**应用系统设计文档**

1. 引言

本文档描述了基于微信小程序的校园二手交易平台的设计，包括软件架构、模块设计、数据库设计和用户界面设计等方面的内容。

2. 软件架构

前端开发：HTML、CSS和JavaScript， 构建小程序界面和实现交互。

后端开发：微信云开发平台结合数据库，以实现后端逻辑和数据管理。

2.1 模块划分

校园二手交易小程序包含以下主要模块：

用户界面模块（View）：负责展示用户界面，接收用户输入，并将用户操作传递给控制器模块。

控制器模块（Controller）：负责处理用户操作，管理任务，与模型模块进行交互。

模型模块（Model）：负责管理任务数据、历史数据和历史订单，提供对数据库的读写操作。

2.2 模块间通信

模块间的通信通过消息传递方式进行，使用事件驱动的机制实现模块间的交互和数据传递.

3. 模块设计

3.1 用户界面模块（View）

设计用户界面，包括注册，登录，“主页”，“我的”，“发布二手物品”，“我买的”，“我卖的”等界面。

实现用户输入的响应，将用户操作事件传递给控制器模块。

3.2 控制器模块（Controller）

响应用户操作事件，处理“发布闲置”，“搜索”，“获取联系方式”等操作。

3.3 模型模块（Model）

设计数据库结构，包括用户表，发布物品表。

提供数据库的读写接口，包括用户的创建、更新和，闲置物品的创建和状态改变。

4. 数据库设计

列举出构建数据库所需的数据结构。

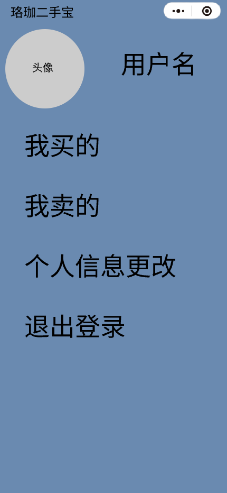
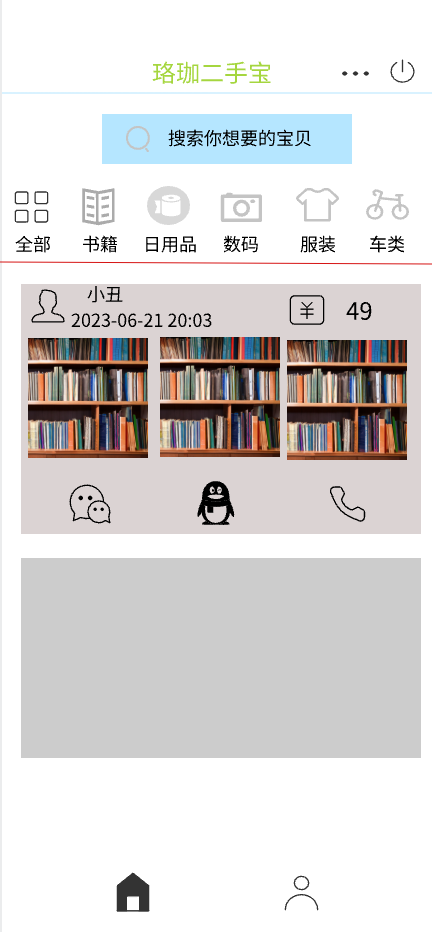
4.1 用户(user)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | User\_id | int | 10 | N | Y |  | 用户ID |
| 2 | User\_num | varchar | 64 | N | N |  | 用户学号 |
| 3 | User\_name | varchar | 64 | N | N |  | 用户昵称 |
| 4 | User\_gender | varchar | 64 | N | N |  | 用户性别 |
| 5 | User\_department | varchar | 64 | Y | N | Y | 用户院系 |
| 6 | User\_phonenum | varchar | 64 | N | Y |  | 用户手机号 |
| 7 | User wxnum | varchar | 64 | N | Y |  | 用户微信号 |
| 8 | User  qqnum | varchar | 64 | N | Y |  | 用户qq号 |

4.2 商品（item）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | Item\_id | Int | 10 | N | Y |  | 商品ID |
| 2 | Item\_name | varchar | 64 | N | N |  | 商品名称 |
| 3 | Item\_type | varchar | 64 | N | N |  | 商品类别 |
| 4 | Item\_detials | varchar | 64 | N | N |  | 商品描述 |
| 5 | Item\_sellor | varchar | 64 | N | N |  | 商品卖家 |
| 6 | Item\_price | varchar | 64 | N | N |  | 商品价格 |
| 7 | Item\_status | varchar | 64 | N | N |  | 商品求购或卖出 |

5. 用户界面设计



6. 总体流程

整体流程如下：

用户创建账号并登录。

买家：

用户可以浏览主页或分类查看二手商品。

用户可以选择商品查看其详情。

用户可以购买商品。

用户可以从“我的”界面查询到相关购买信息。

卖家：

用户可以上架自己的二手商品。

用户可以在“我的”界面中查看目前商品信息（是否被购买等）。

7. 接口定义

# 视图层（View）

class View:

def \_\_init\_\_(self, controller):

self.controller = controller

def display\_homepage(self):

# 显示主页

def display\_item\_details(self, item\_id):

# 显示物品详情

def display\_user\_profile(self, user\_id):

# 显示用户个人资料

def display\_error\_message(self, error\_message):

# 显示错误信息

# 控制层（Controller）

class Controller:

def \_\_init\_\_(self, model):

self.model = model

self.view = View(self)

def view\_homepage(self):

self.view.display\_homepage()

def view\_item\_details(self, item\_id):

item = self.model.get\_item\_details(item\_id)

self.view.display\_item\_details(item)

def view\_user\_profile(self, user\_id):

user = self.model.get\_user\_profile(user\_id)

self.view.display\_user\_profile(user)

def handle\_login(self, username, password):

if self.model.authenticate\_user(username, password):

# 登录成功

self.view.display\_homepage()

else:

# 登录失败

self.view.display\_error\_message("用户名或密码错误")

def handle\_item\_purchase(self, item\_id):

if self.model.check\_item\_availability(item\_id):

self.model.process\_item\_purchase(item\_id)

self.view.display\_user\_profile(current\_user\_id)

else:

self.view.display\_error\_message("该物品已被购买")

# 模型层（Model）

class Model:

def authenticate\_user(self, username, password):

# 验证用户身份

def get\_item\_details(self, item\_id):

# 获取物品详情

def get\_user\_profile(self, user\_id):

# 获取用户个人资料

def check\_item\_availability(self, item\_id):

# 检查物品是否可购买

def process\_item\_purchase(self, item\_id):

# 处理物品购买操作

# 创建实例并建立联系

model = Model()

controller = Controller(model)

view = View(controller)

controller.view\_homepage()