TexBookSub 北航图书订阅app

TexBookSub 北航图书订阅app
- 效果预览
- 简介
- 架构
- 功能设计与实现

效果预览

10:14
page
Verilog 设计实践
cyw20373456
Description
二手出！加急！😒介绍硬件描述语言Verilog HD...
Total Adjustment
Remain/Total
2/2
Confirm Edit
Delete

10:12
page
统地介绍航空飞行器（飞机、有翼导弹）的飞行动力学特性，内容结合工程实际，反映现代飞行动力学的一些新现象、新特点。
Quantity: 1
Price
¥12.00
销量
查看不同季度下该类物品销量
Spring
Summer
Autumn
Winter
Add to Cart

10:15
To Publish
Close
Manage Publish
Verilog 设计实践
cyw20373456
30.0
订阅
发布
已购

10:18
微机原理和接口
20.00¥ | 欧阳皓东
Show details
4
Verilog 设计实践
30.00¥ | cyw20373456
Show details
2
航空飞行器飞行动力学
12.00¥ | 教务处
Show details
2
Checkout More
Summary
Subtotal ¥104.00
Taxes ¥6.88
Total ¥110.88
Order & Clear Cart
订阅
发布
已购

10:17
Config
Aaaa
Aaaa
PUBLISH AMOUNT
10
Recommended?
SINGLE PRICE
30
DESCRIPTION
Good!
SUITED COLLEGE
计算机学院
Submit
订阅
发布
已购

10:17
Cancel
Photos
Save

简介

每到开学时分，课本订购就每到开学时分，课本订购就会成为每一位同学不得不耗费精力的一件琐事。或每个小班群内统计汇总购书情况，由小班长统一汇总提交至教材处、再以班级为单位去教材处领书、分发；或班级推荐的教材购买以后发现整个学期的课根本用不上此教 材；或同学们联系学长学姐购买二手书，经常因为找不到学长学姐出书、错过在大班群中的二手书信息而焦头烂额。可见，开学购买教材是一件极其繁琐、分散的事情。这启发我们， 能否建立一个校内的非营利平台，将教材处购买教材、向同学购买二手教材等教材购置渠道进行统一，在全程在线上进行。

这样一来，一方面方便教材处不用和各个小班对接，方便小班负责人不必处理教材订购、群收款、领书等琐事；另一方面也使得购买教材和出售二手教材的同学不需要再忍受信息不对等带来的不便。会成为每一位同学不得不耗费精力的一件琐事。或每个小班群内统计汇总购 书情况，由小班长统一汇总提交至教材处、再以班级为单位去教材处领书、分发；或班级推荐的教材购买以后发现整个学期的课根本用不上此教材；或同学们联系学长学姐购买二手书，经常因为找不到学长学姐出书、错过在大班群中的二手书信息而焦头烂额。可见，开学购买教材是一件极其繁琐、分散的事情。这启发我们，能否建立一个校内的非营利平台，将教材处购买教材、向同学购买二手教材等教材购置渠道进行统一，在全程在线上进行。

这样一来，一方面方便教材处不用和各个小班对接，方便小班负责人不必处理教材订购、群收款、领书等琐事；另一方面也使得购买教材和出售二手教材的同学不需要再忍受信息不对等带来的不便。

架构

```
├── ContentView.swift
├── FileManager.swift
├── Info.plist
├── Models
│   ├── BoardingPageModel.swift
│   ├── BookModel.swift
│   ├── TextBookModel.xcdatamodeld
│   │   └── TextBookModel.xcdatamodel
│   │       └── contents
│   └── UserModel.swift
├── Preview Content
│   └── Preview Assets.xcassets
│       └── Contents.json
├── ViewModel
│   └── ViewModel.swift
├── Views
│   ├── Book.swift
│   ├── BookDetail.swift
│   ├── BookManage.swift
│   ├── CheckOut.swift
│   ├── HomeView.swift
│   ├── Login.swift
│   ├── NavBar.swift
│   ├── OnBoarding.swift
│   ├── TrendingBook.swift
│   ├── cart.swift
│   └── uploadForm.swift
├── WorldLandmarkApp.swift
└── components
    ├── Photopicker.swift
    ├── PreviewCard.swift
    ├── cardDeck.swift
    ├── searchable.swift
    └── upperChart.swift
```

本项目采用了swiftUI + mvvm模式（伪），其中在Model中建立了两个实体user、book, 在ViewModel中，为了节省力气，把每个视图的ViewModel集成在了一个类里，并通过单例模式, 在每个页面View中获取并共享， 其中将ViewModel设置位@MainActor使得当@published变量变化时，更新各个页面。

```
@MainActor class ViewModel : ObservableObject{

    static let vm = ViewModel()
    @Published var publishedPool:[BookModel]
    @Published var userPool: [UserModel]
    @Published var nowUser: UserModel?
    @Published var displayMode: SearchTypes = SearchTypes.all
    @Published var searchKey: String = ""
```

其中部分组件写在了component中， 会采用@State @Binding绑定传值。但由于单例模式下的viewmodel 会一定可能导致页面刷新过慢，还需要进一步完善。另外，如需持久化，则可以采取该方式进行对json的读写，但由于时间原因，我没由实现User、Book的Codable, 还可以进一步改进

```
   func savePublishPool() {
       do {
           let data1 = try JSONEncoder().encode(publishedPool)
           try data1.write(to: savePublishedPoolPath, options: [.atomic,
.completeFileProtection])
       } catch {
           print("Unable to save data.")
       }
   }

   func saveUser() {
       do {
           let data1 = try JSONEncoder().encode(userPool)
           try data1.write(to: saveUserPoolPath, options: [.atomic, .completeFileProtection])
           let data2 = try JSONEncoder().encode(nowUser)
           try data2.write(to: saveNowUserPath, options: [.atomic, .completeFileProtection])
       } catch {
           print("Unable to save data.")
       }
   }
```

View视图中，其结构为

功能设计与实现

1. 登录注册

在viewModel中记录了当前用户nowUser，UserPool中记录了已经注册的用户，登陆注册的逻辑如下

```
    func makeregister(username: String, password: String) -> Bool{
        if (userPool.contains(where: {$0.username == username})) {
            return false
        } else {
            userPool.append(UserModel(username: username, password: password, cart: [:],
published: []))
//            saveUser()
            return true
        }
    }

    func login(username: String, password: String) -> (Bool, String){
        if let user = userPool.first(where: {$0.username == username}) {
           // do something with foo
            if user.password == password {
                nowUser = user
                return (true, "Login Succeeded")
            } else {
                return (false, "Password mismatched")
            }
        } else {
           // item could not be found
            return (false, "Username Not Found \nPlease Register First")
        }
    }
```

效果（正确为 cyw20373456, 123123）

2. 发布、管理教材

其代码位于`CheckOut.swift` 中的`PublishView`中，可以支持用户的上传发布新的教材，我们在view中对绑定得到参数进行新建Book对象然后调用ViewModel中的方法即可，ViewModel中的实现为

```
    func publishBookToPool(Bk:BookModel) {
        publishedPool.append(Bk)
//        savePublishPool() 如需持久化则加入这个
    }
```

效果为

此外还可以对书籍进行管理

```
    func deletePublish(Bk:BookModel) {
        publishedPool = publishedPool.filter { $0 !== Bk }
//        savePublishPool()
    }
//计算属性获取当前用户的发布
var nowUserPublish: [BookModel] {
        get {
            return publishedPool.filter{$0.publisher == nowUser?.username}
        }
    }
```

3. 购买教材

每个User都维护了一个Cart:[Book:Int] 的列表用于记录当前购买的商品和数量

```
    func buyBook(bk: BookModel, amount: Int) {
        print("buy book", bk, amount)
        nowUser?.addCart(bk: bk, num: amount)
//        saveUser()
    }

    func addCart(bk: BookModel, num: Int) {
        if cart[bk] != nil {
```

```
            cart[bk]! += num
            print(cart)
        } else {
            cart[bk] = num
            print(cart)
        }
    }
```

效果:

最初的购物车

买两本verilog

买完以后

此外我们购物主页的信息，通过计算属性来获取

```
var trending: [BookModel] {
        get {
            return publishedPool.filter{
                $0.recommend
            }
        }
    }


    @Published var searchRes: [BookModel]
    func getSearchRes() {
            switch displayMode {
            case .course:
                searchRes = publishedPool.filter({ $0.courses.hasPrefix(searchKey)})
            case .college:
                searchRes = publishedPool.filter({ $0.college.hasPrefix(searchKey)})
            case .bookName:
                searchRes = publishedPool.filter({ $0.title.hasPrefix(searchKey)})
            case .publisher:
                searchRes =  publishedPool.filter({ $0.publisher.hasPrefix(searchKey)})
            case .all:
                searchRes = publishedPool
            }

    }
```

10:11
Search Info
课程查询
院系查询
书名查询
发布者查询
Search书名查询...

favored list
航空飞行器
飞行动力学
飞行动力学
航空飞行器飞行动力学
飞行动力学
发布者:教务处
¥12.00
订阅
发布
已购

4. 清空购物车

Simulator File Edit Device I/O Features Debug Window Help
iPhone 11
10:18
Checkout More
Summary
Subtotal ¥0.00
Taxes ¥0.00
Total ¥0.00
Order & Clear Cart
intro-pic