Nanjing Which University

选课系统

软件体系结构

描述文档

队长: 王　琨 121250151

队员: 沈静怡 121250121

王天宇 121250156

吴晓晨 121250167

修订历史记录

日期	版本	说明	作者
2013-10-14	<V1.0>	最初版本	Ex 咖喱棒

目录

修订历史记录 ····· 1

目录 ····· 2

1 引言 ····· 3

1.1 编制目的 ····· 3
1.2 词汇表 ····· 3
1.3 参考资料 ····· 3

2 产品概述 ····· 3

3 逻辑视角 ····· 3

3.1 简要概述 ····· 3
3.2 逻辑视角包图 ····· 3

4 组合视角 ····· 5

4.1 开发包图 ····· 5
4.2 运行时进程 ····· 7
4.3 物理部署 ····· 8

5 接口视角 ····· 8

5.1 模块的职责 ····· 8
5.2 用户界面层的分解 ····· 11
5.3 业务逻辑层的分解 ····· 13
5.4 数据层的分解 ····· 15

6 信息视角 ····· 16

6.1 数据持久化对象 ····· 16
6.2 数据库表 ····· 17

1. 引言

1.1 编制目的

本报告详细完成对 Nanjing Which University 选课系统的概要设计，达到指导详细设计开发的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员以及最终用户而编写，是了解系统的导航。

1.2 词汇表

词汇名称	词汇含义	备注
NJWU	Nanjing Which University	
PO	persistant object 持久对象	
VO	value object 数值对象	
teacher	任课老师	
insteacher	院系教务老师	
schteacher	学校教务老师	

1.3 参考资料

（1）NJWU 选课系统用例文档

（2）NJWU 选课系统需求规格说明文档

（3）IEEE 标准

2. 产品概述

参考 NJWU 选课系统用例文档和 NJWU 选课系统需求规格说明文档中对产品的概括描述。

3. 逻辑视角

3.1 简要概述

NJWU 选课系统中，选择了分层体系结构风格，将系统分为三层（展示层、业务逻辑层、数据层）能够很好地示意整个高层抽象。展示层包含 GUI 页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图 1 和图 2 所示。

3.2 逻辑视角包图

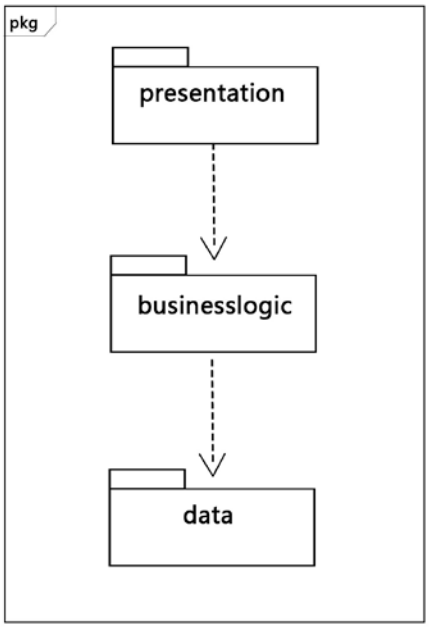

图 1：参照体系结构风格的包图表达视角

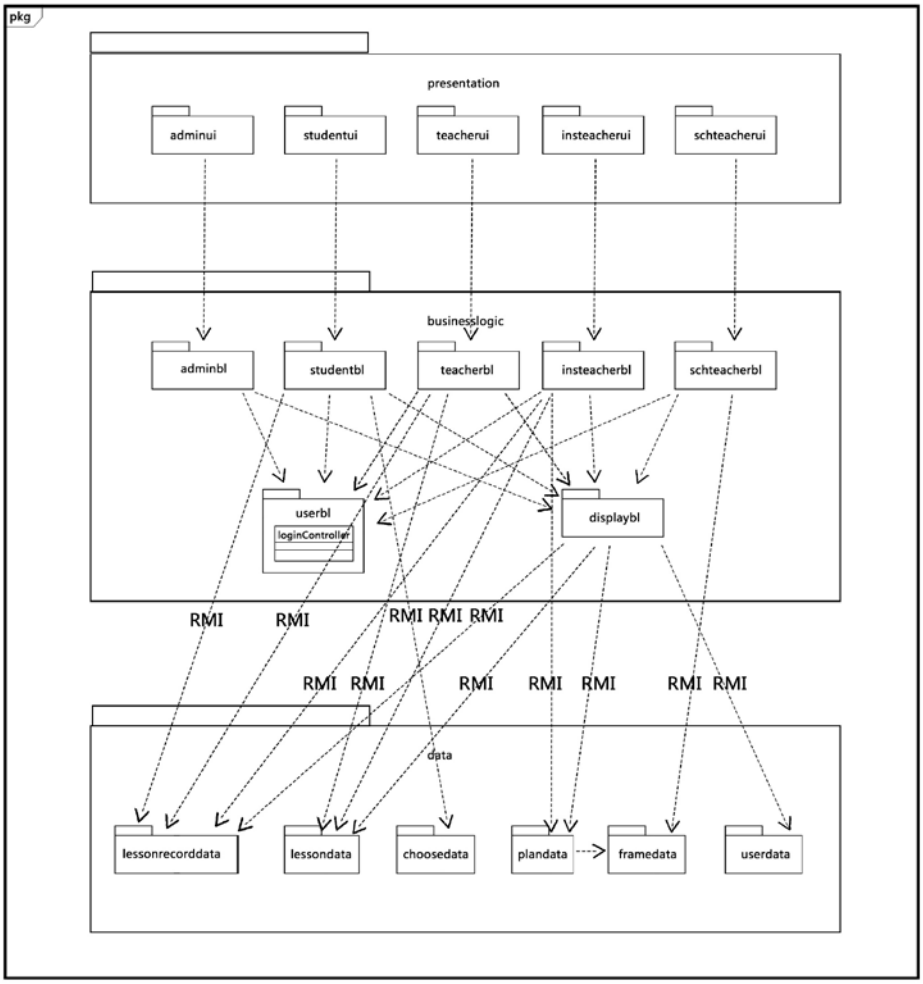

图 2： 软件体系结构逻辑设计方案

4. 组合视角

4.1 开发包图

NJWU 选课系统的最终开发包如表 1 所示。

表 1: NJWU 选课系统的最终开发包设计

开发（物理）包	依赖的其他开发包
mainui	adminui,studentui,teacherui,insteacherui,schteacherui,vo
adminui	adminblservice,界面类库包,vo
adminblservice	
adminbl	adminblservice,userbl,displaybl,userdataservice,po
studentui	studentblservice,界面类库包,vo
studentblservice	
studentbl	studentblservice,userbl,displaybl,lessonrecorddataservice,choosedataservice,po
teacherui	teacherblservice,界面类库包,vo
teacherblservice	
teacherbl	teacherblservice,userbl,displaybl,lessondataservice,lessonrecorddataservice,po

insteacherui	insteacherblservice,界面类库包,vo
insteacherblservice	
insteacherbl	insteacherblservice,userbl,displaybl,plandataservice,lessonrecorddataservice, lessondataservice,po
schteacherui	schteacherblservice,界面类库包,vo
schteacherblservice	
schteacherbl	schteacherblservice,userbl,displaybl,framdataservice,po
userbl	userdataservice
displaybl	userdataservice,lessonrecorddataservice,lessondataservice,plandataservice,less ondataservice,po
userdataservice	Java RMI,po
userdata	userdataservice,Java RMI,po,databaseutility
lessonrecorddataservice	Java RMI,po
lessonrecorddata	lessonrecorddataservice,JavaRMI,po
lessondataservice	Java RMI,po
lessondata	lessondataservice,Java RMI,po,databaseutility
plandataservicce	Java RMI,po
plandata	plandataservice,framedata,Java RMI,po,databaseutility
framedataservice	Java RMI,po
framedata	framedataervice,Java RMI,po,databaseutility
choosedataservice	Java RMI,po
choosedata	choosedataservice,Java RMI,po,databaseutility
vo	
po	
databaseutility	JDBC
Java RMI	
界面类库包	

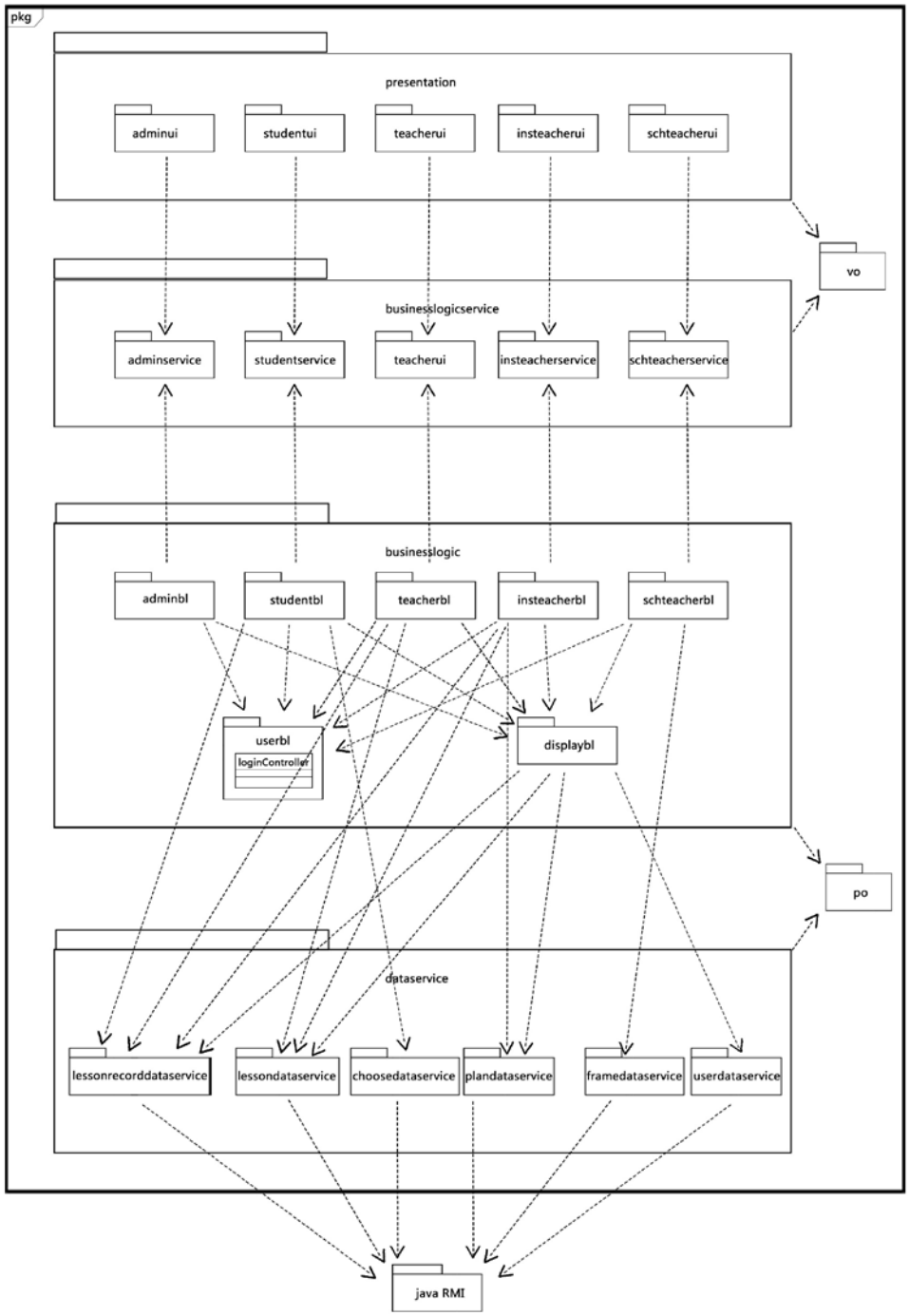

图 3 连锁商店管理系统客户端开发包图

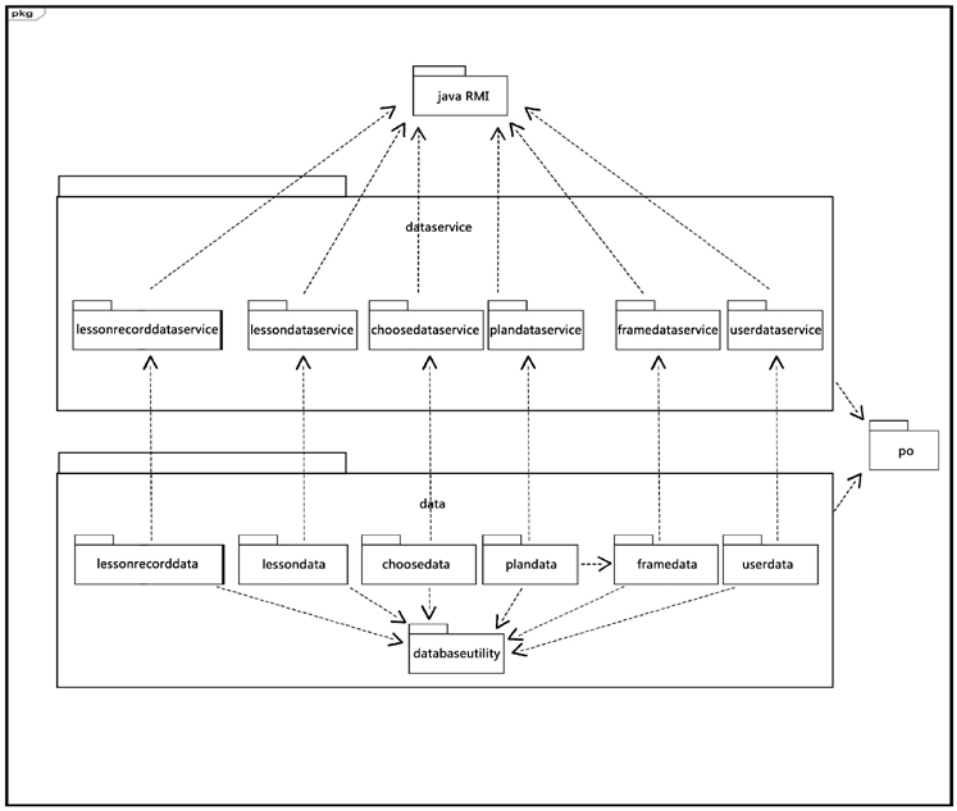

图 4 连锁商店管理系统服务器端开发包图

4.2 运行时进程

在 NJWU 选课系统中，会有多个客户端进程和一个服务器端进程，其进程图如图所示。结合部署图，客户端进程是在客户端机器上进行，服务器端进程在服务器端机器上运行。

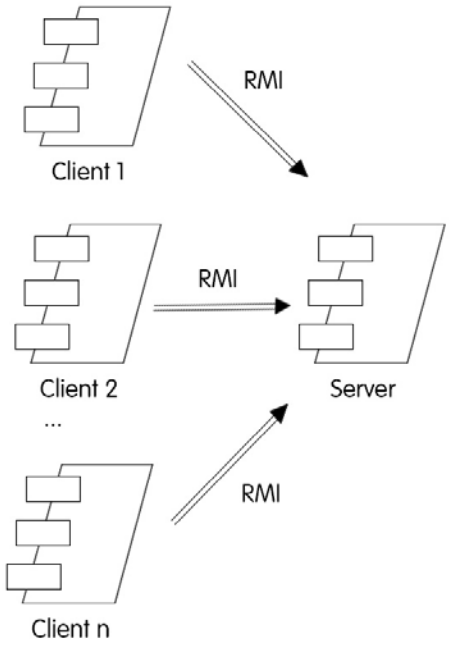

图 5：进程图

4.3 物理部署

NJWU 选课系统中客户端构件是放在客户端机器上，服务器端构件是放在服务器端机器上。在客户端节点上，还要部署 RMIStub 构件。由于 Java RMI 构件属于 JDK 7.0 的一部分。所以，在系统 JDK 环境已经设置好的情况下，不需要再独立部署。部署图如图所示。

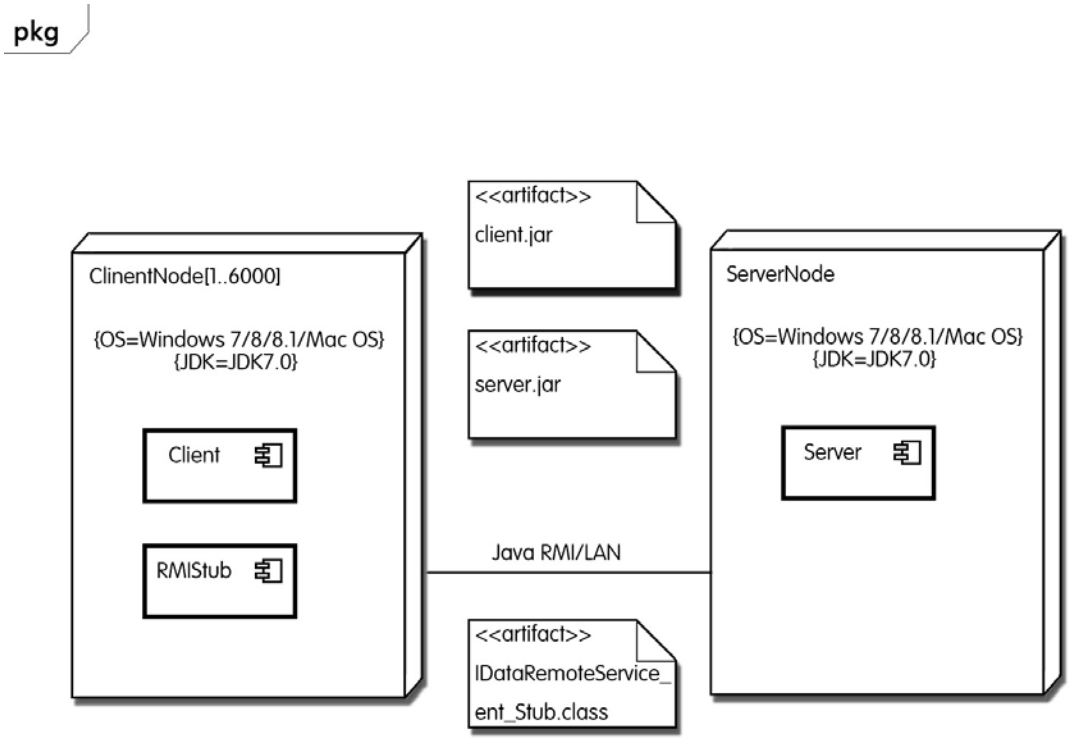

图 6 物理部署

5.接口视角

5.1 模块的职责

客户端模块和服务器模块视图分别如图和图所示。客户端各层与服务器端各层的职责分别如表和表所示。

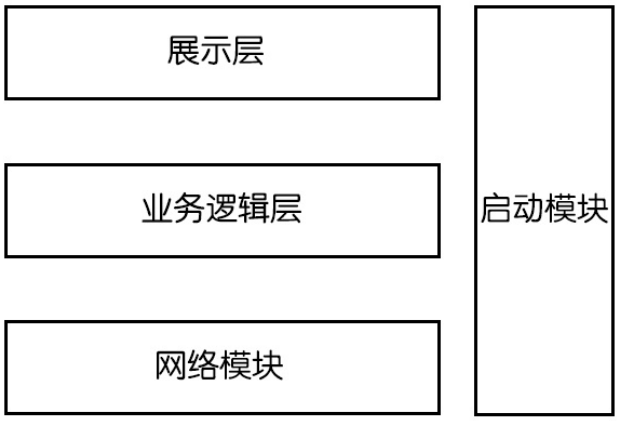

图 7 客户端模块视图

图 8 服务器端模块视图

表 2 客户端各层的职责

层	职责
启动模块	负责初始化网络通信机制，启动用户界面
用户界面层	基于窗口的 NJWU 客户端用户界面
业务逻辑层	对于用户界面的输入进行响应并进行处理业务逻辑
客户端网络模块	利用 Java RMI 机制查找 RMI 服务

表 3 服务器端各层的职责

层	职责
启动模块	负责初始化网络通信机制，启动用户界面
数据层	负责数据的持久化及数据访问接口
服务器端网络模块	利用 Java RMI 机制开启 RMI 服务，注册 RMI 服务

每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层与层调用的接口如表所示。

表 4 层之间调用的接口

接口	服务调用方	服务提供方
UserBLService AdminBLService StudentBLService InsteacherBLService SchteacherBLService TeacherBLService	客户端展示层	客户端业务逻辑层
UserDataService LessonRecordDataService LessonDataService PlanDataService FrameDataService ChooseDataService	客户端业务逻辑层	服务器端数据层

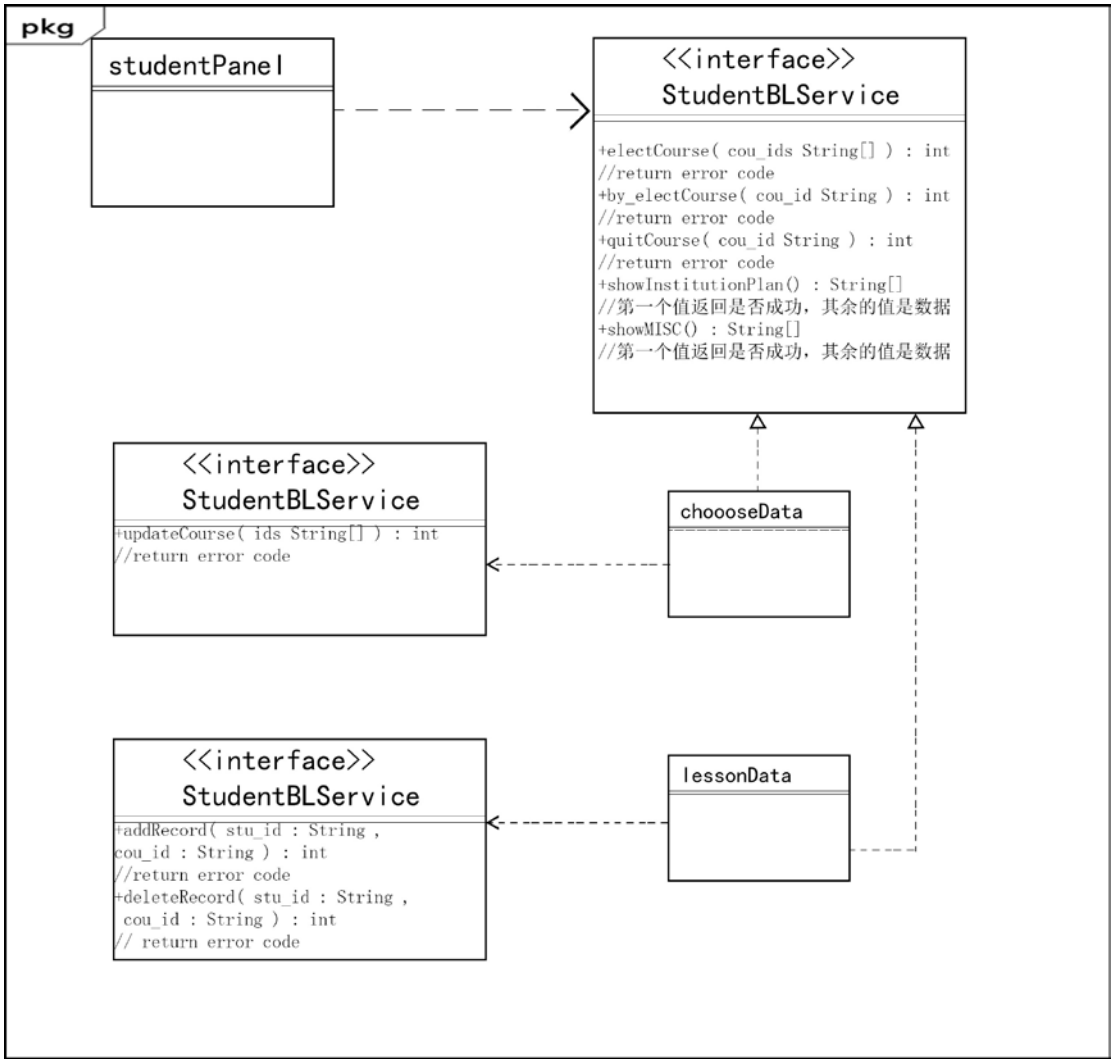

图 9 学生层之间调用的接口

5.2 用户界面层的分解

根据需求，系统存在 35 个界面包括登陆界面、4 个管理员界面、9 个学生界面、8 个学校教务老师界面、5 个任课老师界面、8 个院系教务老师界面。界面跳转图如图所示

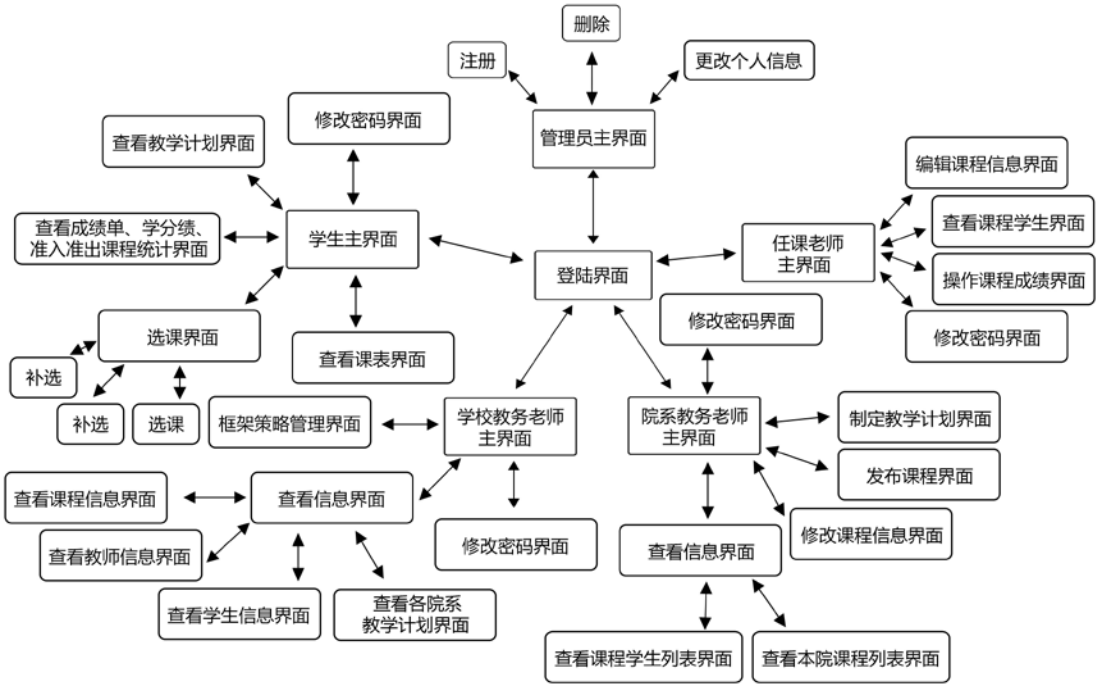

图 10 用户界面跳转

用户界面类如图所示

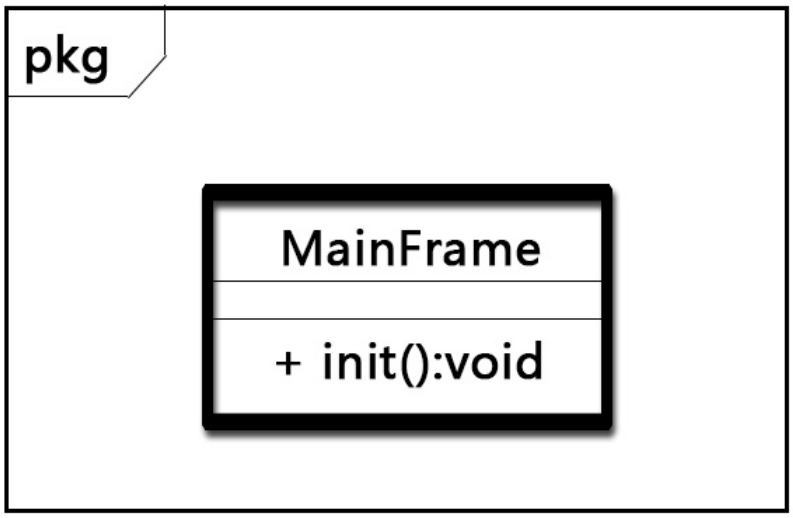

图 11 用户界面类

5.2.1 用户界面层模块的职责

如表所示为用户界面层模块的职责

表 5 用户界面层模块的职责

模块	职责
MainFrame	界面 Frame,负责界面的显示和界面的跳转

5.2.2 用户界面层模块的接口规范

用户界面层模块的接口规范如表所示。

表 6 用户界面层模块的接口规范

Mainframe	语法	init(args:String[])
	前置条件	无
	后置条件	显示 Frame 以及 LoginPanel

用户界面层需要的服务接口如表所示

表 7 用户界面层模块需要的服务接口

服务名	服务
businesslogicservice.LoginBLService	登陆界面的业务逻辑接口
Businesslogicservice.*BLService	每个类型用户都有一个相应的业务逻辑接口

5.2.3 用户界面模块设计原理

用户界面利用 Java 的 Swing 和 AWT 库来实现。

5.3 业务逻辑层的分解

业务逻辑层包括多个针对界面的业务逻辑处理对象，例如，Student 对象负责处理学生功能的业务逻辑，Admin 对象负责处理管理员功能的业务逻辑。业务逻辑的设计如图所示：

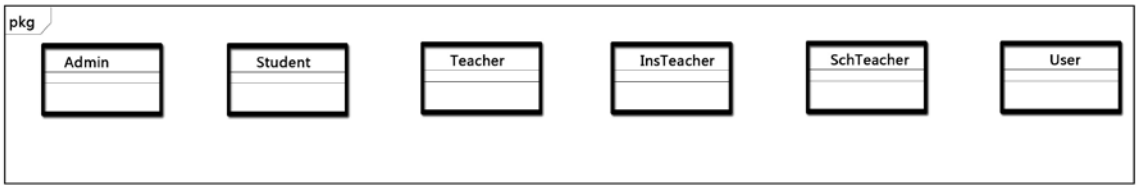

图 12 业务逻辑层的设计

5.3.1 业务逻辑层模块的职责

表 8 业务逻辑层模块的职责

模块	职责
adminbl	负责实现管理员的相关服务
studentbl	负责实现学生的相关服务
teacherbl	负责实现任课老师的相关服务
insteacherbl	负责实现院系教务老师的相关服务
schteacherbl	负责实现学校教务老师的相关服务
userbl	负责实现每种类型用户的登陆、注销以及修改密码的服务
Displaybl	负责实现各种信息的查看，包括课程信息、学生信息、教师信息。

5.3.2 业务逻辑层模块的接口规范

这里以 userbl 和 adminbl 的接口规范为例，如表 和表 所示

表 9 userbl 模块的接口规范

提供的服务(供接口）		
User.Login	语法	Public boolean login(int id, String password);
	前置条件	id 与 password 输入符合规则
	后置条件	查找是否存在相应的 User，根据输入的 password 返回登录验证的结果，包括用户类型以及用户

		Id,并生成相应登录信息，跳转到相应类型用户界面
User.changePassword	语法	Public boolean changePassword(String old, String new);
	前置条件	用户已登录，且新旧密码符合输入规则
	后置条件	修改当前用户的密码，更新数据库
想要的服务（需接口）		
服务名		服务
DatabaseFactory.getStudentDatabase		得到 Student 数据库的服务的引用
DatabaseFactory.getTeacherDatabase		得到 Teacher 数据库服务的的引用
UserDataService.find(int id)		从指定表查找该 id 的用户
UserDataService.update(StudentPO student)		更新指定学生表的信息
UserDataService.update(TeacherPO teacher)		更新指定教师表的信息

表 10 adminbl 模块的接口规范

提供的服务(供接口）		
Admin.addStudent	语法	Public boolean addStudent(StudentPO student);
	前置条件	管理员登陆
	后置条件	模块将该学生信息加入数据库，更新用户列表
Admin.addTeacher	语法	Public boolean addTeacher(TeacherPO teacher);
	前置条件	管理员登陆
	后置条件	模块将该教师信息加入数据库，更新用户列表
Admin.delStudent	语法	Public boolean delStudent(int id);
	前置条件	管理员登陆
	后置条件	模块从数据库删除该学生条目，更新用户列表
Admin.delTeacher	语法	Public boolean delTeacher(int id);
	前置条件	管理员登陆
	后置条件	模块从数据库删除该教师条目，更新用户列表
Admin.modifyTeacher	语法	Public boolean modifyTeacher(TeacherPO teacher)
	前置条件	管理员登陆
	后置条件	模块修改数据库对应 id 教师的信息，更新用户列表
Admin.modifyStudent	语法	Public boolean modifyStudent(StudentPO student)
	前置条件	管理员登陆
	后置条件	模块修改数据库对应 id 学生的信息，更新用户列表
想要的服务（需接口）		
服务名		**服务**
DatabaseFactory.getStudentDatabase		得到 Student 数据库的服务的引用
DatabaseFactory.getTeacherDatabase		得到 Teacher 数据库服务的的引用
UserDataService.findStudentPO(int id)		从学生表查找该 id 的学生
UserDataService.findTeacherPO(int id)		从教师表查找该 id 的教师
UserDataService.update(TeacherPO teacher)		更新教师表的信息

UserDataService.update(StudentPO student)	更新学生表的信息
UserDataService.insert(StudentPO student)	在学生表中插入一学生条目
UserDataService.insert(TeacherPO teacher)	在教师表中插入一教师条目
UserDataService.deleteTeacher(int id)	在指定表中删除指定编号的教师
UserDataService.deleteStudent(int id)	在指定表中删除指定编号的学生

5.4 数据层的分解

数据层主要给业务逻辑层提供数据访问服务，包括对持久化数据的增、删、改、查。Admin 业务逻辑需要的服务由 UserDataService 接口提供。由于持久化数据的保存可能存在多种形式：Txt 文件、序列化文件、数据库等，所示抽象了数据服务。数据层模块的描述具体如图 13 所示：

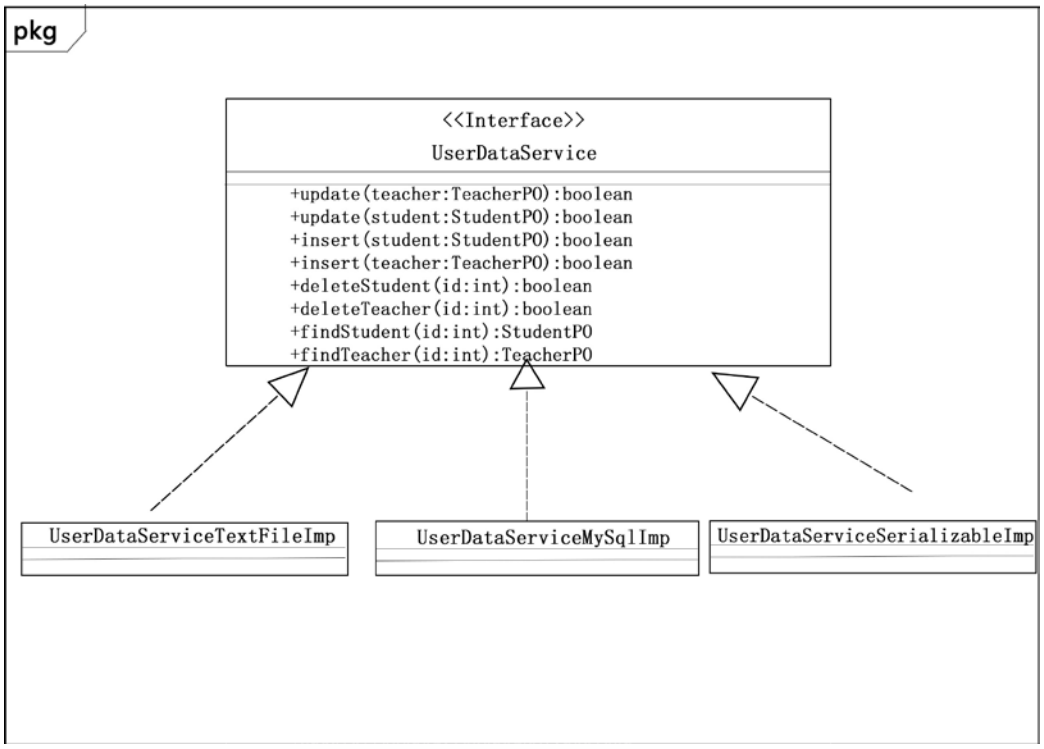

图 13 数据层模块的描述

5.4.1 数据层模块的职责

数据层模块职责如表 11 所示

表 11：数据层模块职责

模块	职责
UserDataService	持久化数据库的接口、提供集体载入、集体保存、增、删、改、查服务
UserDataTextFileImp	基于 Txt 的持久化数据库的接口、提供集体载入、集体保存、增、删、改、查服务
UserDataMySqlImp	基于 Mysql 数据库的持久化数据库的接口、提供集体载入、集体保存、增、删、改、查服务
UserDataSerializableImp	基于可序列化文件的持久化数据库的接口、提供集体载入、集体保存、增、删、改、查服务

5.4.2 数据层模块的接口规范

数据层模块的接口规范如表 12 所示

表 12 数据层模块的接口规范

提供的服务(供接口）		
UserDataService.findStudent	语法	Public StudentPO findStudent(int id);
	前置条件	无
	后置条件	按 id 进行查找返回 StudentPO 结果
UserDataService.findTeacher	语法	Public teacherPO findTeacher(int id);
	前置条件	无
	后置条件	按 id 进行查找返回 TeacherPO 结果
UserDataService.delTeacher	语法	Public boolean delTeacher(int id);
	前置条件	在数据库中存在一个对应 id 的 TeacherPO
	后置条件	删除一个 TeacherPO
UserDataService.delStudent	语法	Public boolean delStudent(int id);
	前置条件	在数据库中存在一个对应 id 的 StudentPO
	后置条件	删除一个 StudentPO
UserDataService.updateTeacher	语法	Public boolean updateTeacher(TeacherPo teacher);
	前置条件	在数据库中存在一个对应 id 的 TeacherPO
	后置条件	更新一个 TeacherPO
UserDataService.updateStudent	语法	Public boolean updateStudent(StudentPO student);
	前置条件	在数据库中存在一个对应 id 的 StudentPO
	后置条件	更新一个 StudentPO
UserDataService.insertTeacher	语法	Public boolean insertTeacher(TeacherPO teacher);
	前置条件	在数据库中不存在一个对应 id 的 TeacherPO
	后置条件	增加一个 TeacherPO
UserDataService.insertStudent	语法	Public boolean insertStudent(StudentPO student);
	前置条件	在数据库中不存在一个对应 id 的 StudentPO
	后置条件	增加一个 StudentPO

6. 信息视角

6.1 数据持久化对象

系统的 PO 类就是对应的相关的实体类，在此只做简单的介绍。

• StudentPO 类包含学生的姓名、学号、密码、院系编号、院系、性别、年级。

• Lesson_abstractPO 类包含抽象课程的院系编号、院系、课程号、所属课程类别编号、所属课程类别、名称、建议学分上限、建议学分下限、课程开始学期、课程结束学期。

• Lesson_uniquePO 类包含课程名、课程号、课程编号（不同于课程号）、教师号、教师姓名、上课学期、上课地点、上课时间（周几，第几节到第几节）、学生上限、当前学生数目、当前状态（0=未发布，1=已发布并提供选课，2=已确认学生名单，3=已结束）

• TeacherPO 类包含教师的姓名、教师工号、密码、所属院系编号（学校教务老师的院系编号为 0），类别（1=学校教务老师，2=院系教务老师，3=任课老师）

• ModulePO 类包含课程模块的编号、名称、学分

• TypePO 类包含课程类别的编号、所属模块编号、类别名称、学分

• Les_RecordPO 类包含一条课程记录的编号、学生学号、课程号以及成绩
• Sel_RecordPO 类包含一条选课记录的编号、学生学号、课程号以及选课时间

例：

持久化用户对象 StudentPO 的定义如图所示：

```java
package po;

import java.io.Serializable;

public class StudentPO implements Serializable {
    int Stu_Id;
    String name;
    String password;
    String institute;
    int Ins_Id;
    String gender;
    int grade;

    public StudentPO(int Stu_Id, String name, String password, int Ins_Id,
            String institute, String gender, int grade) {
        this.Stu_Id = Stu_Id;
        this.name = name;
        this.password = password;
        this.Ins_Id = Ins_Id;
        this.institute = institute;
        this.gender = gender;
        this.gender = gender;
    }

    public String getGender() {
        return gender;
    }

    public int getGrade() {
        return grade;
    }

    public int getIns_Id() {
        return Ins_Id;
    }

    public String getInstitute() {
        return institute;
    }

    public String getName() {
        return name;
    }

    public String getPassword() {
        return password;
    }

    public int getStu_Id() {
        return Stu_Id;
    }
}
```

持久化用户对象 StudentPO 的定义

注：课程分抽象课程与具体课程，原因是同一门课程在不同的院系可能会有不同的信息，包括上课地点、时间、老师、院系，也会存在不同的状态即已发布与未发布，所以这里令 Lesson_uniquePO 实现对应抽象课程的具体化，以抽象课程的课程号作为外键索引。

6.2 数据库表

注：表明 Key 的为主键

表明*的为关联的外键

Student	**Institute**	**Module**	**Sel_Record**
Name	Ins_Id(Key)	Module_Id(Key)	Id(Key)
Stu_Id(Key)	Name	Name	Stu_Id*
Password		Credit	Les_Id*
Ins_Num*			Time
Gender			
Grade			

Lesson_unique	**Teacher**	**Lesson_abstract**	**Type**	**Les_Record**
Name	Name	Ins_Id*	Type_Id(Key)	Les_Num*
Les_Id_Ab*	Tea_Id(Key)	Les_Id_Ab(Key)	Module_Id*	Stu_Num*
Ins_Id*	Password	Type_Id*	Name	Id(Key)
Tea_Id*	Institute	name	Credit	Score
Les_Id(Key)	Type	Min_credit		
term		Max_credit		
Location		Term_start		
Day		Term_end		
Max_Stu_Num				
Cur_Stu_Num				
state				
start				
end				