# 第一章 概述

## 1.1背景

消费观念改变，生活速度的加快，事业的繁忙，宅男一族的出现，这些现象使大学校园有了更多的需求关于跑腿。对于学生来说，可能是是宅在宿舍，想要别人代拿快递；也能是忙于打游戏，想要别人代打一份午饭。对于老师来说，可能是一份文件要快速传递给另一个老师或同学。再加上学长学姐这种自然的关系，这些都促使校园跑腿比城市跑腿有更优势的平台。无论是在需求还是在相互之间的信任度，都远远高于城市跑腿。

## 1.2 概述

一个基于高校平台的即时发布接受订单的跑腿小程序。

## 1.3 优势创新点

1. 高校平台：如背景所说，高校每日的跑腿需求是相当大的，这就意味这相当大的用户群体，而且这个群体间的信任度是比较高的。
2. 即时发布接受：小程序的重要特点是即时发布，在一定时间内自动取消（可人为设置时间），帮助我们在短时间顺利完成跑腿。
3. 微信小程序：微信小程序自带的微特性正与我们的即时性相结合，给用户良好的体验，需要时打开即时发布一个跑腿订单，十分方便。
4. 帮助勤工俭学：这个小程序的更大意义在于帮助那些需要勤工俭学的同学，由于校园区域范围的限定，跑腿同学可能花费很少的时间完成跑腿，甚至不会打断原有的行走计划，只是捎件东西，不会影响时间，学习等，又可以有可观收入。

## 1.4 团队介绍

田凯彬，电子科技大学本科大二在读，负责后台及逻辑开发。

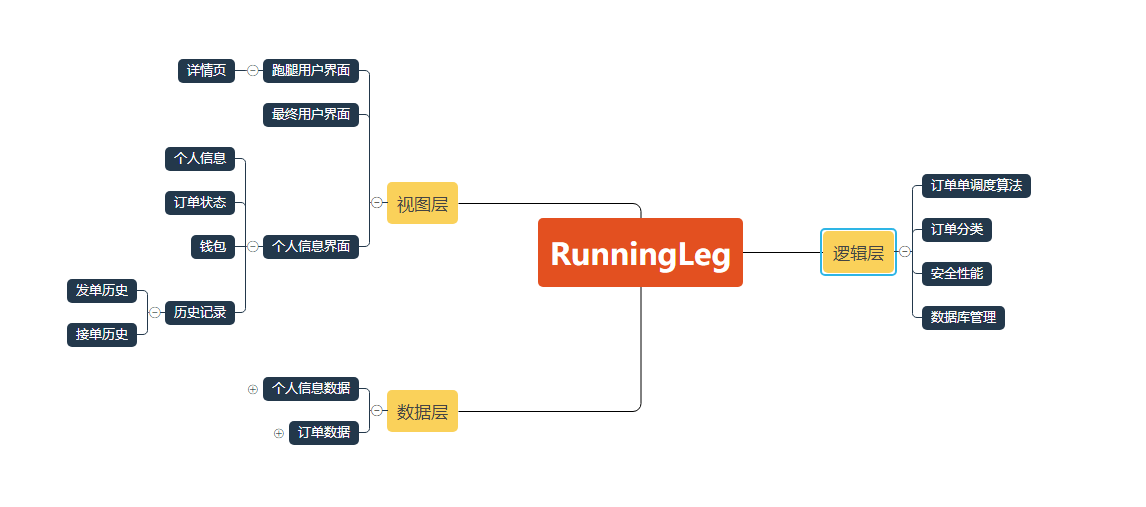
屈萌，电子科技大学本科大二在读，负责界面前端开发。

刘元林，电子科技大学本科大二在读，负责界面前端开发。

# 第二章 作品展示

## 2.1 系统整体设计

系统整体分视图层，逻辑层，数据层，整体设计如图2-1所示



【图2-1】

## 2.2 跑腿用户界面设计

### 2.2.2 功能介绍

本模块为用户进入小程序的首次看到的页面，由简洁的包裹图片以及简单的信息组成，如图2-2所示，用来展示系统中总共需要配送的订单以及订单信息。跑腿用户根据信息筛选选择适合的配送的外卖。点击图片可进入订单详情页如图2-3所示，详情页的信息包括订单的配送地址，配送时间，接收订单用户联系方式，以及该用户对订单的备注，此页信息供配送人员仔细产看，信息不可更改。若点击接收订单按钮则可以到指定地点领取物品经行配送。在查看完信息或接收订单信息之后可以返回首页，再次查看当前需要配送物品信息。

【图2-2】 【图2-3】

### 2.2.3 性能分析

首页展示信息首先会定时从后台拉取数据，其次每当用户返回到该页面时候也会重新刷新再次更新数据，并且支持缓存，考虑到性能与用户体验，每次在用户界面上展示的包裹数不超过七个。最后对于长时间没有接收的物品，系统会通知发单用户无人接收，并在首页取消展示，如何展示数据库上百的订单，是重点问题，可以结合算法，使得界面上展示的 不超过七个，而且优先展示等待时间过长的订单，利用到高响应比优先调度算法。

详情页中信息拉取自最终用户填写信息，接收配送按钮点击后，会给后台发送请求从而改变物品状态，同时首页也不再会出现该物品。对于多次进入详情页查看信息而没有接收的物品，其物品信息应该具有缓存，以减少服务器压力。当配送人员同意配送后会弹框提示接收成功，否则会提示请再次确认。

## 2.3 最终用户界面设计

### 2.3.1 功能介绍

此页面服务于最终用户，可以看到简单的表单的填写。最后领取物品的用户需要在此页面填写详细的关于物品信息，何时何地代取物品，送到什么地方，以及自己的联系方式，最后是选填项关于物品的备注，在配送中注意事项。在页面的下方是确定发单的按钮，点击后既可等待配送人员配送物品。完整页面如图2-4所示



【图2-4】

### 2.3.2 性能分析

由于小程序需要用户登录，因此在用户授权登录的时刻小程序后台可以拿到用户的相关信息，因此为了用户体验，在表单的填写环节，用户的昵称，手机号是可以提前为用户填写好，用户可以根据自己需求更改系统默认的信息。对于地址的选择可以直接根据地图选择合适位置，也可以手写输入。在系统性能方面，用户第一次填写好的信息，可以在以后发单填写时保存。当所有信息填写完毕之后，选择发单按钮，前端会向后端数据库发送请求，并增加一条配送物品记录，返回首页之后，就会在调度算法作用下展示出来。

## 2.4 个人信息界面设计

### 2.4.1 功能介绍

此页面面（图2-5）向所有用户，首次登录的用户可以进入个人信息子页面（图2-6），去完善个人信息，常驻用户也可以进入个人信息子页面去修改个人信息。如果正在接受业务的用户可以在订单状态子页面（图2-7）查看自己所接收的订单状态，假如，此时没有任何业务，此子页面将会提示去接收业务。当订单完成之后，可以去自己的钱包中查看自己的余额变化，如果此用户的任务已经完成，则会在此页面出现付款二维码。最后，用户可以时不时查看自己已经完成的订单历史。订单历史（图2-8）分为接单历史和发单历史界面。完整的个人信息页面如图2-5所示

【图2-5】 【图2-6】

【图2-7】 【图2-8】

### 2.4.2 性能分析

用户首次登录小程序后，后台拿到了用户头像和昵称等基本信息。用户需要去个人信息子页面完善信息。完善后的信息保存到由后台保存到数据库，等待再次被调用。其他子页面的信息更新后也会通过业务逻辑保存到数据库相应的表中。

首次登录用户需要完善个人信息后，之后所有更新的信息都有小程序本身去处理，这无疑增大了此小程序的体验感。

# 第三章 作品实现

## 2.1 跑腿用户界面实现

在此界面中，首先用到wxml与wxss实现基础样式，其中用到flex盒模型布局方式。在前后端交互中用到app.json实现从数据库拉取数据，在刷新按钮上，绑定事件，每点击一次，就会重新获取数据，并且展示在页面上。

拉取数据函数：

sendbill: function () {

var that = this;

app.request({

url: "https://theeighthday.cn/getbill",

success: function (res) {

length = res.data.length

console.log(res.data);

}

})

}

绑定事件函数：

<button bindtap='sendbill'>刷新</button>

## 2.2 最终用户界面实现

此界面由表单组成，为了使得用户填写正确信息并返回数据库，其中用到正则表达式约束填写内容。在页面下方的确定发单按钮上绑定事件，调用接口，返回数据库用户填写的信息

## 2.3 个人信息界面实现

此页面下面包含许多子页面，含有个人信息子页面、钱包子页面、历史记录子页面，订单状态子页面

个人信息总页面，此函数拿到个人的基本信息，并绑定到图2-5页面及其子页面上：

onLoad: function () {

if (app.globalData.userInfo) {

this.setData({

userInfo: app.globalData.userInfo,

hasUserInfo: true

})

} else if (this.data.canIUse) {

// 由于 getUserInfo 是网络请求，可能会在 Page.onLoad 之后才返回

// 所以此处加入 callback 以防止这种情况

app.userInfoReadyCallback = res => {

this.setData({

userInfo: res.userInfo,

hasUserInfo: true

})

}

} else {

// 在没有 open-type=getUserInfo 版本的兼容处理

wx.getUserInfo({

success: res => {

app.globalData.userInfo = res.userInfo

this.setData({

userInfo: res.userInfo,

hasUserInfo: true

})

}

})

}

},

getUserInfo: function (e) {

console.log(e)

app.globalData.userInfo = e.detail.userInfo

this.setData({

userInfo: e.detail.userInfo,

hasUserInfo: true

})

},

kindToggle: function (e) {

var id = e.currentTarget.id, histories = this.data.histories;

for (var i = 0, len = histories.length; i < len; ++i) {

if (histories[i].id == id) {

histories[i].open = !histories[i].open

} else {

histories[i].open = false

}

}

this.setData({

histories: histories

});

}

个人信息子页面函数，可以触发改变、完善个人信息：

changeName: function (e) {

var name = e.detail.value.trim();

if (name) {

wx.setStorageSync('name', name);

}

}，

changeAvatar: function (e) {

var that = this;

wx.chooseImage({

success: function (res) {

var tempFilePaths = res.tempFilePaths;

wx.saveFile({

tempFilePath: tempFilePaths[0],

success: function (res) {

var savedFilePath = res.savedFilePath;

wx.setStorageSync('avatar', savedFilePath);

that.setData({ avatar: savedFilePath });

}

});

}

})

},

订单状态子页面，当没有任何业务时，提示去添加业务

<block>

<view class="empty">

<text class="title">暂无任何订单</text>

<text class="content">请您前往「送外卖」或者「接外卖」添加业务</text>

</view>

</block>

历史记录子页面，根据相应业务，及时更新接单历史页面或者发单历史页面的数据,由于前期未绑定后台数据库，先使用本地数据

<view class='container'>

<view class="quick" wx:for="{{ histories}}" wx:key="{{item}}" >

<text>交易结束时间 : {{item.time}}</text>

<text>交易是否成功 : {{item.success}}</text>

<text>交易金额 : {{item.amount}}元</text>

</view>

/

</view>

histories:[

{

time: "2018/4/20 13：00",

success: "是",

amount: "50"

},

{

time: "2018/4/20 13：00",

success: "是",

amount: "50"

},

{

time: "2018/4/20 13：00",

success: "是",

amount: "50"

},

{

time: "2018/4/20 13：00",

success: "是",

amount: "50"

},]

}

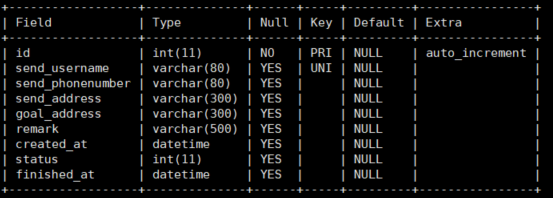
</view>

## 2.4 后端数据库实现

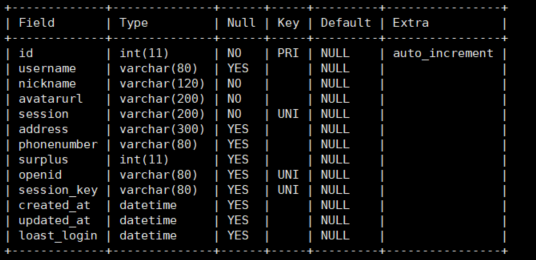
### 2.4.1系统环境搭建

在我们的作品中，服务器采用阿里云ECS，系统为Ubuntu16.04，搭建在Apache2运行平台上，采用Mysql数据库，并申请了证书，进行https连接。

账单表：



用户表：



### 2.4.2服务器实现

服务器开发语言为Python，采用了当前流行的Flask框架，Flask更加轻巧，采用MVC模式，非常适合小程序的敏捷开发。

# 第四章 测试和安全性分析

## 4.1 发布跑腿账单页面测试

暂无

## 4.2接受跑腿账单页面测试

暂无

## 4.3个人信息页面测试

暂无

## 4.4安全性分析

利用cookies，session机制。