**上海海事大学**



**应用软件开发技术课程设计**

**设计题目：“贪吃鱼”零食网店系统**

**姓名： 俞谦**

**学号: 201310311044**

**学 院： 信息工程学院**

**专 业： 计算机科学与技术**

**指导教师： 金世双**

**提交日期： 2016年.05月**

**分工情况：**

**俞谦：服务端设计、客户端逻辑设计、中间层设计**

**朱烨卿：客户端窗口设计、数据库管理、客户端管理**

**杨璐：客户端窗口设计、数据库管理、客户端管理**

**张俊杰：程序测试、数据库设计、服务端管理**

目录

[1. 引言 1](#_Toc286960866)

[1.1   目的 1](#_Toc286960867)

[1.2   项目背景 1](#_Toc286960868)

[1.4   参考资料 1](#_Toc286960869)

[2.   任务概述 1](#_Toc286960870)

[2.1   目标 1](#_Toc286960871)

[2.2   需求概述 1](#_Toc286960872)

[2.3   条件与限制 1](#_Toc286960873)

[3.   总体设计 1](#_Toc286960874)

[3.1   总体结构和模块外部设计 1](#_Toc286960875)

[3.2   功能分配 1](#_Toc286960876)

[4.   接口设计 2](#_Toc286960877)

[4.1   外部接口 2](#_Toc286960878)

[4.2   内部接口 2](#_Toc286960879)

[5.   数据结构设计 2](#_Toc286960880)

[6. 逻辑结构设计 2](#_Toc286960881)

[7. 物理结构设计 2](#_Toc286960882)

[8. 数据结构与程序的关系 2](#_Toc286960883)

[9.   运行设计 2](#_Toc286960884)

[9.1   运行模块的组合 2](#_Toc286960885)

[9.2   运行控制 2](#_Toc286960886)

[9.3   运行时间 3](#_Toc286960887)

[10.   出错处理设计 3](#_Toc286960888)

[10.1   出错输出信息 3](#_Toc286960889)

[10.2 出错处理对策 3](#_Toc286960890)

[11   安全保密设计 3](#_Toc286960891)

[12 维护设计 3](#_Toc286960892)

# 1. 引言

1.1   目的：

迎合网络零食销售需求量的增长，设计该系统以提供中小型零食商铺一个网络销售平台的手机APP软件。

1.2   项目背景

　　●   需要网上销售的零食的主播、商店、小卖部等。

●   提供给中小型客户的APP端解决方案。   
1.3   定义：

IOS .苹果手机系统（默认版本号9.3）

Storyboard Xcode.界面设计器

Cocoa架构 .Xcode UI设计库（类似于#include中的stdio）

Dmz .在局域网中把一台计算机的所有端口暴露到局域网之外，即所有对路由器的端口请求都转发至dmz主机

Property Object-c语言中的成员（数据、控件、方法等）

Cell 苹果APP tableview空间中的一个单元（行）

1.4   参考资料

【1】iOS 9 开发指南(附光盘) iOS9开发教程书籍 iOS 9开发快速入门教材 Xcode 7编程教程 ios8升级书籍.人民邮电出版社.2016版

【2】C#版 Socket编程（最简单的Socket通信功能）. http://blog.sina.com.cn/s/blog\_76b5256c0100wz7c.html. 2012-01-11

# 2.   任务概述

2.1   目标

在IOS客户端上实现商品浏览,个人信息管理,订单系统,评论系统等零食店应有的基础功能;架设服务器，实现多用户管理。

2.2   需求概述

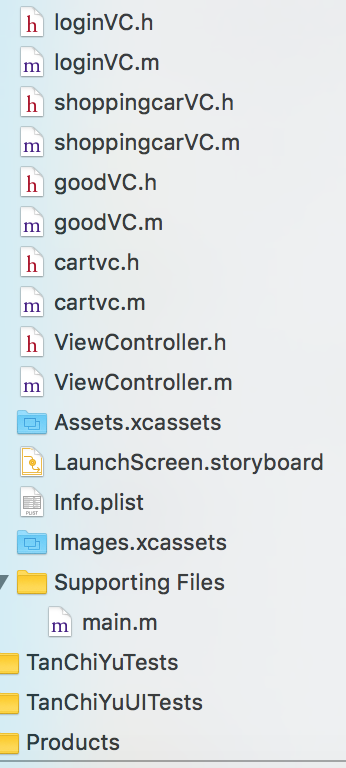
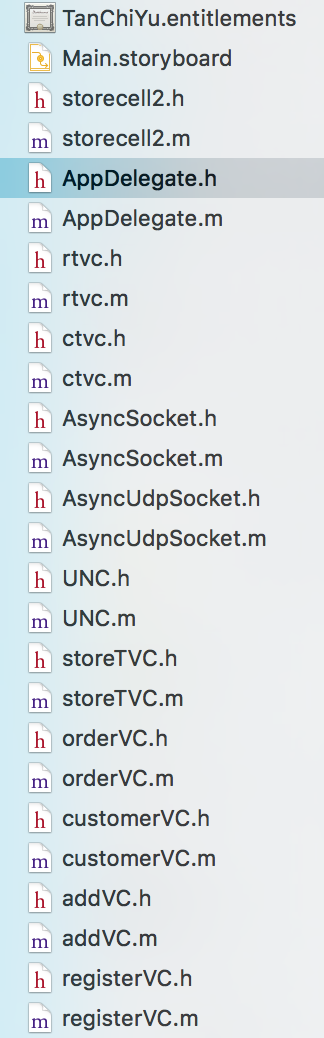
现网络主播喜欢同时开一些零食商铺吸引观众粉丝购买其零食，项目即提供一个网络销售平台，同时也可以成为一些小卖部 小超市的解决方案

2.3   条件与限制

尚未加入定位服务，无法判断是否超出配送范围，仅限买家端APP，需要手动取消不符合条件的订单，增加新商品等。

# 3.   总体设计

3.1   总体结构和模块外部设计

客户端：

3.1.2 客户端代码

文件集（类文件）2

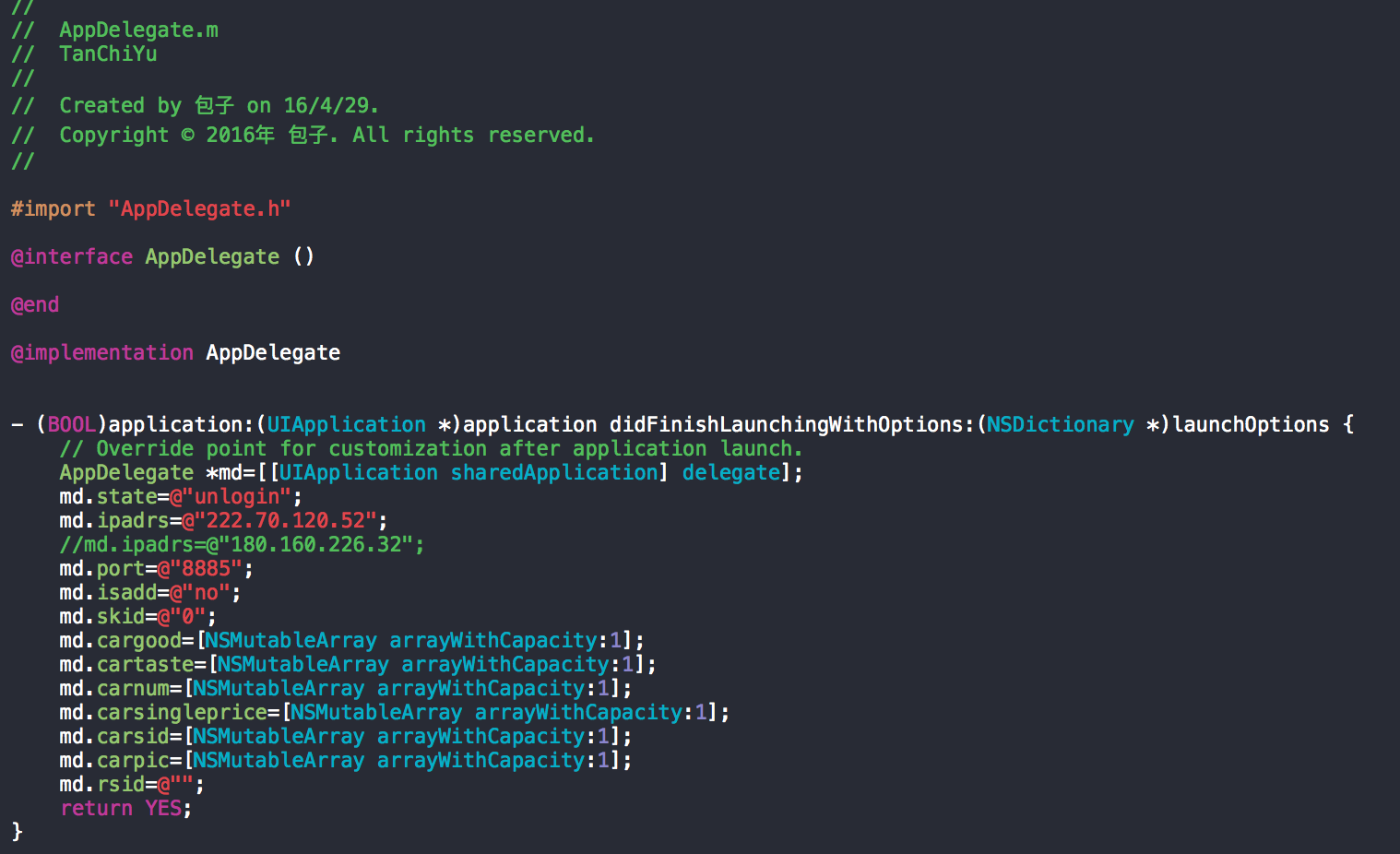
3.1.1 客户端代码

文件集（类文件）1

1. AppDelegate.h/.m 委托 这里“.h”为头文件，用来定义property，“.m”文件加载“.h”文件设置property数据，包括初始化、赋值等。该委托还可以加入各种方法。AppDelegate作用为为整个APP提供一个公用委托接口，实现全局变量，公共方法等方便类与类之间数据传递，通过方法的调用，成员私有化，提供数据成员是否可读/是否可写区别的解决方案。比如服务器的IP地址，访问端口就记录在这个文件内，类似的还有登录状态，订单信息，图片缓存（极大的缩短了载入时间，优化APP响应时间）等等。



3.1.3 AppDelegate.h



3.1.4 AppDelegate.m

1. loginVC.h/.m（登录界面类）、customerVC.h/.m（个人信息界面类）、orderVC.h/.m（订单信息界面类）、registerVC.h/.m（注册界面类）、addVC.h/.m（新增地址界面类）、loginVC.h/.m（登录界面类）“.h”同样为头文件，用来设置局部property成员，实现类内公共变量方法等。

3.1.6 cell.h

3.1.6 cell.h

3.1.6 cell.h

3.1.6 cell.h

3.1.6 cell.h

3.1.6 cell.h

3.1.6 cell.h

3.1.6 cell.h

3.1.6 cell.h

* 1. Socket类的装载

#import <UIKit/UIKit.h>

#import <sys/socket.h>

#import <netinet/in.h>

#import <arpa/inet.h>

#import <unistd.h>

#import "AsyncSocket.h"

* 1. Socket类的声明（在“.h”文件中，以被局部类内全局调用）

AsyncSocket \*askt;

* 1. Socket通信方法封装

-(void)send:(NSString \*)sendmsg{//send为方法名，sendmsg为参数，即需要发送给服务器的字段

pwd=@"err";//用来容纳服务器反馈数据，若无反馈即为“err”

iscomplete=@"err";//用来判断服务器反馈数据是否读取完毕

askt=[[AsyncSocket alloc]initWithDelegate:self];//Socket类的实例初始化，由于Xcode7的ARC编码系统，可以自动release释放内存。

NSError \*err=nil;//声明错误讯息变量

AppDelegate \*md=[[UIApplication sharedApplication] delegate];//声明委托

int port=[md.port intValue];//调取委托中的数据,这里调取的是服务器端口号，Nsstring类型强制转换为int。

if([askt connectToHost:md.ipadrs onPort:port error:&err])//判断是否成功建立Socket通信

{

NSData \*xmldata=[sendmsg dataUsingEncoding:NSUTF8StringEncoding];//即将发送的字段NTF8编码加密，转为二进制流。

[askt writeData:xmldata withTimeout:-1 tag:0];//调用Socket类中的方法，发送数据。

NSData \*rdata;//声明二进制变量

NSString\* astr;//声明字符串

[askt readDataWithTimeout:1 tag:3]; //封装读取服务器反馈的方法，调用该所在的方法即可以自动开始读取反馈。

astr=[[NSString alloc] initWithData:rdata encoding:NSUTF8StringEncoding];

}

else{

pwd=@"err";//反馈连接错误

}

* 1. 由于Xcode7的ARC编码系统，可以自动autorelease()释放内存。新版的编码器极大的方便了内存管理。
  2. 界面跳转的代码实现

AppDelegate \*md=[[UIApplication sharedApplication] delegate];

md.state=\_tusername.text;

UIStoryboard\* storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:[NSBundle mainBundle]];

UNC \*mvc=[storyboardinstantiateViewControllerWithIdentifier:@"UNC1"];

mvc.modalTransitionStyle=UIModalTransitionStyleFlipHorizontal;

[self presentViewController:mvc animated:YES completion:nil];

//其中调取了storyboard中的UI设计，修改了Appdelegate委托中的状态数据，使用presentViewController这个苹果私有API实现modal式的跳转（跳转后原界面类从内存中移除，仅存在在队列信息里）

* 1. Socket类读取服务器反馈信息的方法(举例每个界面不太一样 还有专门读取图片的方法，一下代码仅供参考)

-(void)onSocket:(AsyncSocket \*)sock didReadData:(NSData \*)data withTag:(long)tag{

NSString \* sting =[[NSString alloc]initWithData:data encoding:NSUTF8StringEncoding];//UTF8解码

pwd=sting; //读取的数据保存在pwd这个公共字符串变量里以供其在类的其他方法里被调用。

}

* 1. Socket类读取服务器反馈信息的方法(举例每个界面不太一样 还有专门读取图片的方法，一下代码仅供参考)

NSString \*send2s=@"login;";

send2s=[send2s stringByAppendingString:\_tusername.text];

send2s=[send2s stringByAppendingString:@";"];

send2s=[send2s stringByAppendingString:\_tpwd.text];//处理需要发送的数据，object-c对字符串的处理有点繁琐

[self send:send2s];

\_timer=[NSTimerscheduledTimerWithTimeInterval:0.5f target:self selector:@selector(delayMethod) userInfo:nil repeats:YES];//timer类实例，控制读取延时，由于苹果仅支持0.1秒的interval所以为了保证数据完整性读取速度不是很理想，其中（delayMethod）为调取的处理方法，该方法检测数据完整性，达到完整性条件后执行需要的操作。

* 1. 上述调取的方法(举例，每个界面不太一样 还有专门读取图片的方法，一下代码仅供参考)

- (void)dlyinit{

if ([rcv isEqualToString:@"err"])

{

if (intt>=25)

{

UIAlertView \*mBoxView = [[UIAlertView alloc]initWithTitle:@"提示" message:@"网络错误" delegate:nil cancelButtonTitle:@"确定" otherButtonTitles:nil];

[mBoxView show];

[self dismissViewControllerAnimated:YES completion:nil];

}

else

{

intt++;

[NSTimer scheduledTimerWithTimeInterval:0.1f target:self selector:@selector(dlyinit) userInfo:nil repeats:NO];

}

}

else

{

if (![iscomplete isEqualToString:rcv])

{

iscomplete=rcv;

intt=0;

[askt readDataWithTimeout:1 tag:0];

[NSTimer scheduledTimerWithTimeInterval:0.1f target:self selector:@selector(dlyinit) userInfo:nil repeats:NO];

return;

}

else if([iscomplete isEqualToString:rcv]&&intt<5)

{

intt++;

[askt readDataWithTimeout:1 tag:0];

[NSTimer scheduledTimerWithTimeInterval:0.1f target:self selector:@selector(dlyinit) userInfo:nil repeats:NO];

return;

}

else if([iscomplete isEqualToString:rcv]&&intt>=5)

{

NSArray \*rginfo=[rcv componentsSeparatedByString:@";"];

goodnum=[[rginfo objectAtIndex:0] intValue];

for(int i=1;i<=goodnum;i++)

{

NSString \*rgf=[rginfo objectAtIndex:i];

[goodinfo addObject:rgf];

}

[\_tgod reloadData];

for(int i=0;i<goodnum;i++)

{

rginv=[[goodinfoobjectAtIndex:i]componentsSeparatedByString:@"@"];

rgginv=[[rginvobjectAtIndex:3]componentsSeparatedByString:@"#"];

[sid addObject:[rgginv objectAtIndex:1]];

}

if([[rginfo objectAtIndex:0] intValue]==0){return;}

NSString \*send2s=@"goodpic;";

times=0;

send2s=[send2s stringByAppendingString:[sid objectAtIndex:times]];

intt=0;

[self send:send2s whattag:5];

[tmr invalidate];

tmr=[NSTimer scheduledTimerWithTimeInterval:0.1f target:self selector:@selector(delayinitpic) userInfo:nil repeats:NO];

}

}

}

1. UNC.h/.m（导航类）、storeVC.h/.m（商店界面类）、goodVC.h/.m（产品信息界面类）、shoppingcarVC.h/.m（购物车界面类）
   1. 导航间界面跳转采用push形式

3.1.5 cell.h

* 1. 设置委托

@interface storeTVC : UITableViewController<UITableViewDataSource,UITableViewDelegate,UISearchBarDelegate,UISearchDisplayDelegate>

设置tableview数据填充的委托 设置搜索框的操作方法委托

* 1. 实现委托的方法（部分）

#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView \*)tableView {

#warning Incomplete implementation, return the number of sections

return 1;//返回table的section（区域分类）个数

}

- (NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section {

#warning Incomplete implementation, return the number of rows

return goodnum+1;

}

-(NSString \*)tableView:(UITableView \*)tableView titleForHeaderInSection:(NSInteger)section

{

switch (section) {

case 0:

return @"请选择商品";//返回section的标题、结尾

break;

default:

return @"请选择商品";

break;

}

}

- (CGFloat)tableView:(UITableView \*)tableView heightForRowAtIndexPath:(NSIndexPath \*)indexPath

{

if(indexPath.row==0)//返回行高

{return 60;}

else

{return 150;}

* 1. 实现搜索的委托的方法（部分）

// 键盘中，搜索按钮被按下，执行的方法

- (void)searchBarSearchButtonClicked:(UISearchBar \*)searchBar{

NSLog(@"---%@",searchBar.text);

[self.tsb resignFirstResponder];// 放弃第一响应者

[self.navigationController setNavigationBarHidden:NO animated:YES];

goodnum=0;

goodinfo=[NSMutableArray arrayWithCapacity:1];

goodpic=[NSMutableArray arrayWithCapacity:1];

sid=[NSMutableArray arrayWithCapacity:1];

rows=[NSMutableArray arrayWithCapacity:1];

times=0;

st2=1;

[self initvw];

}

// 当搜索内容变化时，执行该方法。很有用，可以实现时实搜索

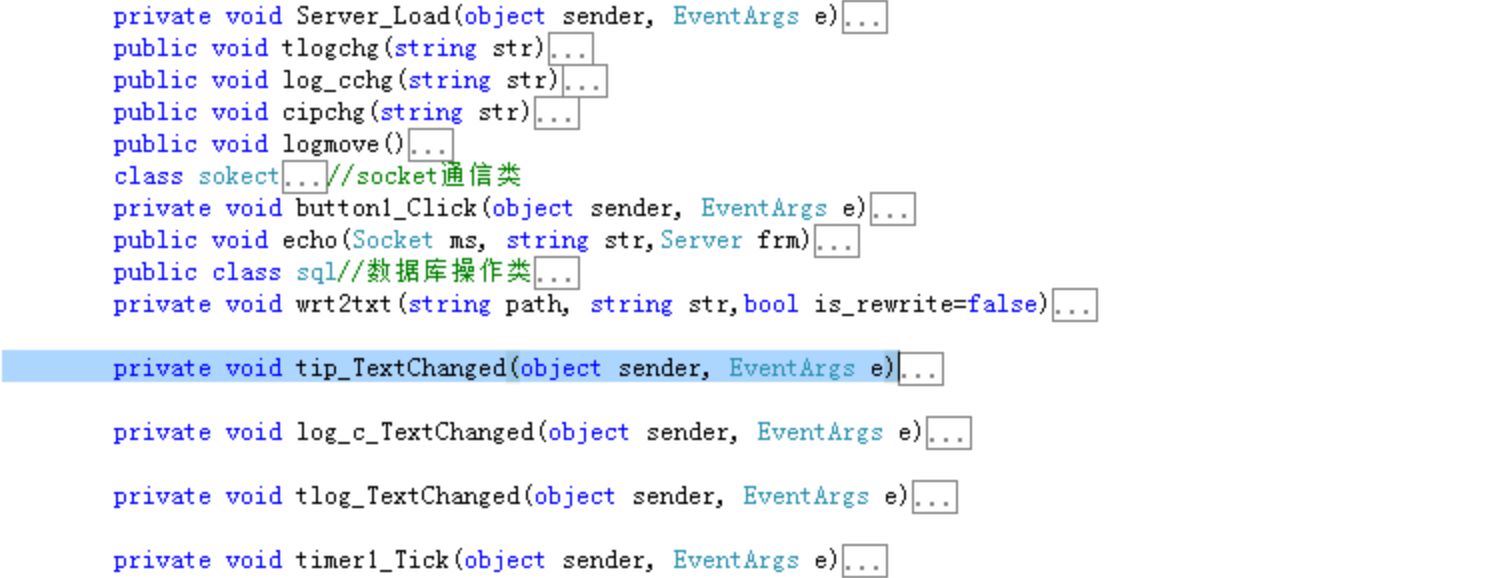
- (void)searchBar:(UISearchBar \*)searchBar textDidChange:(NSString \*)searchText;{ }

1. rtvc.h/.m、ctvc.h/.m、storecell2.h/.m、cartvc.h/.m table的cell

3.1.7 cell.m

3.1.6 cell.h

服务端：





3.2   功能分配

客户端：各页面分别实现登录系统 商店系统 购物车系统 订单系统 个人信息管理系统

服务端：sql类实现数据库查询处理，socket类实现双向通信，echo（）方法是在收到客户端消息后进行处理回复的方法

wrt2txt（）是委托主线程把信息写入日志文件的方法 其他大部分为界面处理的方法

# 4.   接口设计

4.1   外部接口

服务端界面采用winform窗口，通过按钮开启服务端，通过几个textbox观察服务端状态。

客户端使用storyboard进行interface设计实现人机交互的过程。

程序间使用套接字socket通信。

文本格式采用（通信请求）；（分隔符： ； @ # ...）（参数1）；（参数2）…的形式。

4.2   内部接口

客户端中采用AppDelegate委托实现全局变量的修改调取，在各头文件设置范围内全局变量函数方法等。

服务端使用委托实现跨线程调用，参数传递+公共变量 供内部调用函数、数据等。

# 5.   数据结构设计

使用一维或二维数组存取数据库结果。

循环或双循环遍历数组。

# 6. 逻辑结构设计

# 

# 7. 物理结构设计

服务器架设在win7计算机上，通过路由器联网（设置端口映射或dmz），客户端连上internet即可。

# 8. 数据结构与程序的关系

应用于数据库设计查询与处理的过程。

应用于模块间多维数据的传递。

# 9.   运行设计

9.1   运行模块的组合

具有多界面切换，能较好的交换信息，处理数据。

例如：登录系统，提交登录框中的用户名密码，通过套接字，发送到服务端，服务端对比数据库中的数据，返回用户名不存在或密码错误或登陆成功三种信息，不成功登陆跳出提示，成功登陆界面转入商店系统。

9.2   运行控制

软件运行时有有好的界面，基本能够实现用户的数据请求，按照模块间的关系进行调用控制。

9.3   运行时间

运行时间基本达到用户提出的要求，响应时间在0.2秒到1.5秒之间，有图片处理时速度稍慢延长到3秒左右，但是通过多线程控制实现非阻塞型处理过程，所以不会影响用户体验，体感响应时间平均在0.5秒。

# 10.   出错处理设计

10.1   出错输出信息

出错信息通过try{}catch{}抓取并写入日志文件。

10.2.1日志文件夹

10.2 出错处理对策

服务器定时清除缓存。

浏览日志文件判断错误。

10.2.1日志文件内容

10.2.1日志文件夹

# 

# 11   安全保密设计

用户密码设计。

收发信息中不包含任何敏感信息 逻辑上防止密码泄露。

收发信息使用UTF8二进制编码加密。

# 12 维护设计

服务端：在各个类中编写时都加入了try{}catch{}结构 同时在根目录下设置日志文件 夹保存所有收发信息。

客户端：加入了try{}catch{}结构，设置自动断点，连接上设计器即可读取错误断点信息。