**分类号：TP319 U D C：D10621-408-(2016)2672-0**

**密 级：公 开 编 号：2012121153**

**成都信息工程大学**

**学位论文**

**基于iOS教师选课系统App的设计与实现**

|  |  |
| --- | --- |
| **论文作者姓名：** | **张梓萌** |
| **申请学位专业：** | **网络工程** |
| **申请学位类别：** | **工学学士** |
| **指导教师姓名（职称）：** | **陈丁** |
| **论文提交日期：** | **2015年05月30日** |

**基于iOS教师选课系统App的设计与实现**

**摘 要**

随着教学课程的不断变化，教务系统的数据量不断的上涨。为更好的管理教师课程，提高教师选课效率，更好的方便教师选择课程，本系统在移动端（iOS）方向上对此进行了设计与实现，基本实现了教师在移动端（iOS）上的在线课程的选择，查看已选和未选的课程，对已选课程的修正等功能。本系统采用objective-c语言来编写，使用Xcode 9.3进行实现。数据库使用SQL server 进行设计，后台使用C#语言进行编写，使用VS2012进行实现。

**关键词：**教师选课系统；提高效率；课程选择

**The design and implementation of App based on iOS teacher selection system**

**Abstract**

With the continuous change of the teaching curriculum, the data of the educational administration system is rising. For better management of teachers, improve the efficiency in the course of teachers, to better facilitate the teachers choose the course, the system in the mobile terminal (IOS) direction regarding this has carried on the design and implementation of the basic realization of the teachers in the mobile terminal (IOS) online course selection, the view is selected and non selected courses, on the selected course correction function. This system uses the Objective-C language to write, uses the Xcode 9.3 to carry on the realization. Database using server SQL design, the background using the C# language to write, using VS2012 to achieve.

**Key words:** teacher selection system; improve efficiency; course selection

目录

1 引言 5

1.1课题背景 5

1.2 国内外研究现状 5

1.3 本课题研究的意义 5

1.4 本课题的研究方法 5

2 开发工具及软件环境 5

2.1 Xcode简介 5

2.2 SQL server 2008 6

2.3 Microsoft Visual Studio 2012简介 6

2.4 软件环境 6

3 需求分析 7

3.1系统目标 7

3.2系统功求分析 7

3.3 系统可行性分析 7

3.3.1 经济性 7

3.3.2技术性 7

3.3.3操作性 8

4系统设计 8

4.1系统设计思想和系统流程图 8

4.2 系统功能及模块划分 12

4.3 数据库设计 12

4.3.1 数据库设计 12

5 详细设计 13

5.1 创建实体类 13

5.2 定义数据访问程接口 14

5.3 定义业务逻辑层接口 22

5.4 定义工厂类 29

6 系统测试和维护 32

6.1 系统测试 32

6.2 系统维护 32

7 总结和展望 33

参考文献 33

致 谢 34

声 明 35

# 1 引言

## 1.1课题背景

计算机技术是在50年代时创作发明出来，随着几十年的发展，计算机技术也越来越成熟。在现在移动端（iOS）技术越来越受广大人民的喜爱，同时越来越多的人都开始转向移动端技术。现在，教师选课是各大院校管理工作的重要组成部分。但是现在选课方法的缺点是：教师选课时，必须在线下商量，然后教师再决定选择的课程，而且在选课时，容易出现误选，多选等错误。这项工作十分繁琐且容易出错，各部门的审核难度大，消耗大量的人力，难以实现选课信息的共享与沟通。

此系统采用移动端（iOS）技术，通过与后台的结合构建教师选课系统。使得教师对课程的选择更加的方便。教师在移动端可随时选择自己的课程。

本毕业设计的目的主要是为了检验学生学习的开发能力以及软件开发流程，设计开发一个学生系统。既锻炼了学生的实际动手能力，又引导学生进行了一次模拟实际产品的开发，对于学生以后工作能力的培养具有重要的意义。

## 1.2 国内外研究现状

从如今互联网发展速度来看，许多的选课操作都在互联网上操作，并且如今我们又进入了移动端领域，但是现在对于选课系统来说很少有系统进入移动端领域，很多操作都是在web端完成的。

## 1.3 本课题研究的意义

教师是教学之本，没有教师就没有我们学生的存在，而课程就成为老师和学生的羁绊，老师选择一个自己擅长的课程，将知识传递给学生，让学生终生受益。因此，完成和设计此课题，能减少选课的流程，大大的消除我们的教师在选课上的操作，只要有手机就能实现完成选课。

## 1.4 本课题的研究方法

首先分析教师选课需求，通过和老师的交流对其进行分析和总结，然后通过对需求的分析和总结确定此课题要实现的功能，然后确定ER图的建立，在根据ER图建立数据库，在根据数据库将后台的C#接口完成，然后在进行移动端（iOS）的UI设计，根据UI在进行切图，截图等操作，然后在通过对后台的连接完成此课题。

# 2 开发工具及软件环境

## 2.1 Xcode简介

Xcode 是运行在操作系统Mac OS X上的集成开发工具（[IDE](http://baike.baidu.com/subview/5775/5401660.htm" \t "_blank)），由[苹果公司](http://baike.baidu.com/view/15181.htm" \t "_blank)开发。Xcode是开发[OS X](http://baike.baidu.com/view/8497338.htm" \t "_blank) 和 [iOS](http://baike.baidu.com/subview/158983/8747673.htm" \t "_blank) 应用程序的最快捷的方式。Xcode 具有统一的用户[界面设计](http://baike.baidu.com/view/119481.htm" \t "_blank)，[编码](http://baike.baidu.com/subview/237708/11062012.htm" \t "_blank)、测试、[调试](http://baike.baidu.com/view/322913.htm" \t "_blank)都在一个简单的窗口内完成。Xcode 包含有GNU Compiler Collection自由软件 （GCC、 apple-darwin9-gcc-4.0.1 以及 apple-darwin9-gcc-4.2.1, 默认的是第一个），并支持 C语言、C++、Fortran、Objective-C、Objective-C++、Java、AppleScript、Python以及Ruby，还提供Cocoa、Carbon以及Java等编程模式。协力厂商更提供了 GNU Pascal，Free Pascal, Ada, C#, Perl, Haskell 和 D语言。Xcode套件使用 GDB作为其后台调试工具。

## 2.2 SQL server 2008

SQL Server 2008 在[Microsoft](http://baike.baidu.com/view/2422.htm" \t "_blank)的数据平台上发布，可以组织管理任何数据。可以将结构化、半结构化和非结构化文档的数据直接存储到数据库中。可以对数据进行查询、搜索、同步、报告和分析之类的操作。数据可以存储在各种设备上，从数据中心最大的服务器一直到桌面计算机和移动设备，它都可以控制数据而不用管数据存储在哪里。

SQL Server 2008 允许使用 Microsoft .NET 和[Visual Studio](http://baike.baidu.com/view/28727.htm" \t "_blank)开发的自定义应用程序中使用数据，在面向服务的架构（[SOA](http://baike.baidu.com/view/21305.htm" \t "_blank)）和通过 Microsoft BizTalk Server 进行的业务流程中使用数据。信息工作人员可以通过日常使用的工具直接访问数据。

## 2.3 Microsoft Visual Studio 2012简介

Microsoft Visual Studio 2012（简称VS2012）是美国[微软公司](http://baike.baidu.com/view/39784.htm" \t "_blank)的开发工具包系列产品。VS是一个基本[完整](http://baike.baidu.com/subview/631444/8036853.htm" \t "_blank)的开发工具集，它包括了整个[软件生命周期](http://baike.baidu.com/view/47193.htm" \t "_blank)中所需要的大部分工具，如UML工具、代码管控工具、[集成开发环境](http://baike.baidu.com/view/14867.htm" \t "_blank)(IDE)等等。所写的目标代码适用于微软支持的所有平台，包括[Microsoft Windows](http://baike.baidu.com/view/182749.htm" \t "_blank)、[Windows Mobile](http://baike.baidu.com/view/40733.htm" \t "_blank)、[Windows CE](http://baike.baidu.com/view/41539.htm" \t "_blank)、[.NET Framework](http://baike.baidu.com/view/18370.htm" \t "_blank)、[.NET Compact Framework](http://baike.baidu.com/view/1546355.htm" \t "_blank)和Microsoft [Silverlight](http://baike.baidu.com/view/942429.htm" \t "_blank) 及[Windows Phone](http://baike.baidu.com/view/2708518.htm" \t "_blank)。

## 2.4 软件环境

此课题的研发后台是在Windows 7 上，移动端（iOS）开发是在OS X 10.11.4，Windows 7 是由[微软公司](http://baike.baidu.com/view/39784.htm" \t "_blank)（[Microsoft](http://baike.baidu.com/view/2422.htm" \t "_blank)）开发的操作系统，内核版本号为[Windows NT](http://baike.baidu.com/view/41346.htm" \t "_blank) 6.1。[Windows 7](http://baike.baidu.com/view/761518.htm" \t "_blank)可供家庭及商业工作环境、笔记本电脑、平板电脑、多媒体中心等使用。Windows 7也延续了[Windows Vista](http://baike.baidu.com/view/7764.htm" \t "_blank)的[Aero](http://baike.baidu.com/view/748100.htm" \t "_blank) 风格，并且在此基础上增添了些许功能。

OS X是[苹果公司](http://baike.baidu.com/view/15181.htm" \t "_blank)为[Mac](http://baike.baidu.com/subview/32702/4924693.htm" \t "_blank)系列产品开发的专属操作系统。OS X是苹果Mac系列产品的预装系统，处处体现着简洁的宗旨。OS X是全世界第一个基于[FreeBSD](http://baike.baidu.com/view/21459.htm" \t "_blank)系统采用“[面向对象](http://baike.baidu.com/view/125370.htm" \t "_blank)操作系统”的全面的操作系统。“面向对象操作系统”是[史蒂夫·乔布斯](http://baike.baidu.com/view/90660.htm" \t "_blank)（*Steve Jobs*）于1985年被迫离开苹果后成立的NEXT公司所开发的。后来苹果公司收购了NeXT公司。史蒂夫·乔布斯重新担任苹果公司CEO，Mac开始使用的[Mac OS](http://baike.baidu.com/view/157344.htm" \t "_blank)系统得以整合到NeXT公司开发的[Openstep](http://baike.baidu.com/view/4834662.htm" \t "_blank)系统上。

# 3 需求分析

## 3.1系统目标

开发基于移动端（iOS）的App，将教师的基本信息录入数据库作为基本的资料，然后每学期每个学院发布下学期的课程，在某个规定的时间段里，由每个教师进行自由的选课，并且如果在选择的课程上出现了错选的情况，老师可以对所选课程进行退选。并且，老师可以随时方便的拿出自己的手机（iOS）查看自己的选课,也能看到自己的课程教室，课程时间等信息。

## 3.2系统功求分析

现在，随着互联网技术如此的发达，教师选课系统作为一种现代化的教学技术，以越来越受到重视。是一个学校不可缺少的一部分，所以对于老师来说都是至关重要的，所以教师选课能给教师提供重要的信息和快捷的查询手段。所以我们要实现的功能有以下几点：

个人信息：包括教师工行，教师的姓名，教师所属学院，教师的选课信息等。

课程信息：包括课程的教室，课程开课时间，课程选课时间，课程性质等。

登陆功能：教师首次进入系统，需要登陆系统，教师只要没有退出登陆，那么系统会保留上次的登陆信息。

退出登陆：教师可以退出登陆此系统，换另一个账号登陆。

提交选课：教师可以选择学院，查询出每个学院的课程，再选择课程，提交选课。

退选课程：教师可以退选自己已经选择的课程。

## 3.3 系统可行性分析

### 3.3.1 经济性

系统要求的硬件配置不高，实现可行。能实现在移动端（iOS）对选课的操作，提高教师选课的效率。

### 3.3.2技术性

随着互联网的高速发展，越来越多的机构都和Internet有了密切的联系，方便教师对课程的管理。此系统数据库采用广泛运用的SQL Server 2008，能非常快速且仔细的了解数据库的结构，接口的完成采用总所周知的C#语言，移动端（iOS）采用先进的Objective-c语言编写。

### 3.3.3操作性

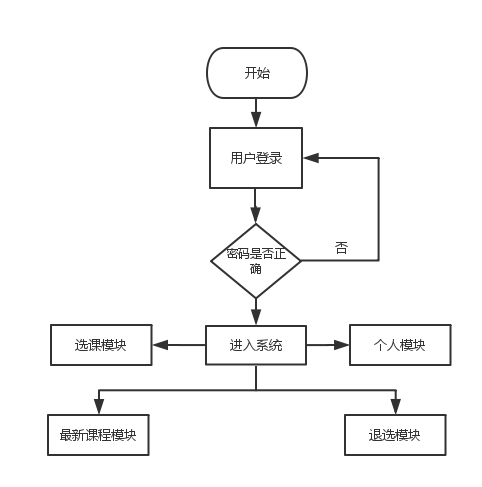
该系统的界面设计友好，操作简单。界面上非常清楚的写着每一个模块的功能。

# 4系统设计

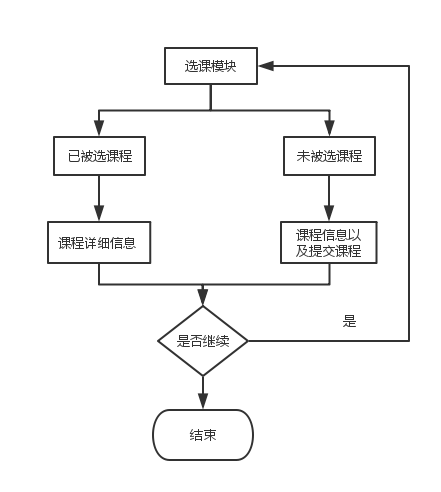
## 4.1系统设计思想和系统流程图

系统设计思想：1、系统分为前台显示和后台维护的两大部分。2、系统的前台和后台都采用MVC的设计模式，模型层：model类的实现，视图层：view的实现，控制层：viewcontroller类的实现，本系统的特点：①页面模块化：系统的界面在设计上都采用了模块化的设计思想，把多个页面共有的部分集成为一个模块。②业务处理封装化：采用用户界面层业务逻辑处理层数据存储层三层结构设计。③面向对象设计：采用面向对象的开发思想，将各个实体封装成对应的数据类，并为每个数据类开发一个操作类。

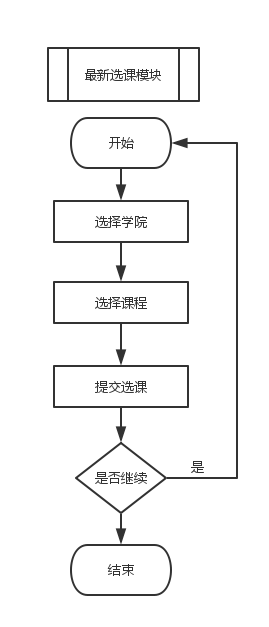
系统流程图：



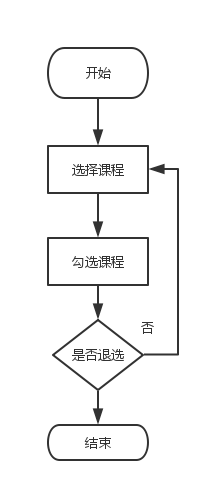
选课模块：



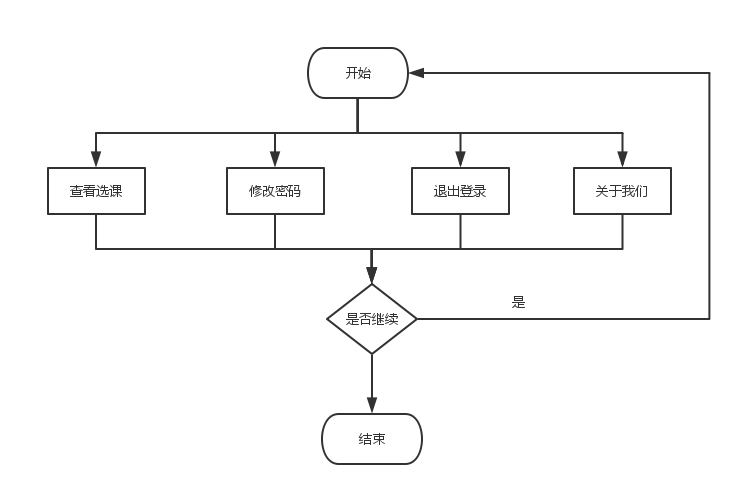
最新课程模板：



退选课程：



个人模块：



## 4.2 系统功能及模块划分

教师网上选课APP由教师选课。教师选课部分供授课教师使用，教师登录系统后，在规定时间内完成对本学期的选课任务。在功能实现上，可分为以下几个部分：

1、账户的登陆、退出功能，教师能在系统中登录（账号是自己的工号），能在个人设置中退出自己的账号。

2、修改密码，教师的登录一般有一个初始的密码（密码为123456），然后在个人设置中修改自己的密码。

3、选课功能，通过学院查询到发布的课程，然后在通过自己的需要查询到课程，然后在选择课程的班级。

4、退选功能，在退选页面中将自己的选课陈列出来，然后勾选出自己要退选的课程。

5、查询自己的选课以及课程的详细信息，在个人中心查看自己的课程以及课程的详细信息。

模块划分：

此系统可分为个人模块，选课模块，退选模块，查询模块。

个人模块：在个人模块中，可以查询自己的选课以及课程的选课信息，以及课程的详细信息。

选课模块：提交自己的选课。

退选模块：将自己的错选课程退出自己的选课。

查询模块：在查询模块中教师可以通过学院来查询到学院发布的课程，也可以查询自己的选课课程。

## 4.3 数据库设计

### 4.3.1 数据库设计

数据库是数据存储的核心设计，是按照数据结构来组织、存储和管理数据的仓库，数据管理不再仅仅是存储和管理数据，而转变成用户所需要的各种数据管理的方式。设计好数据库就相当于完成了整个系统的一半，好的数据库的设计能保证数据的准确性和正确性，只有设计出一个完整的数据库才能有效的降低系统的开发周期和提高系统的运行效率。

此系统的数据库从用户的需求出发，并且完成系统数据库的ER图，能非常清楚的看清系统的数据库框架和结构。

# 5 详细设计

## 5.1 创建实体类

教师类：存储教师的基本信息。

@interface Teacher : NSObject

@property (nonatomic, strong)NSString \*userName;

@property (nonatomic, strong)NSString \*realName;

@property (nonatomic, strong)NSString \*collegeName;

@end

学院类：存储学院的基本信息。

@interface collegeModel : NSObject

@property (nonatomic, strong)NSString \*collegeName;

@end

课程类：存储课程的基本信息。

@interface CourseContent : NSObject

@property (nonatomic, strong)NSString \*courseName;

@property (nonatomic, strong)NSString \*courseChoiseTime;

@property (nonatomic, strong)NSString \*courseTime;

@property (nonatomic, strong)NSString \*courseCollege;

@property (nonatomic, strong)NSString \*courseSourse;

@property (nonatomic, strong)NSString \*courseTeaType;

@end

学期类：存储学期的基本信息。

@interface GradeModel : NSObject

@property (nonatomic, strong)NSString \*year;

@end

班级类：存储班级的基本信息。

@interface ClassModel : NSObject

@property (nonatomic, strong)NSString \*num;

@end

专业类：存储专业的基本信息。

@interface Major : NSObject

@property (nonatomic,strong)NSString \*majorName;

@end

## 5.2 定义数据访问程接口

访问学院接口类：创建一个colleage类，此类定义的接口用于移动端（iOS）的访问学院数据接口。此访问采用http协议，当客服端向服务器发出请求，服务端就会发出响应信息去发给客服端，系统采用post的请求方式，将请求信息放入请求体中。

public class colleage : IHttpHandler

{

public void ProcessRequest(HttpContext context)

{

context.Response.ContentType = "text/plain";

String action = context.Request.Form["action"];

String result = "{\"ret\": 9, \"errMsg\": \"action error\"}";

if (action != null)

{

String data = context.Request.Form["data"];

byte[] bytes = Convert.FromBase64String(data);

String decodedStr = UTF8Encoding.Default.GetString(bytes);

// 解决中文 UTF8 乱码

byte[] buffer2 = Encoding.Convert(Encoding.UTF8, Encoding.Default, bytes, 0, bytes.Length);

decodedStr = Encoding.Default.GetString(buffer2, 0, buffer2.Length);

switch (action)

{

case "list":

result = list(decodedStr);

break;

case "add":

result = add(decodedStr);

break;

case "edit":

result = edit(decodedStr);

break;

case "delete":

result = delete(decodedStr);

break;

default:

break;

}

}

context.Response.Clear();

context.Response.Write(result);

}

private String list(String data) {

BLL.College bCollege = new BLL.College();

JObject jsonData = JObject.Parse(data);

string collegename = jsonData["collegename"].ToString();

string strWhere = " Name LIKE '%" + collegename + "%'";

List<Model.College> colleges = bCollege.GetModelList(strWhere);

string json = JsonConvert.SerializeObject(colleges).ToLower();

return "{\"ret\": 0, \"errMsg\": \"\", \"colleges\": " + json + "}"; ;

}

访问课程接口类：创建一个appcourses类，此类定义的接口用于移动端（iOS）的访问课程数据接口。此访问采用http协议，当客服端向服务器发出请求，服务端就会发出响应信息去发给客服端，系统采用post的请求方式，将请求信息放入请求体中。

public class appcourses : IHttpHandler

{

public void ProcessRequest(HttpContext context)

{

context.Response.ContentType = "application/json";

String action = context.Request.Form["action"];

String result = "{\"ret\": 9, \"errMsg\": \"action error\"}";

if (action != null)

{

String data = context.Request.Form["data"];

byte[] bytes = Convert.FromBase64String(data);

String decodedStr = UTF8Encoding.Default.GetString(bytes);

// 解决中文 UTF8 乱码

byte[] buffer2 = Encoding.Convert(Encoding.UTF8, Encoding.Default, bytes, 0, bytes.Length);

decodedStr = Encoding.Default.GetString(buffer2, 0, buffer2.Length);

switch (action)

{

case "selectable":

result = list(decodedStr, false);

break;

case "selected":

result = list(decodedStr, true);

break;

case "news":

result = news(decodedStr);

break;

default:

break;

}

}

context.Response.Clear();

context.Response.Write(result);

}

private string list(string data, bool isSelected)

{

JObject jsonData = JObject.Parse(data);

string year = jsonData["year"].ToString();

string collegeId = jsonData["collegeid"].ToString();

string userId = jsonData["userid"].ToString();

string strSql = " SELECT TeachingScheduleID, Course.Name as CourseName, College.Name as CollegeName, StartWeek, TotalHour, Grade.Year as GradeYear, Semester.No as SemesterNo, Major.Name as MajorName, TeachingSchedule.Class, Users.RealName as UserName, Status " +

" FROM TeachingSchedule " +

" LEFT JOIN TeachingPlan ON TeachingPlan.PlanID = TeachingSchedule.PlanID " +

" LEFT JOIN Course ON Course.CourseID = TeachingPlan.CourseID " +

" LEFT JOIN Semester ON Semester.SemesterID = TeachingPlan.SemesterID " +

" LEFT JOIN MajorSchedule ON MajorSchedule.MajorScheduleID = Semester.MajorScheduleID " +

" LEFT JOIN Major ON Major.MajorID = MajorSchedule.MajorID " +

" LEFT JOIN Grade ON Grade.GradeID = MajorSchedule.GradeID " +

" LEFT JOIN Users ON Users.UserID = TeachingSchedule.UserID " +

" LEFT JOIN College ON College.CollegeID = Course.CollegeID ";

if (isSelected)

{

strSql += " WHERE College.CollegeID = '"+ collegeId +"' AND " +

" Users.UserID = '"+ userId +"' AND " +

" Grade.Year = '"+ year +"' AND ( Status != -1 AND Status != 0) " +

" ORDER BY GradeYear DESC ";

}

else {

strSql += " WHERE College.CollegeID = '" + collegeId + "' AND " +

" Grade.Year = '" + year + "' AND Status = 0 " +

" ORDER BY GradeYear DESC ";

}

List<\_TeachingSchedule> tSchedules = new List<\_TeachingSchedule>();

DataSet ds = DbHelperSQL.Query(strSql);

for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

{

DataRow dr = ds.Tables[0].Rows[i];

\_TeachingSchedule tsc = new \_TeachingSchedule();

tsc.TeachingScheduleID = dr["TeachingScheduleID"].ToString();

tsc.CourseName = dr["CourseName"].ToString();

tsc.CollegeName = dr["CollegeName"].ToString();

tsc.StartWeek = (int)dr["StartWeek"];

tsc.TotalHour = (int)dr["TotalHour"];

tsc.GradeYear = dr["GradeYear"].ToString();

tsc.SemesterNo = dr["SemesterNo"].ToString();

tsc.MajorName = dr["MajorName"].ToString();

tsc.UserName = dr["UserName"].ToString();

tsc.Status = (int)dr["Status"];

tsc.Class = new List<Model.Class>();

BLL.Class bClass = new BLL.Class();

string[] classes = dr["Class"].ToString().Split('|');

for (int j = 0; j < classes.Length; j++)

{

Model.Class mClass = bClass.GetModel(classes[j]);

if (mClass != null)

{

tsc.Class.Add(mClass);

}

}

tSchedules.Add(tsc);

}

string json = JsonConvert.SerializeObject(tSchedules).ToLower();

return "{\"ret\": 0, \"errMsg\": \"\", \"teachingschedules\": " + json + "}";

}

private string news(string data) {

JObject jsonData = JObject.Parse(data);

string year = jsonData["year"].ToString();

string strSql = " SELECT TeachingScheduleID, Course.Name as CourseName, College.Name as CollegeName, StartWeek, TotalHour, Grade.Year as GradeYear, Semester.No as SemesterNo, Major.Name as MajorName, TeachingSchedule.Class, Users.RealName as UserName, Status " +

" FROM TeachingSchedule " +

" LEFT JOIN TeachingPlan ON TeachingPlan.PlanID = TeachingSchedule.PlanID " +

" LEFT JOIN Course ON Course.CourseID = TeachingPlan.CourseID " +

" LEFT JOIN Semester ON Semester.SemesterID = TeachingPlan.SemesterID " +

" LEFT JOIN MajorSchedule ON MajorSchedule.MajorScheduleID = Semester.MajorScheduleID " +

" LEFT JOIN Major ON Major.MajorID = MajorSchedule.MajorID " +

" LEFT JOIN Grade ON Grade.GradeID = MajorSchedule.GradeID " +

" LEFT JOIN Users ON Users.UserID = TeachingSchedule.UserID " +

" LEFT JOIN College ON College.CollegeID = Course.CollegeID " +

" WHERE Grade.Year LIKE '%"+ year +"%' AND Status = 0 ";

List<\_TeachingSchedule> tSchedules = new List<\_TeachingSchedule>();

DataSet ds = DbHelperSQL.Query(strSql);

for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

{

DataRow dr = ds.Tables[0].Rows[i];

\_TeachingSchedule tsc = new \_TeachingSchedule();

tsc.TeachingScheduleID = dr["TeachingScheduleID"].ToString();

tsc.CourseName = dr["CourseName"].ToString();

tsc.CollegeName = dr["CollegeName"].ToString();

tsc.StartWeek = (int)dr["StartWeek"];

tsc.TotalHour = (int)dr["TotalHour"];

tsc.GradeYear = dr["GradeYear"].ToString();

tsc.SemesterNo = dr["SemesterNo"].ToString();

tsc.MajorName = dr["MajorName"].ToString();

tsc.UserName = dr["UserName"].ToString();

tsc.Status = (int)dr["Status"];

tsc.Class = new List<Model.Class>();

BLL.Class bClass = new BLL.Class();

string[] classes = dr["Class"].ToString().Split('|');

for (int j = 0; j < classes.Length; j++)

{

Model.Class mClass = bClass.GetModel(classes[j]);

if (mClass != null)

{

tsc.Class.Add(mClass);

}

}

tSchedules.Add(tsc);

}

string json = JsonConvert.SerializeObject(tSchedules).ToLower();

return "{\"ret\": 0, \"errMsg\": \"\", \"teachingschedules\": " + json + "}";

}

public bool IsReusable

{

get

{

return false;

}

}

}

}

移动端（iOS）数据访问类：创建一个NetHelper类，此类定义的接口用于移动端（iOS）的访问数据。此访问采用http协议，当客服端向服务器发出请求，服务端就会发出响应信息去发给客服端，同时系统采用post的请求方式，将请求信息放入请求体中。

typedef void(^netBlock)(id responseObject);

typedef void(^errBlock)(id err);

#import <Foundation/Foundation.h>

@interface NetHelper : NSObject

+ (void)postRequest:(NSString \*)urlStr withActionStr:(NSString \*)action withDataStr:(NSString \*)dataStr withNetBlock:(netBlock)block withErrBlock:(errBlock)errblock;

@end

+ (void)postRequest:(NSString \*)urlStr withActionStr:(NSString \*)action withDataStr:(NSString \*)dataStr withNetBlock:(netBlock)block withErrBlock:(errBlock)errblock

{

NSURLSession \*session = [NSURLSession sharedSession];

NSURL \*url = [NSURL URLWithString:urlStr];

NSMutableURLRequest \*request = [NSMutableURLRequest requestWithURL:url];

request.HTTPMethod = @"POST";

NSString \*body = [NSString stringWithFormat:@"action=%@&data=%@",action,[dataStr base64EncodedString]];

request.HTTPBody = [body dataUsingEncoding:NSUTF8StringEncoding];

NSURLSessionDataTask \*dataTask = [session dataTaskWithRequest:request completionHandler:^(NSData \* \_Nullable data, NSURLResponse \* \_Nullable response, NSError \* \_Nullable error) {

NSDictionary \*dict = [NSJSONSerialization JSONObjectWithData:data options:kNilOptions error:nil];

NSLog(@"%@",dict);

block(dict);

}];

[dataTask resume];

}

## 5.3 定义业务逻辑层接口

登陆接口：创建一个user类，此类定义的接口用于移动端（iOS）的登陆接口。此访问采用http协议，当客服端向服务器发出请求，服务端就会发出响应信息去发给客服端，系统采用post的请求方式，将请求信息放入请求体中。

public class user : IHttpHandler

{

public void ProcessRequest(HttpContext context)

{

context.Response.ContentType = "text/plain";

String action = context.Request.Form["action"];

String result = "{\"ret\": 9, \"errMsg\": \"action error\"}";

if (action != null)

{

String data = context.Request.Form["data"];

byte[] bytes = Convert.FromBase64String(data);

String decodedStr = UTF8Encoding.Default.GetString(bytes);

switch (action)

{

case "login":

result = login(decodedStr);

break;

case "register":

result = register(decodedStr);

break;

case "changepwd":

result = changepwd(decodedStr);

break;

default:

break;

}

}

context.Response.Clear();

context.Response.Write(result);

}

private String login(String data) {

String result = "";

JObject loginData = JObject.Parse(data);

String username = loginData["username"].ToString();

String password = loginData["password"].ToString();

BLL.Users bUser = new BLL.Users();

if (bUser.Exists(username, password)) {

Model.Users userInfo = bUser.GetModelByName(username);

String json = JsonConvert.SerializeObject(userInfo).ToLower();

result = "{\"ret\": 0, \"errMsg\": \"\", \"user\": "+ json +"}";

} else {

result = "{\"ret\": 1, \"errMsg\": \"username or password error\"}";

}

return result;

}

提交选课接口：创建一个teachingschedule类，此类定义的接口用于移动端（iOS）的选课接口。此访问采用http协议，当客服端向服务器发出请求，服务端就会发出响应信息去发给客服端，同时系统采用post的请求方式，将请求信息放入请求体中。

public class teachingschedule : IHttpHandler

{

public void ProcessRequest(HttpContext context)

{

context.Response.ContentType = "text/plain";

String action = context.Request.Form["action"];

String result = "{\"ret\": 9, \"errMsg\": \"action error\"}";

if (action != null)

{

String data = context.Request.Form["data"];

byte[] bytes = Convert.FromBase64String(data);

String decodedStr = UTF8Encoding.Default.GetString(bytes);

// 解决中文 UTF8 乱码

byte[] buffer2 = Encoding.Convert(Encoding.UTF8, Encoding.Default, bytes, 0, bytes.Length);

decodedStr = Encoding.Default.GetString(buffer2, 0, buffer2.Length);

switch (action)

{

case "list":

result = list(decodedStr);

break;

case "add":

result = add(decodedStr);

break;

case "edit":

result = edit(decodedStr);

break;

case "delete":

result = delete(decodedStr);

break;

case "check":

result = check(decodedStr);

break;

case "select":

result = select(decodedStr);

break;

case "giveup":

result = giveup(decodedStr);

break;

case "enable":

result = enable(decodedStr);

break;

case "disable":

result = disable(decodedStr);

break;

default:

break;

}

}

context.Response.Clear();

context.Response.Write(result);

}

private string select(string data)

{

BLL.TeachingSchedule bTeachingSchedule = new BLL.TeachingSchedule();

JObject jsonData = JObject.Parse(data);

String tscId = jsonData["teachingscheduleid"].ToString();

string userId = jsonData["userid"].ToString();

BLL.Users bUsers = new BLL.Users();

Model.Users mUser = bUsers.GetModel(userId);

if (mUser != null)

{

if (mUser.Role == 3)

{

Model.TeachingSchedule mTSchedul = bTeachingSchedule.GetModel(tscId);

if (mTSchedul != null)

{

if (mTSchedul.Status != 0)

{

return "{\"ret\": 3, \"errMsg\": \"can not select\"}";

}

mTSchedul.Status = 1;

mTSchedul.UserID = userId;

bTeachingSchedule.Update(mTSchedul);

return "{\"ret\": 0, \"errMsg\": \"\"}";

}

else

{

return "{\"ret\": 2, \"errMsg\": \"id error\"}";

}

}

else

{

return "{\"ret\": 3, \"errMsg\": \"perssmsion denyed\"}";

}

}

else

{

return "{\"ret\": 2, \"errMsg\": \"user error\"}";

}

}

退选课程接口：创建一个teachingschedule类，此类定义的接口用于移动端（iOS）的退选课程接口。此访问采用http协议，当客服端向服务器发出请求，服务端就会发出响应信息去发给客服端，同时系统采用post的请求方式，将请求信息放入请求体中。

private string giveup(string data)

{

BLL.TeachingSchedule bTeachingSchedule = new BLL.TeachingSchedule();

JObject jsonData = JObject.Parse(data);

String tscId = jsonData["teachingscheduleid"].ToString();

string userId = jsonData["userid"].ToString();

BLL.Users bUsers = new BLL.Users();

Model.Users mUser = bUsers.GetModel(userId);

if (mUser != null)

{

if (mUser.Role == 3)

{

Model.TeachingSchedule mTSchedul = bTeachingSchedule.GetModel(tscId);

if (mTSchedul != null)

{

if (mTSchedul.Status != 1)

{

return "{\"ret\": 3, \"errMsg\": \"can not give up\"}";

}

if (mTSchedul.UserID != userId) {

return "{\"ret\": 3, \"errMsg\": \"perssmsion denyed, you can't operate others courses\"}";

}

mTSchedul.Status = 0;

mTSchedul.UserID = "";

bTeachingSchedule.Update(mTSchedul);

return "{\"ret\": 0, \"errMsg\": \"\"}";

}

else

{

return "{\"ret\": 2, \"errMsg\": \"id error\"}";

}

}

else

{

return "{\"ret\": 3, \"errMsg\": \"perssmsion denyed\"}";

}

}

else

{

return "{\"ret\": 2, \"errMsg\": \"user error\"}";

}

}

修改密码接口：创建一个teachingschedule类，此类定义的接口用于移动端（iOS）的退选课程接口。此访问采用http协议，当客服端向服务器发出请求，服务端就会发出响应信息去发给客服端，同时系统采用post的请求方式，将请求信息放入请求体中。

private String changepwd(String data) {

String result = "";

JObject loginData = JObject.Parse(data);

String username = loginData["username"].ToString();

String password = loginData["password"].ToString();

String newPassWd = loginData["newpassword"].ToString();

BLL.Users bUser = new BLL.Users();

if (bUser.Exists(username, password))

{

Model.Users userInfo = bUser.GetModelByName(username);

userInfo.Password = newPassWd;

if (bUser.Update(userInfo))

{

String json = JsonConvert.SerializeObject(userInfo).ToLower();

result = "{\"ret\": 0, \"errMsg\": \"\", \"user\": " + json + "}";

}

else {

result = "{\"ret\": 1, \"errMsg\": \"exec db operation error\"}";

}

}

else

{

result = "{\"ret\": 1, \"errMsg\": \"username or password error\"}";

}

return result;

}

## 5.4 定义工厂类

创建移动端（iOS）弹出窗口类：创建一个AlertNotice类，用于移动端所有弹出信息的显示。具体创建如下：

AlertNotice.h文件

typedef void(^clickCallBack)();

#import <Foundation/Foundation.h>

typedef NS\_ENUM(NSInteger,AlertNoticeType)

{

kEnterThePwd = 0,

kEnterOutSystem = 1,

kEnterCommitCourse = 2,

kEnterCancle = 3

};

@interface AlertNotice : NSObject

+(void)showAlert: (AlertNoticeType)type withTitle:(NSString \*)title withContent:(NSString \*)content withVC:(UIViewController \*)vc clickLeftBtn:(clickCallBack)leftCallBack clickRightBtn:(clickCallBack)rightCallBack;

+ (void)showAlertNotType :(NSString \*)title withContent:(NSString \*)content withVC:(UIViewController \*)vc clickLeftBtn:(clickCallBack)CallBack;

@end

AlertNotice.m文件

#import "AlertNotice.h"

@implementation AlertNotice

+ (AlertNotice \*)shareAlert

{

static AlertNotice \*alert = nil;

if (!alert)

{

alert = [[AlertNotice alloc] init];

}

return alert;

}

+ (void)showAlert:(AlertNoticeType)type withTitle:(NSString \*)title withContent:(NSString \*)content withVC:(id)vc clickLeftBtn:(clickCallBack)leftCallBack clickRightBtn:(clickCallBack)rightCallBack

{

[[AlertNotice shareAlert] showAlert:type withTitle:title withContent:content withVC:vc clickLeftBtn:leftCallBack clickRightBtn:rightCallBack];

}

+ (void)showAlertNotType:(NSString \*)title withContent:(NSString \*)content withVC:(UIViewController \*)vc clickLeftBtn:(clickCallBack)CallBack

{

[[AlertNotice shareAlert] showAlertNotType:title withContent:content withVC:vc clickLeftBtn:CallBack];

}

- (void)showAlertNotType:(NSString \*)title withContent:(NSString \*)content withVC:(UIViewController \*)vc clickLeftBtn:(clickCallBack)CallBack

{

UIAlertController \*alertController = [UIAlertController alertControllerWithTitle:title message:content preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction \*action = [UIAlertAction actionWithTitle:@"确定" style:0 handler:^(UIAlertAction \* \_Nonnull action) {

CallBack();

}];

[alertController addAction:action];

[vc presentViewController:alertController animated:YES completion:nil];

}

- (void)showAlert:(AlertNoticeType)type withTitle:(NSString \*)title withContent:(NSString \*)content withVC:(id)vc clickLeftBtn:(clickCallBack)leftCallBack clickRightBtn:(clickCallBack)rightCallBack

{

NSString \*leftBtnTitle;

NSString \*rightBtnTitle;

switch (type)

{

case 1:

title = @"提示";

content = @"是否退出登录";

leftBtnTitle = @"确定";

rightBtnTitle = @"取消";

break;

case 0:

title = @"提示";

content = @"是否确定修改密码";

leftBtnTitle = @"确定";

rightBtnTitle = @"取消";

break;

case 2:

leftBtnTitle = @"确定";

rightBtnTitle = @"取消";

break;

case 3:

title = @"提示";

leftBtnTitle = @"确定";

rightBtnTitle = @"取消";

default:

break;

}

UIAlertController \*alertController = [UIAlertController alertControllerWithTitle:title message:content preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction \*leftAction = [UIAlertAction actionWithTitle:leftBtnTitle style:0 handler:^(UIAlertAction \* \_Nonnull action) {

leftCallBack();

}];

UIAlertAction \*rightAction = [UIAlertAction actionWithTitle:rightBtnTitle style:0 handler:^(UIAlertAction \* \_Nonnull action) {

rightCallBack();

}];

[alertController addAction:leftAction];

[alertController addAction:rightAction];

[vc presentViewController:alertController animated:YES completion:nil];

}

# 6 系统测试和维护

## 6.1 系统测试

## 6.2 系统维护

# 7 总结和展望

# [参考文献](论文格式摸板(2007).doc)

[1] 索传军.论电子资源服务绩效评估的含义与影响因素分析[J].图书情报知识,2005,(5):50-53

[2] 彭冬莲,彭备芳. 读者满意度测评方法研究[J].图书馆学研究,2005,(8):52-56

[3] 张四新.读者满意度评价模式初探[J].图书馆工作与研究,2003,(4):45-48

[4] 杨春,刘富军,孟立新.读者满意度的模型构筑[J].情报杂志,2005,(7):35-40

[5] 张榜树.试论图书馆的读者满意度[J].情报理论与实践,2004,(1):52-54

[6] 徐飚.市场调查学[M].北京：北京工业大学出版社,2003.06

# 致 谢

本文是在陈丁老师的热情关心和指导下完成的，他渊博的知识和严谨的治学态度使我受益匪浅，对顺利完成本课题起到了极大的作用。在此向他表示我最衷心的感谢！

在论文完成过程中，本人还得到了李飞老师和何建波同学的热心帮助，本人向他们表示深深的谢意！

最后向在百忙之中评审本文的各位专家、老师表示衷心的感谢！

姓 名：张梓萌 性别：男

出生年月：1994年1月10日 民族：汉

E-mail: 770715576@qq.com

# 声 明

本论文的工作是2015年12月至2016 年6月在成都信息工程大学信息安全工程学院完成的。文中除了特别加以标注地方外，不包含他人已经发表或撰写过的研究成果，也不包含为获得成都信息工程大学或其他教学机构的学位或证书而使用过的材料。

关于学位论文使用权和研究成果知识产权的说明：

本人完全了解成都信息工程大学有关保管使用学位论文的规定，其中包括：

（1）学校有权保管并向有关部门递交学位论文的原件与复印件。

（2）学校可以采用影印、缩印或其他复制方式保存学位论文。

（3）学校可以学术交流为目的复制、赠送和交换学位论文。

（4）学校可允许学位论文被查阅或借阅。

（5）学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

除非另有科研合同和其他法律文书的制约，本论文的科研成果属于成都信息工程大学。

特此声明！

作者签名：张梓萌

2016年05月15日