</view>

<view class="task" id="task\_course">

<textarea v-model="tel" :placeholder="this.old\_tel" class="text"></textarea>

</view>

<view class="hint">邮箱修改：</view>

<view class="task" id="task\_title">

<textarea v-model="email" :placeholder="this.old\_email" class="text"></textarea>

</view>

<!-- <view class="task" id="task\_content">

<textarea value="" placeholder="请输入任务内容"></textarea>

</view> -->

</view>

<view id="submit">

<button @click="submit()">提交</button>

</view>

</view>

</template>

<script>

export default {

data() {

return {

idnum: '',

tel: '',

email: '',

old\_tle: '',

old\_email: '',

}

},

onShow() {

this.idnum = uni.getStorageSync("globalUser");

console.log(this.idnum)

this.old()

},

methods: {

submit: function() {

if (this.tel == '' && this.email == '') {

uni.showToast({

title: '请至少修改一项',

icon: 'none',

duration: 1000

});

} else {

if (this.tel.length != 0) {

if (this.tel.length != 11) {

uni.showToast({

title: '请输入正确的手机号格式',

icon: 'none',

duration: 1000

});

} else {

uniCloud.callFunction({

name: 'edit\_I\_tel',

data: {

idnum: this.idnum,

tel: this.tel,

}

})

}

}

if (this.email.length != 0) {

uniCloud.callFunction({

name: 'edit\_I\_email',

data: {

idnum: this.idnum,

email: this.email,

}

})

}

if(this.email.length != 0 && this.tel.length == 11){

uni.showToast({

title: '修改成功',

icon: 'none',

duration: 1000

});

}

}

},

old: function() {

uniCloud.callFunction({

name: 'old\_I\_info',

data: {

idnum: this.idnum

}

}).then((res) => {

const {

result

} = res;

this.old\_tel = result.data[0].tel;

this.old\_email = result.data[0].email;

// this.test = result.data[0].tasktitle;

})

}

}

}

</script>

<style>

.hint {

margin-left: 30rpx;

}

.main {

margin-top: 80rpx;

}

.task {

height: 80rpx;

border-radius: 20rpx;

background-color: #F2F2F2;

margin: 20rpx;

}

#task\_content {

height: 950rpx;

}

#submit {

margin: 40px;

}

.text {

padding: 15rpx;

}

</style>

### 1.3.9添加个人任务界面

<template>

<view>

<view class="main">

<view class="task" id="task\_course">

<textarea v-model="task\_course" placeholder="请输入课程名" class="text"></textarea>

</view>

<view class="task" id="task\_title">

<textarea v-model="task\_title" placeholder="请输入任务" class="text"></textarea>

</view>

<long-date

type="between"

getDayNum="10"

:openStatus="true"

startTime="2021-12-03"

endTime="2022-02-01"

chooesMaxDay="8"

@select="Time">

</long-date>

<view class="task" id="task\_content">

<textarea v-model="task\_content" placeholder="请输入任务内容" class="text"></textarea>

</view>

</view>

<view id="submit">

<button @click="submit\_task()" class="btn">提交</button>

</view>

</view>

</template>

<script>

export default {

data() {

return {

task\_course:'',

task\_title:'',

task\_ddl:'',

task\_content:'',

iffinish:'',

idnum:'',

task\_ddl:"",

}

},

onLoad() {

this.idnum=uni.getStorageSync("globalUser");

// console.log(this.idnum);

},

methods: {

Time(val) {

console.log(val);

this.task\_ddl=val;

},

submit\_task: function(){

uniCloud.callFunction({

name:"add\_personaltask",

data: {

task\_course: this.task\_course,

idnum: this.idnum,

task\_title: this.task\_title,

task\_content: this.task\_content,

task\_ddl:this.task\_ddl,

},

// 成功

success(res) {

console.log("success");

uni.showToast({

title: '添加任务成功',

icon: 'none',

duration: 1000

});

},

//失败

fail(e) {

console.log(e);

}

})

uni.switchTab({

url: '../Task/Task'

});

}

}

}

</script>

<style>

.main{

margin-top: 80rpx;

}

.task{

height: 80rpx;

background-color: #F2F2F2;

border-radius: 20rpx;

margin: 20rpx;

}

.text{

padding-top: 15rpx;

margin-top: 20rpx;

margin-right: 30rpx;

margin-left: 20rpx;

}

#task\_content{

height: 950rpx;

}

#submit{

margin: 40px;

}

.btn{

background-color:#7CCD7C;

color: #FFFFFF;

}

</style>

### 1.3.10个人信息界面

<template>

<view>

<view class="content">

<view class="detail">

身份：

<view class="detailvalue">{{id}}</view>

</view>

<view class="detail">

姓名：

<view class="detailvalue">{{name}}</view>

</view>

<view class="detail">

教工号/学号：

<view class="detailvalue">{{idnum}}</view>

</view>

<view class="detail">

手机号：

<view class="detailvalue">{{tel}}

</view>

</view>

<view class="detail">

邮箱：

<view class="detailvalue">{{email}}</view>

</view>

</view>

<button @click="edit()" class="btn">编辑</button>

</view>

</template>

<script>

export default {

data() {

return {

id: "",

name: "",

idnum: '',

tel: "",

email: "",

}

},

onShow() {

this.idnum = uni.getStorageSync("globalUser");

console.log(this.idnum)

this.get\_stu\_info()

},

methods: {

edit: function() {

uni.navigateTo({

url: '../EditI/EditI'

})

},

get\_stu\_info: function() {

uniCloud.callFunction({

name: "get\_stu\_info",

data: {

idnum: this.idnum,

},

}).then((res) => {

const {

result

} = res

this.id = result.id,

this.name = result.data[0].name,

this.idnum = result.data[0].idnum,

this.tel = result.data[0].tel,

this.email = result.data[0].email

})

}

}

}

</script>

<style>

.content {

margin-top: 30rpx;

}

.detail {

font-size: 40rpx;

margin-top: 20rpx;

margin-right: 30rpx;

margin-left: 30rpx;

padding: 20rpx;

background-color: #F2F2F2;

border-radius: 20rpx;

}

.detailvalue {

text-decoration: underline;

color: #006400;

}

.btn {

width: 400rpx;

display: block;

margin: 40rpx auto;

background-color:#7CCD7C;

color: #FFFFFF;

}

</style>

### 1.3.11登录界面

<template>

<view class="zai-box">

<image src="../../static/logo.jpg" mode='aspectFit' class="zai-logo"></image>

<view class="zai-form">

<!-- <select-lay :zindex="1211" :value="tval" name="name" placeholder="请选择身份" :options="datalist"

@selectitem="selectitem" id="select">

</select-lay> -->

<!-- <view class="title">实时获取输入值：{{inputrole}},{{inputid}},{{inputpwd}}</view> -->

<!-- <view class="role" v-if="inputrole==t">身份：老师</view>

<view class="role" v-if="inputrole==s">身份：学生</view>

<view class="role" v-if="inputrole==m">身份：管理员</view> -->

<!-- <input class="zai-input" @input="onKeyInputidnum" placeholder-class placeholder="请输入教工号/学号" v-model="inputteacherid"/> -->

<input class="zai-input" placeholder-class placeholder="请输入学号" v-model="inputidnum" />

<input class="zai-input" placeholder-class password placeholder="请输入密码" v-model="inputpwd" />

<view class="zai-label-forget">忘记密码？</view>

<button class="zai-btn" @click="login()">立即登录</button>

<navigator url="../register/register" hover-class="none" class="zai-label">还没有账号？点此注册.</navigator>

</view>

</view>

</template>

<script>

import selectlay from "../../components/uni-transition/uni-transition.vue"

export default {

data() {

return {

//模拟数据列表

datalist: [],

//模拟初始数据

tval: "value2",

arrays: ['---请选择---', '教师', '学生', '管理员'],

index: 0,

inputid: '学生',

inputidnum: '',

inputpwd: '',

statu: '',

data: '',

msg: '',

ifpass: ''

}

idnum: ''

},

onLoad() {

this.idnum = uni.getStorageSync("globalUser");

console.log(this.idnum)

},

onReady() {

this.datalist = [{

label: "教师",

value: "教师"

},

{

label: "学生",

value: "学生"

},

{

label: "管理员",

value: "管理员"

}

];

},

methods: {

selectitem(index, item) {

// console.log(item)

if (index >= 0) {

this.tval = item.value;

this.inputid = item.value;

} else {

this.tval = ""

}

},

// onKeyInputidnum: function(event) {

// // this.inputteacherid = event.target.value

// // console.log("test-------",this.inputValue)

// },

login() {

if (this.inputrole == '') {

uni.showToast({

title: '请选择身份',

icon: 'none',

duration: 1000

});

return false;

}

if (this.inputid == '') {

uni.showToast({

title: '请输入教工号/学号',

icon: 'none',

duration: 1000

});

return false;

}

if (this.inputpwd == '') {

uni.showToast({

title: '请输入密码',

icon: 'none',

duration: 1000

});

return false;

}

uniCloud.callFunction({

name: "judge",

data: {

id: this.inputid,

idnum: this.inputidnum,

pwd: this.inputpwd,

}

}).then((res) => {

const {

result

} = res

this.ifpass = result.ifpass

if (this.ifpass == 1) {

console.log("okok")

uni.setStorageSync("globalUser", this.inputidnum);

uni.setStorageSync("globalUserId", this.inputid);

if (this.inputid == "学生") {

uni.switchTab({

url: '../Course/Course?idnum=' + this.inputidnum

});

}

else if(this.inputid == "教师"){

uni.navigateTo({

url: '../TeacherCourse/TeacherCourse?idnum=' + this.inputidnum

});

}

else if(this.inputid == "管理员"){

uni.navigateTo({

url: '../Check/Check'

});

}

} else {

console.log("nonoo"),

uni.showToast({

title: '输入的账号或密码不正确',

icon: 'none',

duration: 1000

});

}

})

},

},

components: {

selectlay

}

}

</script>

<style>

.zai-box {

padding: 0 100upx;

position: relative;

}

.zai-logo {

top: 20rpx;

width: 100%;

height: 310upx;

}

.zai-title {

position: absolute;

top: 0;

line-height: 360upx;

font-size: 68upx;

color: #fff;

text-align: center;

width: 100%;

margin-left: -100upx;

}

.zai-form {

margin-top: 100upx;

}

.zai-input {

background: #e2f5fc;

margin-top: 30upx;

border-radius: 100upx;

padding: 20upx 40upx;

font-size: 36upx;

}

.input-placeholder,

.zai-input {

color: #94afce;

}

.zai-label {

margin-top: 130rpx;

padding: 60upx 0;

text-align: center;

font-size: 30upx;

color: #a7b6d0;

}

.zai-label-forget {

margin-top: -5rpx;

padding: 60upx 0;

text-align: center;

font-size: 30upx;

color: #a7b6d0;

}

.zai-btn {

top: 160rpx;

background: #228B22;

color: #fff;

border: 0;

border-radius: 100upx;

font-size: 36upx;

}

.zai-btn:after {

border: 0;

}

/\*按钮点击效果\*/

.zai-btn.button-hover {

transform: translate(1upx, 1upx);

}

</style>

### 1.3.12注册界面

<template>

<view class="zai-box">

<image src="../../static/logo.jpg" mode='aspectFit' class="zai-logo"></image>

<view class="zai-form">

<select-lay :zindex="1211" :value="tval" name="name" placeholder="请选择身份" :options="datalist"

@selectitem="selectitem" id="select">

</select-lay>

<input class="zai-input" placeholder-class placeholder="请输入教工/学生姓名" v-model="regname" />

<input class="zai-input" placeholder-class placeholder="请输入教工号/学号" v-model="regidnum" />

<input class="zai-input" placeholder-class placeholder="请输入验证手机号" v-model="regtel" />

<input class="zai-input" placeholder-class placeholder="请输入验证邮箱" v-model="regemail" />

<input class="zai-input" placeholder-class password placeholder="请输入密码" v-model="regpwd" />

<input class="zai-input" placeholder-class password placeholder="请再输入一次密码" v-model="regpwd2" />

<!-- <view>"身份："+{{regid}} +"name:"+{{regname}}</view> -->

<button class="zai-btn" @click="reg()">立即注册</button>

<navigator url="../zaizai-login/index" open-type='navigateBack' hover-class="none" class="zai-label">

已有账号，点此去登录.</navigator>

</view>

</view>

</template>

<script>

export default {

data() {

return {

//模拟数据列表

datalist: [],

//模拟初始数据

tval: "value2",

arrays: ['---请选择---', '教师', '学生', '管理员'],

index: 0,

regid: "",

regname: "",

regidnum: "",

regtel: "",

regemail: "",

regpwd: "",

regpwd2: "",

ifpass:'',

}

},

onReady() {

this.datalist = [{

label: "教师",

value: "教师"

},

{

label: "学生",

value: "学生"

},

];

},

methods: {

selectitem(index, item) {

console.log(item)

if (index >= 0) {

this.tval = item.value;

this.regid = item.value;

} else {

this.tval = ""

}

},

reg() {

if (this.regid == '') {

uni.showToast({

title: '选择身份',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regname == '') {

uni.showToast({

title: '请输入姓名',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regtel == '') {

uni.showToast({

title: '请输入手机号',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regtel.length != 11) {

uni.showToast({

title: '请输入正确格式的手机号',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regemail == '') {

uni.showToast({

title: '请输入邮箱',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regpwd == '') {

uni.showToast({

title: '请输入密码',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regpwd2 == '') {

uni.showToast({

title: '请再输入一次密码',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regpwd != this.regpwd2) {

uni.showToast({

title: '两次输入的密码不一致',

icon: 'none',

duration: 1000

});

return false;

}

uniCloud.callFunction({

name:"ifreg",

data:{

id:this.regid,

idnum:this.regidnum,

// pwd:this.inputpwd,

}

}).then((res)=>{

const {result}=res

this.ifpass=result.ifpass

if(this.ifpass==1){

console.log("该教工号/学号已被注册"),

uni.showToast({

title: '该教工号/学号已被注册',

icon: 'none',

duration: 1000

});

}

})

if(this.ifpass!=1){

uniCloud.callFunction({

name: 'reg', // 云函数名字

// 传输数据

data: {

id: this.regid,

name: this.regname,

idnum: this.regidnum,

tel: this.regtel,

email: this.regemail,

pwd:this.regpwd,

},

// 成功

success(res) {

console.log("success");

uni.showToast({

title: '注册申请发送成功',

icon: 'none',

duration: 1000

});

uni.navigateTo({

url: "../Login/Login"

})

},

//失败

fail(e) {

console.log(e);

}

})

}

// uniCloud.callFunction({

// name: 'reg', // 云函数名字

// // 传输数据

// data: {

// id: this.regid,

// name: this.regname,

// idnum: this.regidnum,

// tel: this.regtel,

// email: this.regemail,

// pwd:this.regpwd,

// },

// // 成功

// success(res) {

// console.log("success");

// uni.showToast({

// title: '注册申请发送成功',

// icon: 'none',

// duration: 1000

// });

// uni.navigateTo({

// url: "../login/login"

// })

// },

// //失败

// fail(e) {

// console.log(e);

// }

// })

}

},

}

</script>

<style>

.zai-box {

padding: 0 100upx;

position: relative;

}

.zai-logo {

top: 20rpx;

width: 100%;

height: 310upx;

}

.zai-title {

position: absolute;

top: 0;

line-height: 360upx;

font-size: 68upx;

color: #fff;

text-align: center;

width: 100%;

margin-left: -100upx;

}

.zai-form {

margin-top: 60upx;

}

.zai-input {

background: #e2f5fc;

margin-top: 30upx;

border-radius: 100upx;

padding: 20upx 40upx;

font-size: 36upx;

}

.input-placeholder,

.zai-input {

color: #94afce;

}

.zai-label {

margin-top: 10rpx;

padding: 60upx 0;

text-align: center;

font-size: 30upx;

color: #a7b6d0;

}

.zai-btn {

top: 10rpx;

background: #228B22;

color: #fff;

border: 0;

border-radius: 100upx;

font-size: 36upx;

margin-top: 60upx;

}

.zai-btn:after {

border: 0;

}

/\*按钮点击效果\*/

.zai-btn.button-hover {

transform: translate(1upx, 1upx);

}

</style>

### 1.3.13个人任务列表

<template>

<view>

<view>

<uni-list>

<view v-for="item in tasklist" @click="todetail(item.\_id)" @mouseover="confirm\_task\_id(item.\_id)"

@touchstart="touchStart" @touchend="touchEnd">

<uni-list-item :title="'@'+item.task\_course+'-'+item.task\_title" :note="item.task\_ddl+'-'+item.task\_content" :rightText="item.iffinish">

</uni-list-item>

<!-- <button @click="confirm()">完成</button> -->

</view>

</uni-list>

</view>

</view>

</template>

<script>

export default {

data() {

return {

list: [{

index: 1,

data: 1

},

{

index: 2,

data: 2,

},

{

index: 3,

data: 3,

},

{

index: 4,

data: 4,

},

{

index: 5,

data: 5,

}

],

idnum: '',

tasklist: [],

test: "",

startX: 0,

\_id: "",

}

},

onLoad() {

this.idnum = uni.getStorageSync("globalUser");

// console.log(this.idnum);

this.gettasklist();

},

onPullDownRefresh() {

//监听下拉刷新动作的执行方法，每次手动下拉刷新都会执行一次

console.log('refresh');

this.gettasklist();

setTimeout(function() {

uni.stopPullDownRefresh(); //停止下拉刷新动画

}, 1000);

},

methods: {

/\*\*

\* 触摸开始

\* @param {Object} e

\*/

touchStart: function(e) {

if (e.touches.length == 1) {

//设置触摸起始点水平方向位置

this.startX = e.touches[0].clientX;

}

},

/\*\*

\* 触摸结束

\* @param {Object} e

\*/

touchEnd: function(e) {

if (e.changedTouches.length == 1) {

//手指移动结束后水平位置

var endX = e.changedTouches[0].clientX;

let diff = endX - this.startX;

if (Math.abs(diff) > 20) {

if (diff > 0) {

console.log("右滑...");

} else {

console.log("左滑...");

uni.showModal({

title: '提示',

content: '确认完成任务了吗？',

cancelText: "取消", // 取消按钮的文字

confirmText: "确认", // 确认按钮的文字

showCancel: true, // 是否显示取消按钮，默认为 true

confirmColor: '#f55850',

cancelColor: '#39B54A',

success: (res) => {

if (res.confirm) {

console.log('comfirm') //点击确定之后执行的代码

uniCloud.callFunction({

name: 'finish\_task', // 云函数名字

// 传输数据

data: {

task\_id: this.\_id,

},

// 成功

success(res) {

console.log("successllalalalalala");

uni.showToast({

title: '任务完成！',

icon: 'none',

duration: 1000

});

},

//失败

})

} else {

console.log('cancel') //点击取消之后执行的代码

}

}

})

}

}

}

},

confirm\_task\_id: function(id) {

// console.log(data)

// console.log(id);

// this.$emit("click",event);

this.\_id = id;

console.log("hahattt")

console.log(this.\_id)

},

todetail: function(id) {

// console.log(data)

// console.log(id);

// this.$emit("click",event);

uni.navigateTo({

url: '../TaskDetail/TaskDetail?taskid=' + id

})

},

gettasklist: function() {

// console.log(data)

uniCloud.callFunction({

name: 'taskshow',

data: {

id:'学生'

idnum: this.idnum,

// iffinish: '未完成',

ing:1

}

}).then((res) => {

const {

result

} = res;

this.tasklist = result.data;

// this.test = result.data[0].tasktitle;

})

},

confirm: function() {

console.log("1")

}

},

onNavigationBarButtonTap: function(e) {

console.log(e.text); //提交

console.log(e.fontSize); //16px

if (e.text == "+") {

uni.navigateTo({

url: '../EditTask/EditTask'

})

}

if (e.text == "历史") {

uni.navigateTo({

url: '../TaskHis/TaskHis'

})

}

}

}

</script>

<style>

</style>

### 1.3.14个人任务详情

<template>

<view>

<view class="main" >

<view class="task" id="task\_course">

<textarea :value="this.task\_course" placeholder="请输入指定任务的课程" class="text"></textarea>

</view>

<view class="task" id="task\_ddl">

<textarea :value="this.task\_ddl" placeholder="请输入截止日期" class="text"></textarea>

</view>

<view class="task" id="task\_title">

<textarea :value="this.task\_title" placeholder="请输入任务标题" class="text"></textarea>

</view>

<view class="task" id="task\_content">

<textarea :value="this.task\_content" placeholder="请输入任务内容" class="text"></textarea>

</view>

<view id="submit">

<button class="btn">提交</button>

</view>

</view>

</view>

</template>

<script>

export default {

data() {

return {

// list:[

// {

// id:1,

// course:软件工程,

// title:课程预备项目,

// content:完成预备作业,

// },

// {

// id:2,

// course:高级程序设计,

// title:实验3,

// content:完成实验3,

// },

// {

// id:3,

// course:项目管理,

// title:大作业1,

// content:完成大作业,

// },

// {

// id:4,

// course:项目实践,

// title:课程实践项目,

// content:完成大作业,

// },

// ],

taskid:"",

task\_course:"",

task\_ddl:"",

task\_title:"",

task\_content:""

}

},

onLoad(option) {

this.taskid = option.taskid;

// console.log(option)

console.log(option.taskid)

this.gettaskdetail();

},

methods: {

gettaskdetail:function(){

uniCloud.callFunction({

name: 'taskdetailshow',

data: {

taskid: this.taskid,

}

}).then((res) => {

const {

result

} = res

console.log("success" + result)

// this.info=result.data

this.task\_course = result.data[0].task\_course

this.task\_ddl = result.data[0].task\_ddl

this.task\_title = result.data[0].task\_title

this.task\_content = result.data[0].task\_content

console.log(this.id)

// this.name=result.data.name

// this.tel=result.data.tel

// this.email=result.data.email

})

}

}

}

</script>

<style>

.text{

margin-top: 20px;

padding-top: 8px;

padding-left: 5px;

}

.main{

margin-top: 80rpx;

}

.task{

height: 80rpx;

background-color: #F2F2F2;

border-radius: 20rpx;

margin: 20rpx;

}

#task\_content{

height: 900rpx;

}

#submit{

margin: 40px;

}

.btn{

background-color:#7CCD7C;

color: #FFFFFF;

}

</style>

# 2教师端

## 2.1云函数

### 2.1.1添加课程

'use strict';

const db = uniCloud.database()

exports.main = async (event, context) => {

//event为客户端上传的参数

console.log('event : ', event)

/\* 最后都会在coourse\_stu表中填加一条记录，表明这个学生与课程有联系\*/

let which\_course = await db.collection('course').where({

/\* 获取刚添加的课程id号 \*/

course\_day: event.course\_day,

start\_time: event.start\_time,

end\_time: event.end\_time,

course\_add: event.course\_add,

}).get()

for (let i = 0; i < 3000; i++) {

if (i == 2900) {

let result = await db.collection('course\_tea').add({

course\_id: which\_course.data[0].\_id,

course\_name: event.course\_name,

teacher\_idnum: event.idnum,

course\_day: event.course\_day,

start\_time: event.start\_time,

end\_time: event.end\_time,

})

}

}

//返回数据给客户端

return {

courseid: which\_course.data[0].\_id,

msg: '添加课程学生的联系'

}

};

### 2.1.2添加闹钟

'use strict';

const db = uniCloud.database();

const collection = db.collection("alarm");

exports.main = async (event, context) => {

//该闹钟是否存在

let res1 = await collection.where({

alarm\_title:event.alarm\_title,

alarm\_content:event.alarm\_content,

dateTime : new Date(+new Date(event.dateTime) + 8 \* 3600 \*1000).toISOString().replace(/T/g,' ').replace(/\.[\d]{3}Z/,''),

user\_id:event.user\_id

}).get()

console.log(res1)

if (res1.affectedDocs != 0) {

return "该闹钟已存在";

}

let res2 = await collection.add({

alarm\_title:event.alarm\_title,

alarm\_content:event.alarm\_content,

dateTime : new Date(+new Date(event.dateTime) + 8 \* 3600 \*1000).toISOString().replace(/T/g,' ').replace(/\.[\d]{3}Z/,''),

user\_id:event.user\_id

})

return res2;

};

### 2.1.3课程是否存在

'use strict';

const db = uniCloud.database()

exports.main = async (event, context) => {

//event为客户端上传的参数

console.log('event : ', event)

let course\_exist = await db.collection('course').where({

/\* 查看当前课程表中是否存在该课程 \*/

// idnum: event.idnum,

course\_name: event.course\_name,

course\_day: event.course\_day,

start\_time: event.start\_time,

end\_time: event.end\_time,

course\_add: event.course\_add,

}).count()

//返回数据给客户端

return {

code:200,

msg:'获取审核表成功',

coursenum: course\_exist.total

}

};

### 2.1.4删除闹钟

'use strict';

const db = uniCloud.database();

const collection = db.collection("alarm");

exports.main = async (event, context) => {

let res = await collection.where({

\_id:event.\_id

}).remove()

return res

};

### 2.1.5编辑邮箱

'use strict';

const db = uniCloud.database() //对数据库的对象获取；

exports.main = async (event, context) => {

let collection = db.collection('student')

let res = await collection.where({

idnum: event.idnum,

}).update({

email: event.email,

})

return {

code: 200,

msg: '邮箱修改成功',

data: res

}

}

### 2.1.6编辑电话

'use strict';

const db = uniCloud.database() //对数据库的对象获取；

exports.main = async (event, context) => {

let collection = db.collection('student')

let res = await collection.where({

idnum:event.idnum,

}).update({

tel: event.tel,

})

return {

code: 200,

msg: '查询成功',

data: res

}

}

### 2.1.7课程任务是否存在

'use strict';

const db = uniCloud.database()

exports.main = async (event, context) => {

//event为客户端上传的参数

console.log('event : ', event)

let course\_count = await db.collection('course\_tea').where({

id: event.id,

idnum: event.idnum,

pwd: event.pwd,

}).count()

if (course\_count.total == 0) {

return {

msg: "此课程不存在",

course\_count: 0

}

} else {

//返回数据给客户端

return {

code: 200,

msg: '获取审核表成功',

// data: checktable.data

data: res.data,

course: course\_count

}

}

};

### 2.1.8课程是否存在

'use strict';

const db = uniCloud.database()

exports.main = async (event, context) => {

//event为客户端上传的参数

console.log('event : ', event)

/\* 判断该课程是否和该教师有关联 \*/

let course\_count = await db.collection('course\_tea').where({

course\_day: event.course\_day,

start\_time: event.start\_time,

end\_time: event.end\_time,

}).count()

/\* 该课程是否和该教师无关联 \*/

if (course\_count.total == 0) {

return {

msg: "此教师的该课程不存在",

course\_count: 0

}

} else {

for (let i = 0; i < 2000; i++) {

if (i == 1990) {

/\* 获取该课程的编号 \*/

let course = await db.collection('course\_tea').where({

course\_day: event.course\_day,

start\_time: event.start\_time,

end\_time: event.end\_time,

// task\_ddl:event.task\_ddl,

}).get()

while (i < 9000) {

i++;

if (i == 8000) {

let result = await db.collection('course\_task').add({

course\_name: event.task\_course,

teacher\_idnum: event.teacher\_idnum,

course\_id: course.data[0].course\_id,

task\_title: event.task\_title,

task\_ddl: event.task\_ddl,

task\_content: event.task\_content,

})

// for (let j = 0; j < 900000; j++) {

// if (j == 899999) {

// let task = await db.collection('course\_task').where({

// course\_name: event.task\_course,

// teacher\_idnum: event.teacher\_idnum,

// course\_id: course.data[0].course\_id,

// task\_title: event.task\_title,

// task\_ddl: event.task\_ddl,

// task\_content: event.task\_content,

// }).get()

return {

code: 200,

msg: '添加任务成功',

// data: checktable.data,

course\_count: 1,

course\_id: course.data[0].course\_id,

// task\_id: task.data[0].\_id

}

// }

// }

//返回数据给客户端

}

}

}

}

}

};

### 2.1.9获取完成率

'use strict';

const db = uniCloud.database()

exports.main = async (event, context) => {

//event为客户端上传的参数

console.log('event : ', event)

let finish\_count = await db.collection('course\_task\_stu').where({

task\_id: event.task\_id,

iffinish: "1"

}).count()

let all\_count = await db.collection('course\_task\_stu').where({

task\_id: event.task\_id,

}).count()

//返回数据给客户端

return {

code: 200,

msg: '获取完成率成功',

// data: checktable.data

// finish\_count:finish\_count.total,

// all\_count:all\_count.total,

data:finish\_count.total/all\_count.total,

}

};

### 2.1.10获取教师课程任务列表

'use strict';

const db = uniCloud.database()

exports.main = async (event, context) => {

//event为客户端上传的参数

console.log('event : ', event)

let checktable = await db.collection('course\_task').where({

teacher\_idnum: event.idnum,

// iffinish: event.iffinish ,

// ing: event.ing

}).get()

//返回数据给客户端

return {

code:200,

msg:'获取审核表成功',

data:checktable.data

}

};

### 2.1.2添加课程

## 2.2组件

### 2.2.1 进度

<template>

<view

class="circle-progress"

:style="{

width: widthPx + 'px',

height: widthPx + 'px',

backgroundColor: bgColor

}"

>

<!-- 有的不支持canvas-id属性，必须用id属性 -->

<canvas

class="canvas-bg"

:canvas-id="1212"

:id="1212"

:style="{

width: widthPx + 'px',

height: widthPx + 'px'

}"

></canvas>

<canvas

class="canvas"

:canvas-id="elId"

:id="elId"

:style="{

width: widthPx + 'px',

height: widthPx + 'px'

}"

></canvas>

<slot></slot>

</view>

</template>

<script>

export default {

name: 'circle-progress',

props: {

// 圆环进度百分比值

percent: {

type: Number,

default: 0,

// 值在0到100之间

validator: val => {

return val >= 0 && val <= 100;

}

},

// 圆环底色（灰色的圆环）

inactiveColor: {

type: String,

default: '#ececec'

},

// 圆环激活部分的颜色

activeColor: {

type: String,

default: '#009dff'

},

// 圆环线条的宽度，单位rpx

borderWidth: {

type: [Number, String],

default: 14

},

// 整个圆形的宽度，单位rpx

width: {

type: [Number, String],

default: 200

},

// 整个圆环执行一圈的时间，单位ms

duration: {

type: [Number, String],

default: 1500

},

// 圆环进度区域的背景色

bgColor: {

type: String,

default: '#ffffff'

}

},

data() {

return {

canvasId: this.randomId(), //一个页面多个圆形进度

elId: this.randomId(),

widthPx: uni.upx2px(this.width), // 转成px后的整个组件的背景宽度

borderWidthPx: uni.upx2px(this.borderWidth), // 转成px后的圆环的宽度

startAngle: -Math.PI / 2, // canvas画圆的起始角度，默认为3点钟方向，定位到12点钟方向

progressContext: null, // 活动圆的canvas上下文

newPercent: 0, // 当动态修改进度值的时候，保存进度值的变化前后值，用于比较用

oldPercent: 0 // 当动态修改进度值的时候，保存进度值的变化前后值，用于比较用

};

},

watch: {

percent(nVal, oVal = 0) {

if (nVal > 100) nVal = 100;

if (nVal < 0) oVal = 0;

this.newPercent = nVal;

this.oldPercent = oVal;

setTimeout(() => {

this.drawCircleByProgress(oVal);

}, 50);

}

},

created() {

// 赋值，用于加载后第一个画圆使用

this.newPercent = this.percent;

this.oldPercent = 0;

},

mounted() {

this.drawProgressBg();

this.drawCircleByProgress(this.oldPercent);

},

methods: {

//一个页面多个progress时ID需不同

randomId(){

return 'progressId'+parseInt(Math.random()\*1000000)

},

drawProgressBg() {

let ctx = uni.createCanvasContext(this.canvasId, this);

ctx.setLineWidth(this.borderWidthPx); // 设置圆环宽度

ctx.setStrokeStyle(this.inactiveColor); // 线条颜色

ctx.beginPath(); // 开始描绘路径

let radius = this.widthPx / 2;

ctx.arc(radius, radius, radius - this.borderWidthPx, 0, 2 \* Math.PI, false);

ctx.stroke(); // 对路径进行描绘

ctx.draw();

},

drawCircleByProgress(progress) {

let ctx = this.progressContext;

if (!ctx) {

ctx = uni.createCanvasContext(this.elId, this);

this.progressContext = ctx;

}

// 表示进度的两端为圆形

ctx.setLineCap('round');

// 设置线条的宽度和颜色

ctx.setLineWidth(this.borderWidthPx);

ctx.setStrokeStyle(this.activeColor);

// 计算过渡时间

let time = Math.floor(this.duration / 200);

let endAngle = ((2 \* Math.PI) / 100) \* progress + this.startAngle;

ctx.beginPath();

// 半径为整个canvas宽度的一半

let radius = this.widthPx / 2;

ctx.arc(radius, radius, radius - this.borderWidthPx, this.startAngle, endAngle, false);

ctx.stroke();

ctx.draw();

// 增大了百分比

if (this.newPercent > this.oldPercent) {

progress++;

if (progress > this.newPercent) return;

} else {

// 减少百分比

progress--;

if (progress < this.newPercent) return;

}

setTimeout(() => {

// 定时器，为了让进度条有动画效果

this.drawCircleByProgress(progress);

}, time);

}

}

};

</script>

<style scoped>

.circle-progress {

position: relative;

display: flex;

align-items: center;

justify-content: center;

}

.canvas-bg {

position: absolute;

}

.canvas {

position: absolute;

}

</style>

### 2.2.2 进度圈

<template>

<view class="cmd-circle">

<canvas :canvas-id="cid" :style="calCircleStyle"></canvas>

</view>

</template>

<script>

/\*\*

\* 进度圈组件

\* @description 用圈显示一个操作完成的百分比时，为用户显示该操作的当前进度和状态。

\* @tutorial https://ext.dcloud.net.cn/plugin?id=259

\* @property {String} cid 画布编号 - 默认defaultCanvas

\* @property {String} type 进度圈类型 - 圆圈形：circle、仪表盘：dashboard，默认圆圈形：circle

\* @property {Number} percent 进度圈百分比值 - 显示范围0-100 ，可能数比较大就需要自己转成百分比的值

\* @property {Boolean} show-info 进度圈进度状态信息 - 显示进度数值或状态图标，默认true

\* @property {String} font-color 进度圈文字信息颜色

\* @property {String} font-size 进度圈文字信息大小 - 默认：14

\* @property {String} status 进度圈状态 - 正常：normal、完成：success、失败：exception，默认正常：normal

\* @property {Number} stroke-width 进度圈线条宽度 - 建议在条线的宽度范围：1-50，与进度条显示宽度有关，默认：6

\* @property {String} stroke-color 进度圈的颜色 - 设置后status状态无效

\* @property {String} stroke-background 进度圈的底圈颜色 - 默认：#eeeeee

\* @property {String} stroke-shape 进度圈两端的形状 - 圆：round、方直角：square，默认圆：round

\* @property {Number} width 进度圈布宽度 - 默认80

\* @property {String} gap-degree 进度圈形缺口角度 - 可取值 0 ~ 360，仅支持类型：circle

\* @property {String} gap-position 进度圈形缺口位置 - 可取值'top', 'bottom', 'left', 'right'，仅支持类型：circle

\* @example <cmd-circle id="circle1" type="circle" :percent="75"></cmd-circle>

\*/

export default {

name: "cmd-circle",

props: {

// 画布编号 默认defaultCanvas

cid: {

type: String,

default: "defaultCanvas"

},

// 圈类型默认：circle，可选 circle dashboard

type: {

type: String,

validator: val => {

return ['circle', 'dashboard'].includes(val);

},

default: 'circle'

},

// 圈进度百分比值

percent: {

type: Number,

validator: val => {

return val >= 0 && val <= 100;

},

default: 0

},

// 圈是否显示进度数值或状态图标

showInfo: {

type: Boolean,

default: true

},

// 圈文字信息颜色

fontColor: {

type: String,

default: "#595959"

},

// 圈文字信息大小 默认14

fontSize: {

type: Number,

default: 14

},

// 圈进度状态，可选：normal success exception

status: {

type: String,

validator: val => {

return ['normal', 'success', 'exception'].includes(val);

},

default: 'normal'

},

// 圈线条宽度1-50，与width有关

strokeWidth: {

type: Number,

default: 6

},

// 圈的颜色，设置后status状态无效

strokeColor: {

type: String,

default: ''

},

// 圈的底圈颜色 默认：#eeeeee

strokeBackground: {

type: String,

default: '#eeeeee'

},

// 圈两端的形状 可选：'round', 'square'

strokeShape: {

type: String,

validator: val => {

return ['round', 'square'].includes(val);

},

default: 'round'

},

// 圈画布宽度

width: {

type: Number,

default: 80

},

// 圈缺口角度，可取值 0 ~ 360，仅支持类型：circle

gapDegree: {

type: Number,

validator: val => {

return val >= 0 && val <= 360;

},

default: 360

},

// 圈缺口开始位置,可取值'top', 'bottom', 'left', 'right'，仅支持类型：circle

gapPosition: {

type: String,

validator: val => {

return ['top', 'bottom', 'left', 'right'].includes(val);

},

default: 'top'

}

},

data() {

return {

// 画布实例

ctx: {},

// 圈半径

width2px: ""

}

},

computed: {

// 计算设置圈样式

calCircleStyle() {

return `width: ${this.width}px;

height: ${this.width}px;`

},

// 计算圈状态

calStatus() {

let status = {}

switch (this.status) {

case 'normal':

status = {

color: "#1890ff",

value: 1

};

break;

case 'success':

status = {

color: "#52c41a",

value: 2

};

break;

case 'exception':

status = {

color: "#f5222d",

value: 3

};

break;

}

return status

},

// 计算圈缺口角度

calGapDegree() {

return this.gapDegree <= 0 ? 360 : this.gapDegree

},

// 计算圈缺口位置

calGapPosition() {

let gapPosition = 0

switch (this.gapPosition) {

case 'bottom':

gapPosition = 90;

break;

case 'left':

gapPosition = 180;

break;

case 'top':

gapPosition = 270;

break;

case 'right':

gapPosition = 360;

break;

}

return gapPosition

},

},

watch: {

// 监听百分比值改变

percent(val) {

this.drawStroke(val);

}

},

mounted() {

// 创建画布实例

this.ctx = uni.createCanvasContext(this.cid, this)

// upx转px 圈半径大小

this.width2px = uni.upx2px(this.width)

// 绘制初始

this.$nextTick(() => {

this.drawStroke(this.percent)

})

},

methods: {

// 绘制圈

drawStroke(percent) {

percent = percent >= 100 ? 100 : percent < 0 ? 0 : percent

// 圈条进度色

let color = this.strokeColor || this.calStatus.color

// 是否圈中心显示信息

if (this.showInfo) {

switch (this.calStatus.value) {

case 1:

if (percent >= 100) {

// 设置打勾

this.drawSuccess()

percent = 100

color = "#52c41a"

} else {

// 设置字体

this.drawText(percent)

}

break;

case 2:

// 设置打勾

this.drawSuccess()

percent = 100

color = "#52c41a"

break;

case 3:

// 设置打叉

this.drawException()

percent = 0

color = "#f5222d"

break;

default:

break;

}

}

// 缺口

let gapPosition = this.calGapPosition

let gapDegree = this.calGapDegree

// 仪表固定

if (this.type === "dashboard") {

gapPosition = 135

gapDegree = 270

}

// 圈型条宽

this.ctx.setLineCap(this.strokeShape)

this.ctx.setLineWidth(this.strokeWidth)

// 位置原点

this.ctx.translate(this.width2px, this.width2px)

// 缺口方向

this.ctx.rotate(gapPosition \* Math.PI / 180)

// 圈底

this.ctx.beginPath()

this.ctx.arc(0, 0, this.width2px - this.strokeWidth, 0, gapDegree \* Math.PI / 180)

this.ctx.setStrokeStyle(this.strokeBackground)

this.ctx.stroke()

// 圈进度

this.ctx.beginPath()

this.ctx.arc(0, 0, this.width2px - this.strokeWidth, 0, percent \* gapDegree \* Math.PI / 18000)

this.ctx.setStrokeStyle(color)

this.ctx.stroke()

// 绘制

this.ctx.draw()

},

// 绘制文字格式

drawText(percent) {

this.ctx.beginPath()

this.ctx.setFontSize(this.fontSize)

this.ctx.setFillStyle(this.fontColor)

this.ctx.setTextAlign('center')

this.ctx.fillText(`${percent}%`, this.width2px, this.width2px + this.fontSize / 2)

this.ctx.stroke()

},

// 绘制成功打勾

drawSuccess() {

let x = this.width2px - this.fontSize / 2

let y = this.width2px + this.fontSize / 2

this.ctx.beginPath()

this.ctx.setLineCap('round')

this.ctx.setLineWidth(this.fontSize / 4)

this.ctx.moveTo(this.width2px, y)

this.ctx.lineTo(y, x)

this.ctx.moveTo(this.width2px, y)

this.ctx.lineTo(x, this.width2px)

this.ctx.setStrokeStyle("#52c41a")

this.ctx.stroke()

},

// 绘制异常打叉

drawException() {

let x = this.width2px - this.fontSize / 2

let y = this.width2px + this.fontSize / 2

this.ctx.beginPath()

this.ctx.setLineCap('round')

this.ctx.setLineWidth(this.fontSize / 4)

this.ctx.moveTo(x, x)

this.ctx.lineTo(y, y)

this.ctx.moveTo(y, x)

this.ctx.lineTo(x, y)

this.ctx.setStrokeStyle("#f5222d")

this.ctx.stroke()

}

}

};

</script>

<style>

.cmd-circle {

display: inline-block;

box-sizing: border-box;

list-style: none;

margin: 0;

padding: 0;

}

</style>

### 2.2.3课程表

<template>

<view class="timetable">

<view class="header"> <!-- 显示周几-->

<view class="header-item" v-for="(item,index) in week" :key="item" :style="{ color: todayWeekIndex === index ? '#4070FF' : 'unset' }">{{ item }}</view>

</view>

<view class="main">

<view class="row" v-for="(item,index) in timetableType" :key="index"><!-- 显示节数及每节课时间 -->

<view class="time-item" >

<view class="index">{{ item.index }}</view><!-- 显示节数 -->

<view class="time">{{ item.name }}</view><!-- 显示每节课的上课时间 -->

</view>

</view>

<view class="course-container">

<view class="week" v-for="(week, weekIndex) in courseData" :key="weekIndex">

<view class="courseList" v-for="(course, courseIndex) in week" :key="courseIndex">

<view @click="handleCourseClick(course, weekIndex, courseIndex)" class="course" :style="{ height: (course.length \* 79) + 'px', background: course.backgroundColor}" v-if="course.length > 0">{{ course.name }}</view>

<!-- <view @click="testCourseClick()"class="course" :style="{ height: (course.length \* 79) + 'px', background: course.backgroundColor}" v-if="course.length > 0">{{ course.name }}</view> -->

</view>

</view>

</view>

</view>

</view>

</template>

<script>

export default {

name: 'Timetable',

props: {

timetableType: {

type: Array,

default: () => {

return [

{ index: '1', name: '08:00\n08:45' },

{ index: '2', name: '08:50\n09:35' },

{ index: '3', name: '09:50\n10:35' },

{ index: '4', name: '10:40\n11:25' },

{ index: '5', name: '11:30\n12:15' },

{ index: '6', name: '13:30\n14:15' },

{ index: '7', name: '14:20\n15:05' },

{ index: '8', name: '15:20\n16:05' },

{ index: '9', name: '16:10\n16:55' },

{ index: '10', name: '18:30\n19:15' },

{ index: '11', name: '19:20\n20:05' },

{ index: '12', name: '20:10\n20:55' }

]

}

},

week: {

type: Array,

default: () => {

return ['一', '二', '三', '四', '五', '六', '日']

}

},

timetables: {

type: Array,

default: () => {

return []

}

},

palette: {

type: Array,

default: () => {

return []

}

}

},

data () {

return {

allPalette: [...this.palette, '#f05261', '#48a8e4', '#ffd061', '#52db9a', '#70d3e6', '#52db9a', '#3f51b5', '#f3d147', '#4adbc3', '#673ab7', '#f3db49', '#76bfcd', '#b495e1', '#ff9800', '#8bc34a'],

course\_id:""

}

},

computed: {

courseData () {

// 为数据标记背景颜色的函数

let paletteIndex = 0

const getBackgroundColor = () => {

const backgroundColor = this.allPalette[paletteIndex]

paletteIndex++

if (paletteIndex >= this.allPalette.length) {

paletteIndex = 0

}

return backgroundColor

}

// 合并

const listMerge = []

this.timetables.forEach(function (list, i) {

if (!listMerge[i]) {

listMerge[i] = []

}

list.forEach(function (item, index) {

if (!index) {

return listMerge[i].push({ name: item, length: 1, backgroundColor: item === '' ? 'none' : getBackgroundColor() })

}

if (item === (listMerge[i][index - 1] || {}).name && item) {

const sameIndex = (listMerge[i][index - 1] || {}).sameIndex

if (sameIndex || sameIndex === 0) {

listMerge[i][sameIndex].length++

return listMerge[i].push({ name: item, length: 0, sameIndex: sameIndex })

}

listMerge[i][index - 1].length++

return listMerge[i].push({ name: item, length: 0, sameIndex: index - 1 })

} else {

return listMerge[i].push({ name: item, length: 1, backgroundColor: item === '' ? 'none' : getBackgroundColor() })

}

})

})

return listMerge

},

todayWeekIndex () {

let weekIndex = new Date().getDay() - 1

if (weekIndex === -1) {

weekIndex = 6

}

return weekIndex

}

},

methods: {

handleCourseClick:function(course, weekIndex, courseIndex) {

const data = {

index: courseIndex + 1,

length: course.length,

week: this.week[weekIndex],

weekIndex: weekIndex,

name: course.name

}

console.log(`星期${data.week}; 第${data.index}节课; 课程名:${data.name}; 课节:${data.length}`)

console.log(data)

//根据周几第几节课和地点，查该课程的\_id

let addr=data.name.split('@')

let week = "周" + data.week

let index = data.index

let add = addr[1]

uniCloud.callFunction({

name:"searchCourseByTimeAddr",

data:{

course\_day:week,

start\_time:index+'',

//start\_time:"3",

course\_add:add

}

}).then((res)=>{

const{result} = res

console.log(result)

this.course\_id = result.data[0].\_id

console.log(this.course\_id)

uni.navigateTo({

url:'/pages/CourseTask/CourseTask?\_id=' + this.course\_id

// url:'/pages/course\_task/course\_task?\_id=61b6079bb804f80001909fcc'// + this.course\_id

})

})

this.$emit('courseClick', data)//子组件向父组件传值,,第一个值为父组件定义函数，第二个是需要传递的参数

},

// toDetail() {

// this.$emit("click",event);

// console.log("点击了！")

// console.log(this.course\_id)

// uni.navigateTo({

// url:'../course\_task/course\_task?\_id='+this.course\_id

// })

// },

testCourseClick:function(){

console.log("test")

},

toTask:function(){

}

}

}

</script>

<style scoped lang="scss">

.timetable{

background: white;

border: 1px solid #E4E7ED;

border-radius: 8rpx;

.header{

padding-left: 88rpx;

height: 56rpx;

display: flex;

align-items: center;

background: #F5F7FA;

.header-item{

flex: 1;

text-align: center;

}

}

.main{

position: relative;

.row{

height: 85px;

position: relative;

&:after{

content: '';

height: 0;

width: 100%;

position: absolute;

bottom: 0;

left: 0;

border-bottom: 1rpx dashed #E4E7ED;

}

.time-item{

height: 100%;

width: 88rpx;

text-align: center;

background: #F5F7FA;

.index{

color: #909399;

padding-bottom: 8rpx;

padding-top: 16rpx;

}

}

}

.course-container{

position: absolute;

top: 0;

left: 90rpx;

width: calc(100% - 280rpx);

// width: 88rpx;

height: 100%;

display: flex;

.week{

flex: 1;

width: 0;

flex-grow: 1;

display: flex;

flex-direction: column;

.courseList{

word-break: break-all;

color: white;

overflow: hidden;

.course{

width: 70rpx;

padding: 8rpx;

left:40rpx;

// border-radius: 16rpx;

border-radius: 16rpx;

text-align: center;

}

}

}

}

}

}

</style>

## 2.3页面

### 2.3.1添加闹钟

<template>

<view class = "all">

<view class="main" >

<view class="alarm" id="alarm\_title">

<input class = "alarm\_input" placeholder="闹钟标题" v-model="alarm\_title"></input>

<!-- <input v-model="inputSerialNum" style="padding: 10rpx;font-size: 30rpx;" class="uni-input" focus placeholder="请输入值" /> -->

</view>

<view class="alarm" id="alarm\_content">

<textarea class = "alarm\_input" placeholder="请输入闹钟内容" v-model="alarm\_content"></textarea>

</view>

<long-date

type="day"

getDayNum="10"

:openStatus="true"

startTime="2021-10-01"

endTime="2022-06-01"

chooesMaxDay="8"

@select="Time">

</long-date>

<view id="submit">

<button type="primary" @click="addAlarm()">添加</button>

</view>

</view>

</view>

</template>

<script>

//导入子组件

import longDate from "@/components/long-date/long-date.vue"

export default {

components: {

longDate

},

data() {

return {

alarm\_title:"",

alarm\_content:"",

dateTime : '',

user\_id:"31901245"

}

},

methods: {

Time(val) {

//console.log(1)

this.dateTime = val

//console.log(val)

},

addAlarm() {

var alarmMills = new Date(this.dateTime).getTime()

var nowMills = new Date().getTime()

console.log(alarmMills)

console.log(nowMills)

if (alarmMills - nowMills < 0) {

uni.showToast({

title: '闹钟时间不能小于当前时间',

icon: 'none',

duration:1000

})

return false;

}

uniCloud.callFunction({

name:"addAlarm",

data:{

alarm\_title:this.alarm\_title,

alarm\_content:this.alarm\_content,

dateTime:this.dateTime,

user\_id:this.user\_id

},

success(res) {

console.log(res)

},

fail(res) {

console.log(res)

}

})

uni.switchTab({

url:'../alarm/alarm'

})

}

}

}

</script>

<style>

.main{

background-color: #FFFFFF;

height: 1200rpx;

margin-top: 30px;

}

.alarm{

background-color: #F2F2F2;

height: 70rpx;

/\* border:#c9daf8; \*/

border-radius: 20rpx;

/\* border-style:groove;\*/

margin-top: 30rpx;

margin-left: 20rpx;

margin-right: 20rpx;

padding-left: 30rpx;

padding-right: 30rpx;

padding-top: 15rpx;

}

.alarm\_input {

font-size: 38rpx;

}

#alarm\_title {

}

#alarm\_content{

height: 200rpx;

}

#submit{

margin: 40px;

}

</style>

### 2.3.2闹钟列表

<template>

<view>

<view >

<!-- <view id="1">

<button type="primary" @click="loadAlarm()">加载</button>

</view> -->

<uni-list>

<uni-list >

<view v-for="item in list" @click="toDetail(item.\_id)">

<uni-list-item :title="item.dateTime" :note="item.alarm\_title" ></uni-list-item>

</view>

</uni-list>

</uni-list>

</view>

</view>

</template>

<script>

export default {

data() {

return {

user\_id:"31901245",

list:[]

}

},

onLoad(){

this.loadAlarm()

},

onShow(){

this.loadAlarm()

},

methods: {

toDetail(\_id){

//监听

this.$emit("click",event);

uni.navigateTo({

url:'../alarm\_detail/alarm\_detail?\_id=' + \_id

//url:'../alarm\_detail/alarm\_detail?alarm\_title=' + i + '&alarm\_content=' + c

//url:'../alarm\_detail/alarm\_detail?alarm\_title=' + list[index].alarm\_title + '&alarm\_content=' + list[index].alarm\_content

})

},

loadAlarm() {

uniCloud.callFunction({

name:"loadAlarm",

data:{

user\_id:this.user\_id

}

}).then((res)=>{

const {result} = res

this.list = result.data

})

}

},

onNavigationBarButtonTap:function(e){

// console.log(e.text);//提交

// console.log(e.fontSize);//16px

uni.navigateTo({

url:'../addalarm/addalarm'

})

}

}

</script>

<style>

</style>

### 2.3.3闹钟详情

<template>

<view class = "all">

<view class="main" >

<view class="alarm" id="alarm\_title">

<input class = "alarm\_input" placeholder="闹钟标题" v-model="alarm\_title"></input>

<!-- <input v-model="inputSerialNum" style="padding: 10rpx;font-size: 30rpx;" class="uni-input" focus placeholder="请输入值" /> -->

</view>

<view class="alarm" id="alarm\_content">

<textarea class = "alarm\_input" placeholder="请输入闹钟内容" v-model="alarm\_content"></textarea>

</view>

<long-date

type="day"

getDayNum="10"

:openStatus="true"

startTime="2021-10-01"

endTime="2022-06-01"

chooesMaxDay="8"

@select="Time">

</long-date>

<view id="submit">

<button type="primary" @click="updateAlarm()">修改</button>

</view>

<view id="delete">

<button type="primary" @click="deleteAlarm()">删除</button>

</view>

</view>

</view>

</template>

<script>

//导入子组件

import longDate from "@/components/long-date/long-date.vue"

export default {

components: {

longDate

},

data() {

return {

\_id:"",

alarm\_title:"",

alarm\_content:"",

dateTime : ''

}

},

onLoad(option) {

this.\_id = option.\_id

this.getAlarmDetail(option);

},

methods: {

getAlarmDetail(option) {

uniCloud.callFunction({

name:'getAlarmDetail',

data:{

\_id : this.\_id

}

}).then((res)=>{

const {result} = res

this.alarm\_title = result.data[0].alarm\_title

this.alarm\_content = result.data[0].alarm\_content

console.log(this.alarm\_title)

})

},

Time(val) {

//console.log(1)

this.dateTime = val

//console.log(val)

},

updateAlarm() {

//console.log("执行了吗")

//console.log(this.dateTime)

uniCloud.callFunction({

name:"updateAlarm",

data:{

\_id:this.\_id,

alarm\_title:this.alarm\_title,

alarm\_content:this.alarm\_content,

dateTime:this.dateTime

},

success(res) {

},

fail(res) {

console.log(res)

}

})

uni.switchTab({

url:'../alarm/alarm'

})

},

deleteAlarm() {

uniCloud.callFunction({

name:"deleteAlarm",

data:{

\_id:this.\_id

},

success(res) {

console.log(res)

},

fail(res) {

console.log(res)

}

}),

uni.switchTab({

url:'../alarm/alarm'

})

}

}

}

</script>

<style>

.main{

background-color: #FFFFFF;

height: 1200rpx;

margin-top: 30px;

}

.alarm{

background-color: #F2F2F2;

height: 70rpx;

/\* border:#c9daf8; \*/

border-radius: 20rpx;

/\* border-style:groove;\*/

margin-top: 30rpx;

margin-left: 20rpx;

margin-right: 20rpx;

padding-left: 30rpx;

padding-right: 30rpx;

padding-top: 15rpx;

}

.alarm\_input {

font-size: 38rpx;

}

#alarm\_title {

}

#alarm\_content{

height: 200rpx;

}

#submit{

margin-left: 50px;

margin-right:50px;

margin-top: 30px;

}

#delete{

margin-left: 50px;

margin-right:50px;

margin-top: 20px;

}

</style>

### 2.3.4课程表

<template>

<view>

<view>

<lpxtimetable :timetables="timetables"></lpxtimetable>

</view>

</view>

</template>

<script>

import lpxtimetable from '../../components/lpx-timetable/lpx-timetable.vue'

export default {

data() {

return {

timetables: [

['', '', '', '', '', '', '', '', '', '', '', ''],

['', '', '', '', '', '', '', '', '', '', '', ''],

['', '', '', '', '', '', '', '', '', '', '', ''],

['', '', '', '', '', '', '', '', '', '', '', ''],

['', '', '', '', '', '', '', '', '', '', '', '']

],

courseList: [

],

tea\_idnum: "",

course\_name: "",

course\_day: "",

start\_time: "",

end\_time: "",

course\_add: "",

theCourse: []

}

},

onLoad() {

//this.stu\_idnum = "31901238"

this.tea\_idnum = uni.getStorageSync("globalUser");

this.loadCourse()

},

onShow(){

this.loadCourse()

},

onPullDownRefresh() {

//监听下拉刷新动作的执行方法，每次手动下拉刷新都会执行一次

console.log('refresh');

this.loadCourse();

setTimeout(function() {

uni.stopPullDownRefresh(); //停止下拉刷新动画

}, 1000);

},

methods: {

loadCourse() {

uniCloud.callFunction({

name: "loadTeaCourse",

data: {

tea\_idnum: this.tea\_idnum

}

}).then((res) => {

const {

result

} = res

this.courseList = result.data

this.updateTimeTables()

})

},

updateTimeTables() {

//for循环根据每一门课的\_id到course中查找具体的课程详细,并改变timetables

for (var i = 0; i < this.courseList.length; i++) {

//this.searchCourse(this.courseList[i].course\_id)

this.searchCourse(this.courseList[i].course\_id)

// for (var j = this.start\_time; j <= this.end\_time; j++) {

// // console.log(j)

// //console.log(this.timetables[this.course\_day][j])

// this.timetables[this.course\_day][j] = this.course\_name

// }

}

},

searchCourse(course\_id) {

uniCloud.callFunction({

name:'searchCourse',

data:{

course\_id:course\_id

}

}).then((res)=>{

const {result} = res

this.theCourse = result.data

//该课程的详细信息

this.course\_name = result.data[0].course\_name

if (result.data[0].course\_day == "周一") {

this.course\_day = 0

} else if (result.data[0].course\_day == "周二") {

this.course\_day = 1

} else if (result.data[0].course\_day == "周三") {

this.course\_day = 2

} else if (result.data[0].course\_day == "周四") {

this.course\_day = 3

} else if (result.data[0].course\_day == "周五") {

this.course\_day = 4

}

console.log(this.course\_name)

console.log(this.course\_day)

this.start\_time = result.data[0].start\_time

this.end\_time = result.data[0].end\_time

this.course\_add = result.data[0].course\_add

console.log(this.start\_time + " " + this.end\_time)

for (var j = this.start\_time; j <= this.end\_time; j++) {

//this.timetables[this.course\_day][j - 1]= this.course\_name

this.$set(this.timetables[this.course\_day],j - 1,this.course\_name +"@"+this.course\_add)

}

//console.log("周三第二节课" + this.timetables[2][1])

//console.log("周三第三节课" + this.timetables[2][2])

// console.log(this.timetables)

// this.timetables[0][1] = "软件工程"

//this.$set(this.timetables[0],1,"软件工程")

console.log(this.timetables[0][1])

})

}

},

components: {

lpxtimetable

},

onNavigationBarButtonTap: function(e) {

console.log(e.text); //提交

console.log(e.fontSize); //16px

uni.navigateTo({

url: '../editcourse/editcourse'

})

}

}

</script>

<style>

</style>

### 2.3.5添加课程

<template>

<view>

<view class="main">

<view class="task" id="task\_course">

<textarea placeholder="请输入课程名" v-model="course\_name" class="text"></textarea>

</view>

<view class="task">

<select-lay :zindex="1211" :value="course\_day" name="name" placeholder="请选择课程所在日" :options="datalist"

@selectitem="selectdayitem" id="select">

</select-lay>

</view>

<view class="start\_endtime" id="starttime">

<select-lay :zindex="1211" :value="start\_time" name="name" placeholder="开始节数" :options="timelist"

@selectitem="selectstarttimeitem" id="select">

</select-lay>

</view>

<view class="start\_endtime" id="endtime">

<select-lay :zindex="1211" :value="end\_time" name="name" placeholder="结束节数" :options="timelist"

@selectitem="selectendtimeitem" id="select">

</select-lay>

</view>

<view class="task" id="task\_title">

<textarea value="" placeholder="请输入课程地点" v-model="course\_add" class="text"></textarea>

</view>

<!-- <view>{{course\_name}}</view>

<view>{{course\_day}}</view>

<view>{{start\_time}}</view>

<view>{{end\_time}}</view>

<view>{{course\_add}}</view> -->

<!-- <view class="task" id="task\_content">

<textarea value="" placeholder="请输入任务内容"></textarea>

</view> -->

</view>

<!-- <view class="title">实时获取输入值：{{course\_name}},{{course\_day}},{{start\_time}},{{end\_time}},{{course\_add}}</view> -->

<view id="submit">

<button @click="add\_course()" class="btn">提交</button>

</view>

</view>

</template>

<script>

export default {

data() {

return {

datalist: [],

timelist: [],

//模拟初始数据

tval: "value2",

// arrays: ['---请选择---','教师','学生','管理员'],

index: 0,

course\_name: '',

course\_day: '',

start\_time: '',

end\_time: '',

course\_add: '',

idnum: '',

coursenum: '',

id: '教师'

}

},

methods: {

selectdayitem(index, item) {

// console.log(item)

if (index >= 0) {

// this.tval = item.value;

this.course\_day = item.value;

} else {

this.tval = ""

}

},

selectstarttimeitem(index, item) {

// console.log(item)

if (index >= 0) {

// this.tval = item.value;

this.start\_time = item.value;

} else {

this.tval = ""

}

},

selectendtimeitem(index, item) {

// console.log(item)

if (index >= 0) {

// this.tval = item.value;

this.end\_time = item.value;

} else {

this.tval = ""

}

},

add\_course: function() {

uniCloud.callFunction({

name: "course\_exist",

data: {

course\_name: this.course\_name,

course\_day: this.course\_day,

start\_time: this.start\_time,

end\_time: this.end\_time,

course\_add: this.course\_add,

idnum: this.idnum,

id: ''

},

}).then((res) => {

const {

result

} = res

// console.log(result.coursenum )

// this.info=result.data

this.coursenum = result.coursenum

console.log(this.coursenum)

})

if (this.coursenum == 0) {

uniCloud.callFunction({

name: "add\_course",

data: {

course\_name: this.course\_name,

course\_day: this.course\_day,

start\_time: this.start\_time,

end\_time: this.end\_time,

course\_add: this.course\_add,

idnum: this.idnum,

},

// 成功

success(res) {

console.log("添加课程success");

uni.showToast({

title: '添加课程成功',

icon: 'none',

duration: 1000

});

},

})

}

if (this.coursenum != 0){

console.log("不必再添课")

}

let j = 0;

for (let i = 0; i < 98000; i++) {

if (i == 97900) {

if (this.id == '学生') {

uniCloud.callFunction({

name: "add\_course\_stu",

data: {

course\_name: this.course\_name,

course\_day: this.course\_day,

start\_time: this.start\_time,

end\_time: this.end\_time,

course\_add: this.course\_add,

idnum: this.idnum,

},

// 成功

success(res) {

console.log("添加课程success");

uni.showToast({

title: '添加课程成功',

icon: 'none',

duration: 1000

});

},

})

}

if (this.id == '教师') {

uniCloud.callFunction({

name: "add\_course\_tea",

data: {

course\_name: this.course\_name,

course\_day: this.course\_day,

start\_time: this.start\_time,

end\_time: this.end\_time,

course\_add: this.course\_add,

idnum: this.idnum,

},

// 成功

success(res) {

console.log("添加课程success");

uni.showToast({

title: '添加课程成功',

icon: 'none',

duration: 1000

});

},

})

}

}

}

// uniCloud.callFunction({

// name:"add\_course",

// data: {

// course\_name: this.course\_name,

// course\_day: this.course\_day,

// start\_time: this.start\_time,

// end\_time: this.end\_time,

// course\_add: this.course\_add,

// idnum:this.idnum,

// },

// // 成功

// success(res) {

// console.log("添加课程success");

// uni.showToast({

// title: '添加课程成功',

// icon: 'none',

// duration: 1000

// });

// },

// //失败

// fail(e) {

// console.log(e);

// }

// })

// uni.switchTab({

// url: '../course/course'

// });

}

},

// onBackPress() {

// uni.navigateTo({

// url: "../course/course",

// });

// return true

// },

onLoad(option) {

this.idnum = uni.getStorageSync("globalUser");

console.log(this.idnum)

// this.id = option.id

},

onReady() {

this.datalist = [{

label: "周一",

value: "周一"

},

{

label: "周二",

value: "周二"

},

{

label: "周三",

value: "周三"

},

{

label: "周四",

value: "周四"

},

{

label: "周五",

value: "周五"

},

{

label: "周六",

value: "周六"

},

{

label: "周日",

value: "周日"

}

];

this.timelist = [{

label: "1",

value: "1"

},

{

label: "2",

value: "2"

},

{

label: "3",

value: "3"

},

{

label: "4",

value: "4"

},

{

label: "5",

value: "5"

},

{

label: "6",

value: "6"

},

{

label: "7",

value: "7"

},

{

label: "8",

value: "8"

},

{

label: "9",

value: "9"

},

{

label: "10",

value: "10"

},

{

label: "11",

value: "11"

},

{

label: "12",

value: "12"

}

];

},

}

</script>

<style>

.main {

margin-top: 80rpx;

}

.task,

.start\_endtime {

height: 70rpx;

/\* border: #006400; \*/

background-color: #F2F2F2;

border-radius: 20rpx;

/\* border-style: groove; \*/

margin: 20rpx;

padding: 10rpx;

}

.start\_endtime {

width: 300rpx;

}

.text {

margin-top: 10rpx;

margin-left: 15rpx;

}

#starttime {

float: left;

}

#endtime {

float: right;

}

#task\_content {

height: 950rpx;

}

#submit {

margin: 40px;

}

.btn {

color: #FFFFFF;

background-color: #7CCD7C;

}

#task\_title {

margin-top: 200rpx;

}

</style>

### 2.3.6添加课程任务

<template>

<view>

<view class="main">

<view class="task" id="task\_course">

<textarea v-model="task\_course" placeholder="请输入课程名" class="text"></textarea>

</view>

<view class="task">

<select-lay :zindex="1211" :value="course\_day" name="name" placeholder="请选择课程所在日" :options="datalist"

@selectitem="selectdayitem" id="select">

</select-lay>

</view>

<view class="start\_endtime" id="starttime">

<select-lay :zindex="1211" :value="start\_time" name="name" placeholder="开始节数" :options="timelist"

@selectitem="selectstarttimeitem" id="select">

</select-lay>

</view>

<view class="start\_endtime" id="endtime">

<select-lay :zindex="1211" :value="end\_time" name="name" placeholder="结束节数" :options="timelist"

@selectitem="selectendtimeitem" id="select">

</select-lay>

</view>

<view class="task" id="task\_title">

<textarea v-model="task\_title" placeholder="请输入任务" class="text"></textarea>

</view>

<long-date type="between" getDayNum="10" :openStatus="true" startTime="2021-12-03" endTime="2022-02-01"

chooesMaxDay="8" @select="Time">

</long-date>

<view class="task" id="task\_content">

<textarea v-model="task\_content" placeholder="请输入任务内容" class="text"></textarea>

</view>

</view>

<view id="submit">

<button @click="submit\_task()">提交</button>

</view>

</view>

</template>

<script>

export default {

data() {

return {

task\_course: '',

task\_title: '',

task\_ddl: '',

task\_content: '',

iffinish: '',

idnum: '',

task\_ddl: "",

task\_day: '',

start\_time: '',

end\_time: '',

course\_exist: '',

course\_day: '',

timelist: [],

datalist: [],

task\_id: '',

course\_id: '',

stu\_list: [1,2,3]

}

},

onLoad() {

this.idnum = uni.getStorageSync("globalUser");

// console.log(this.idnum);

},

onReady() {

this.datalist = [{

label: "周一",

value: "周一"

},

{

label: "周二",

value: "周二"

},

{

label: "周三",

value: "周三"

},

{

label: "周四",

value: "周四"

},

{

label: "周五",

value: "周五"

},

{

label: "周六",

value: "周六"

},

{

label: "周日",

value: "周日"

}

];

this.timelist = [{

label: "1",

value: "1"

},

{

label: "2",

value: "2"

},

{

label: "3",

value: "3"

},

{

label: "4",

value: "4"

},

{

label: "5",

value: "5"

},

{

label: "6",

value: "6"

},

{

label: "7",

value: "7"

},

{

label: "8",

value: "8"

},

{

label: "9",

value: "9"

},

{

label: "10",

value: "10"

},

{

label: "11",

value: "11"

},

{

label: "12",

value: "12"

}

];

},

methods: {

selectdayitem(index, item) {

// console.log(item)

if (index >= 0) {

// this.tval = item.value;

this.course\_day = item.value;

} else {

this.tval = ""

}

},

selectstarttimeitem(index, item) {

// console.log(item)

if (index >= 0) {

// this.tval = item.value;

this.start\_time = item.value;

} else {

this.tval = ""

}

},

selectendtimeitem(index, item) {

// console.log(item)

if (index >= 0) {

// this.tval = item.value;

this.end\_time = item.value;

} else {

this.tval = ""

}

},

Time(val) {

console.log(val);

this.task\_ddl = val;

},

submit\_task: function() {

/\* 首先判断输入的课程名是否存在于course\_tea表中 \*/

uniCloud.callFunction({

name: "exist\_course\_tea",

data: {

teacher\_idnum: this.idnum,

task\_course: this.task\_course,

start\_time: this.start\_time,

end\_time: this.end\_time,

task\_title: this.task\_title,

task\_ddl: this.task\_ddl,

task\_content: this.task\_content,

}

}).then((res) => {

const {

result

} = res

if (result.course\_count == "0") {

uni.showToast({

title: '课程不存在',

icon: 'none',

duration: 1000

});

} else {

this.course\_id = result.course\_id //获取课程id

// console.log(result.task\_id);

uni.showToast({

title: '添加任务成功',

icon: 'none',

duration: 1000

});

uniCloud.callFunction({ //获取task\_id

name: "getTaskId",

data: {

teacher\_idnum: this.idnum,

task\_course: this.task\_course,

start\_time: this.start\_time,

end\_time: this.end\_time,

task\_title: this.task\_title,

task\_ddl: this.task\_ddl,

task\_content: this.task\_content,

}

}).then((res) => {

const {

result

} = res

this.task\_id = result.data

uniCloud.callFunction({

name: "get\_task\_stu",

data: {

course\_id: this.course\_id,

}

}).then((res) => {

const {

result

} = res

this.stu\_list = result.stu\_list

console.log(this.stu\_list.length)

for (let t = 0; t < this.stu\_list.length; t++) {

uniCloud.callFunction({

name: "add\_task\_stu",

data: {

stu\_idnum: this.stu\_list[t].stu\_idnum,

task\_id: this.task\_id,

task\_title: this.task\_title,

}

})

}

uni.switchTab({

url:"../task/task"

})

})

})

}

})

}

}

}

</script>

<style>

.main {

margin-top: 80rpx;

}

#task\_title {

margin-top: 180rpx;

}

.start\_endtime {

height: 70rpx;

/\* border: #006400; \*/

background-color: #F2F2F2;

border-radius: 20rpx;

/\* border-style: groove; \*/

margin: 20rpx;

padding: 10rpx;

}

.start\_endtime {

width: 300rpx;

}

#starttime {

float: left;

}

#endtime {

float: right;

}

.task {

height: 80rpx;

background-color: #F2F2F2;

border-radius: 20rpx;

margin: 20rpx;

}

.text {

padding-top: 15rpx;

margin-top: 20rpx;

margin-right: 30rpx;

margin-left: 20rpx;

}

#task\_content {

height: 950rpx;

}

#submit {

margin: 40px;

}

</style>

### 2.3.7个人信息

<script src="http://at.alicdn.com/t/font\_2971072\_04vv2tcbr43c.js">

</script>

<template>

<view>

<view class="content">

<view class="detail">

<svg class="icon" aria-hidden="true">

<use xlink:href="#icon-shenfenxuanze"></use>

</svg>

<view class="detailvalue">身份：{{id}}</view>

</view>

<view class="detail">

<svg class="icon" aria-hidden="true">

<use xlink:href="#icon-geren1"></use>

</svg>

<view class="detailvalue">

<view class="detailvalue">姓名：{{name}}</view>

</view>

</view>

<view class="detail">

<svg class="icon" aria-hidden="true">

<use xlink:href="#icon-shenfen"></use>

</svg>

<view class="detailvalue">工号：{{idnum}}</view>

</view>

<view class="detail">

<svg class="icon" aria-hidden="true">

<use xlink:href="#icon-shouji1"></use>

</svg>

<view class="detailvalue">手机：{{tel}}

</view>

</view>

<view class="detail">

<svg class="icon" aria-hidden="true">

<use xlink:href="#icon-youxiang3"></use>

</svg>

<view class="detailvalue">邮箱：{{email}}</view>

</view>

</view>

<button @click="edit()" class="btn">编辑</button>

</view>

</template>

<script>

export default {

data() {

return {

id: "教师",

name: "",

idnum: '',

tel: "",

email: "",

}

},

onShow() {

this.idnum = uni.getStorageSync("globalUser");

console.log(this.idnum)

this.get\_tea\_info()

},

methods: {

edit: function() {

uni.navigateTo({

url: '../editI/editI'

})

},

get\_tea\_info: function() {

uniCloud.callFunction({

name: "get\_tea\_info",

data: {

idnum: this.idnum,

},

}).then((res) => {

const {

result

} = res

this.name = result.data[0].name,

this.idnum = result.data[0].idnum,

this.tel = result.data[0].tel,

this.email = result.data[0].email

})

}

}

}

</script>

<style type="text/css">

.icon {

width: 1.2em;

height: 1.2em;

vertical-align: -0.15em;

fill: currentColor;

overflow: hidden;

margin-right: 5px;

}

.content {

margin-top: 30rpx;

}

.detailvalue\_item1{

float: left;

}

.detailvalue\_item2{

float: ri;

}

.detail {

font-size: 40rpx;

margin-top: 20rpx;

margin-right: 30rpx;

margin-left: 30rpx;

padding: 20rpx;

/\* background-color: #F2F2F2; \*/

background-color: #F2F2F2;

border-radius: 20rpx;

display: flex;

flex-direction: row;

}

.detailvalue {

/\* text-decoration: underline; \*/

color: #3F536E;

}

.btn {

width: 400rpx;

display: block;

margin: 40rpx auto;

background-color: #7CCD7C;

color: #FFFFFF;

}

</style>

### 2.3.8登录

<template>

<view class="zai-box">

<image src="../../static/logo.jpg" mode='aspectFit' class="zai-logo"></image>

<view class="zai-form">

<!-- <select-lay :zindex="1211" :value="tval" name="name" placeholder="请选择身份" :options="datalist"

@selectitem="selectitem" id="select">

</select-lay> -->

<!-- <view class="title">实时获取输入值：{{inputrole}},{{inputid}},{{inputpwd}}</view> -->

<!-- <view class="role" v-if="inputrole==t">身份：老师</view>

<view class="role" v-if="inputrole==s">身份：学生</view>

<view class="role" v-if="inputrole==m">身份：管理员</view> -->

<!-- <input class="zai-input" @input="onKeyInputidnum" placeholder-class placeholder="请输入教工号/学号" v-model="inputteacherid"/> -->

<input class="zai-input" placeholder-class placeholder="请输入教工号" v-model="inputidnum" />

<input class="zai-input" placeholder-class password placeholder="请输入密码" v-model="inputpwd" />

<view class="zai-label-forget">忘记密码？</view>

<button class="zai-btn" @click="login()">立即登录</button>

<navigator url="../register/register" hover-class="none" class="zai-label">还没有账号？点此注册.</navigator>

</view>

</view>

</template>

<script>

import selectlay from "../../components/uni-transition/uni-transition.vue"

export default {

data() {

return {

//模拟数据列表

datalist: [],

//模拟初始数据

tval: "value2",

arrays: ['---请选择---', '教师', '学生', '管理员'],

index: 0,

inputid: '教师',

inputidnum: '',

inputpwd: '',

statu: '',

data: '',

msg: '',

ifpass: '',

id:'教师'

}

idnum: ''

},

onLoad() {

this.idnum = uni.getStorageSync("globalUser");

console.log(this.idnum)

},

onReady() {

this.datalist = [{

label: "教师",

value: "教师"

},

{

label: "学生",

value: "学生"

},

{

label: "管理员",

value: "管理员"

}

];

},

methods: {

selectitem(index, item) {

// console.log(item)

if (index >= 0) {

this.tval = item.value;

this.inputid = item.value;

} else {

this.tval = ""

}

},

// onKeyInputidnum: function(event) {

// // this.inputteacherid = event.target.value

// // console.log("test-------",this.inputValue)

// },

login() {

// if (this.inputrole == '') {

// uni.showToast({

// title: '请选择身份',

// icon: 'none',

// duration: 1000

// });

// return false;

// }

if (this.inputid == '') {

uni.showToast({

title: '请输入教工号/学号',

icon: 'none',

duration: 1000

});

return false;

}

if (this.inputpwd == '') {

uni.showToast({

title: '请输入密码',

icon: 'none',

duration: 1000

});

return false;

}

uniCloud.callFunction({

name: "judge",

data: {

id: this.inputid,

idnum: this.inputidnum,

pwd: this.inputpwd,

}

}).then((res) => {

const {

result

} = res

this.ifpass = result.ifpass

if (this.ifpass == 1) {

console.log("okok")

uni.setStorageSync("globalUser", this.inputidnum);

uni.setStorageSync("globalUserId", this.inputid);

if (this.inputid == "学生") {

uni.switchTab({

url: '../course/course?idnum=' + this.inputidnum

});

}

else if(this.inputid == "教师"){

uni.switchTab({

url: '../teacher\_course/teacher\_course?idnum=' + this.inputidnum

});

}

else if(this.inputid == "管理员"){

uni.navigateTo({

url: '../check/check'

});

}

} else {

console.log("nonoo"),

uni.showToast({

title: '输入的账号或密码不正确',

icon: 'none',

duration: 1000

});

}

})

},

},

components: {

selectlay

}

}

</script>

<style>

.zai-box {

padding: 0 100upx;

position: relative;

}

.zai-logo {

top: 20rpx;

width: 100%;

height: 310upx;

}

.zai-title {

position: absolute;

top: 0;

line-height: 360upx;

font-size: 68upx;

color: #fff;

text-align: center;

width: 100%;

margin-left: -100upx;

}

.zai-form {

margin-top: 100upx;

}

.zai-input {

background: #e2f5fc;

margin-top: 30upx;

border-radius: 100upx;

padding: 20upx 40upx;

font-size: 36upx;

}

.input-placeholder,

.zai-input {

color: #94afce;

}

.zai-label {

margin-top: 130rpx;

padding: 60upx 0;

text-align: center;

font-size: 30upx;

color: #a7b6d0;

}

.zai-label-forget {

margin-top: -5rpx;

padding: 60upx 0;

text-align: center;

font-size: 30upx;

color: #a7b6d0;

}

.zai-btn {

top: 160rpx;

background: #228B22;

color: #fff;

border: 0;

border-radius: 100upx;

font-size: 36upx;

}

.zai-btn:after {

border: 0;

}

/\*按钮点击效果\*/

.zai-btn.button-hover {

transform: translate(1upx, 1upx);

}

</style>

### 2.3.9注册

<template>

<view class="zai-box">

<image src="../../static/logo.jpg" mode='aspectFit' class="zai-logo"></image>

<view class="zai-form">

<select-lay :zindex="1211" :value="tval" name="name" placeholder="请选择身份" :options="datalist"

@selectitem="selectitem" id="select">

</select-lay>

<input class="zai-input" placeholder-class placeholder="请输入教工/学生姓名" v-model="regname" />

<input class="zai-input" placeholder-class placeholder="请输入教工号/学号" v-model="regidnum" />

<input class="zai-input" placeholder-class placeholder="请输入验证手机号" v-model="regtel" />

<input class="zai-input" placeholder-class placeholder="请输入验证邮箱" v-model="regemail" />

<input class="zai-input" placeholder-class password placeholder="请输入密码" v-model="regpwd" />

<input class="zai-input" placeholder-class password placeholder="请再输入一次密码" v-model="regpwd2" />

<!-- <view>"身份："+{{regid}} +"name:"+{{regname}}</view> -->

<button class="zai-btn" @click="reg()">立即注册</button>

<navigator url="../zaizai-login/index" open-type='navigateBack' hover-class="none" class="zai-label">

已有账号，点此去登录.</navigator>

</view>

</view>

</template>

<script>

export default {

data() {

return {

//模拟数据列表

datalist: [],

//模拟初始数据

tval: "value2",

arrays: ['---请选择---', '教师', '学生', '管理员'],

index: 0,

regid: "",

regname: "",

regidnum: "",

regtel: "",

regemail: "",

regpwd: "",

regpwd2: "",

ifpass:'',

}

},

onReady() {

this.datalist = [{

label: "教师",

value: "教师"

},

{

label: "学生",

value: "学生"

},

];

},

methods: {

selectitem(index, item) {

console.log(item)

if (index >= 0) {

this.tval = item.value;

this.regid = item.value;

} else {

this.tval = ""

}

},

reg() {

if (this.regid == '') {

uni.showToast({

title: '选择身份',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regname == '') {

uni.showToast({

title: '请输入姓名',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regtel == '') {

uni.showToast({

title: '请输入手机号',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regtel.length != 11) {

uni.showToast({

title: '请输入正确格式的手机号',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regemail == '') {

uni.showToast({

title: '请输入邮箱',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regpwd == '') {

uni.showToast({

title: '请输入密码',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regpwd2 == '') {

uni.showToast({

title: '请再输入一次密码',

icon: 'none',

duration: 1000

});

return false;

}

if (this.regpwd != this.regpwd2) {

uni.showToast({

title: '两次输入的密码不一致',

icon: 'none',

duration: 1000

});

return false;

}

uniCloud.callFunction({

name:"ifreg",

data:{

id:this.regid,

idnum:this.regidnum,

// pwd:this.inputpwd,

}

}).then((res)=>{

const {result}=res

this.ifpass=result.ifpass

if(this.ifpass==1){

console.log("该教工号/学号已被注册"),

uni.showToast({

title: '该教工号/学号已被注册',

icon: 'none',

duration: 1000

});

}

})

if(this.ifpass!=1){

uniCloud.callFunction({

name: 'reg', // 云函数名字

// 传输数据

data: {

id: this.regid,

name: this.regname,

idnum: this.regidnum,

tel: this.regtel,

email: this.regemail,

pwd:this.regpwd,

},

// 成功

success(res) {

console.log("success");

uni.showToast({

title: '注册申请发送成功',

icon: 'none',

duration: 1000

});

uni.navigateTo({

url: "../login/login"

})

},

//失败

fail(e) {

console.log(e);

}

})

}

// uniCloud.callFunction({

// name: 'reg', // 云函数名字

// // 传输数据

// data: {

// id: this.regid,

// name: this.regname,

// idnum: this.regidnum,

// tel: this.regtel,

// email: this.regemail,

// pwd:this.regpwd,

// },

// // 成功

// success(res) {

// console.log("success");

// uni.showToast({

// title: '注册申请发送成功',

// icon: 'none',

// duration: 1000

// });

// uni.navigateTo({

// url: "../login/login"

// })

// },

// //失败

// fail(e) {

// console.log(e);

// }

// })

}

},

}

</script>

<style>

.zai-box {

padding: 0 100upx;

position: relative;

}

.zai-logo {

top: 20rpx;

width: 100%;

height: 310upx;

}

.zai-title {

position: absolute;

top: 0;

line-height: 360upx;

font-size: 68upx;

color: #fff;

text-align: center;

width: 100%;

margin-left: -100upx;

}

.zai-form {

margin-top: 60upx;

}

.zai-input {

background: #e2f5fc;

margin-top: 30upx;

border-radius: 100upx;

padding: 20upx 40upx;

font-size: 36upx;

}

.input-placeholder,

.zai-input {

color: #94afce;

}

.zai-label {

margin-top: 10rpx;

padding: 60upx 0;

text-align: center;

font-size: 30upx;

color: #a7b6d0;

}

.zai-btn {

top: 10rpx;

background: #228B22;

color: #fff;

border: 0;

border-radius: 100upx;

font-size: 36upx;

margin-top: 60upx;

}

.zai-btn:after {

border: 0;

}

/\*按钮点击效果\*/

.zai-btn.button-hover {

transform: translate(1upx, 1upx);

}

</style>

### 2.3.10课程任务进度详情

<template>

<view>

<view class="rate">

<cmd-circle cid="circle30" type="dashboard" :percent="this.rate" style="margin-top: 10px;margin-left: 10rpx;"></cmd-circle>

</view>

<view class="main" >

<link rel="stylesheet" href="http://at.alicdn.com/t/font\_2971072\_u0os1dwc1wk.css">

<view class="task" id="task\_course">

<i class="iconfont icon-wodekecheng" style="margin-top: 10rpx;margin-right: 10px;"></i>

<textarea :value="'任务课程 '+this.task\_course" placeholder="任务课程" ></textarea>

</view>

<view class="task" id="task\_ddl">

<i class="iconfont icon-59" style="margin-top: 10rpx;margin-right: 10px;"></i>

<textarea :value="'截止日期 '+this.task\_ddl" placeholder="截止日期"></textarea>

</view>

<view class="task" id="task\_title">

<i class="iconfont icon-biaoti" style="margin-top: 10rpx;margin-right: 10px;"></i>

<textarea :value="'任务标题 '+this.task\_title" placeholder="任务标题"></textarea>

</view>

<view class="task" id="task\_content">

<i class="iconfont icon-fuwuneirong" style="margin-top: 10rpx;margin-right: 10px;"></i>

<textarea :value="'任务内容 '+this.task\_content" placeholder="任务内容"></textarea>

</view>

<view id="submit">

<button style="border-radius: 10rpx;">提醒</button>

</view>

</view>

</view>

</template>

<script>

import cmdCircle from "@/components/cmd-circle/cmd-circle.vue"

export default {

components: {cmdCircle},

data() {

return {

// list:[

// {

// id:1,

// course:软件工程,

// title:课程预备项目,

// content:完成预备作业,

// },

// {

// id:2,

// course:高级程序设计,

// title:实验3,

// content:完成实验3,

// },

// {

// id:3,

// course:项目管理,

// title:大作业1,

// content:完成大作业,

// },

// {

// id:4,

// course:项目实践,

// title:课程实践项目,

// content:完成大作业,

// },

// ],

taskid:"",

task\_course:"",

task\_ddl:"",

task\_title:"",

task\_content:"",

rate:""

}

},

onLoad(option) {

this.taskid = option.taskid;

// console.log(option)

console.log(option.taskid)

this.gettaskdetail();

this.rate= option.rate

},

methods: {

gettaskdetail:function(){

uniCloud.callFunction({

name: 'taskdetailshow',

data: {

taskid: this.taskid,

id:'教师'

}

}).then((res) => {

const {

result

} = res

// console.log("success" + result)

// this.info=result.data

this.task\_course = result.data[0].course\_name

this.task\_ddl = result.data[0].task\_ddl

this.task\_title = result.data[0].task\_title

this.task\_content = result.data[0].task\_content

console.log(this.id)

// this.name=result.data.name

// this.tel=result.data.tel

// this.email=result.data.email

})

}

}

}

</script>

<style>

.main{

margin-top: 40rpx;

}

.rate{

background: url(../../components/rig\_login/static/zaizai-login/register.png) repeat fixed right;

}

.task{

height: 80rpx;

/\* border:#006400; \*/

background-color: #F2F2F2;

border-radius: 20rpx;

/\* border-style:groove; \*/

margin: 15rpx;

padding-top: 22rpx;

padding-left: 8rpx;

display: flex;

flex-direction: row;

}

#task\_content{

height: 900rpx;

}

#submit{

margin: 40px;

}

.icon {

width: 1em;

height: 1em;

vertical-align: -0.15em;

fill: currentColor;

overflow: hidden;

}

</style>

# 管理员端

## 云函数

## 组件

## 页面