

Nanjing Which University

选课系统

软件详细设计

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
2013-10-30	<V1.0>	最初版本	Ex 咖喱棒
2013-11-9	<V1.1>	第二次提交版本 增加一些类的方法	Ex 咖喱棒

目录

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.体系结构设计概述.....	3
4.结构视角.....	4
4.1 业务逻辑层的整体结构说明.....	4
4.2 业务逻辑层的分解.....	4
4.2.1 userbl 模块.....	4
4.2.2 adminbl 模块.....	7
4.2.3 studentbl 模块.....	16
4.2.4 teacherbl 模块.....	24
4.2.5 insteacherbl 模块.....	29
4.2.6 schteacherbl 模块.....	37
4.3 数据层的整体结构说明.....	47
4.4 数据库的详细设计.....	48
5.依赖视角.....	51

1.引言

1.1 编制目的

本报告详细完成对南京哪个大学选课系统的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

1.2 词汇表

词汇名称	词汇含义	备注
NJWU	南京哪个大学	
InsTeacher	院系教务老师	
SchTeacher	学校教务处老师	
Teacher	任课老师	此处与前两类老师区分 •
Lesson_abstract	抽象课程	用于指定教学计划
Lesson_unique	具体课程	为抽象课程的具体化

1.3 参考资料

NJWU 选课系统体系结构描述文档 V1.2

NJWU 选课系统需求规格说明文档 V1.1

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.体系结构设计概述

参考 NJWU 选课系统体系结构描述文档中对体系结构设计的概述。

4.结构视角

4.1 业务逻辑层的整体结构说明

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们会添加 businesslogicservice.adminbl.AdminBLService 接口。业务逻辑层和数据层之间添加 dataservice.userdataservice.StudentDataService 接口。

对于业务逻辑层的设计，我们采用按身份分包的方式，并另外分出一个包专门负责数据的展示。这样设计的理由一方面是有利于展示层与业务逻辑层之间的交互，另外一方面也是因为几乎每个身份都拥有展示数据的职能且有许多交叉部分，而需求表明不同的身份的用户职能都各自相关的数据进行增删修改，其职能是分割开了的。所以业务逻辑层中 studentbl,teacherbl,insteacherbl,schteacherbl 以及 adminbl 分别承担了对相关数据进行增删改的职能并同时依赖于 displaybl,而 displaybl 则拥有对数据进行查找并返回的职能。另外 userbl 承担了用户登陆与修改密码的职能也同时被 5 个身份的 bl 所依赖。

4.2 业务逻辑层的分解

业务逻辑层的开发包图见软件体系结构文档图 3

4.2.1 userbl 模块

(1) 模块概述

userbl 模块承担的需求为：

- 1.负责管理用户的登陆与注销。
- 2.负责管理用户对密码的修改。

Userbl 模块的职责及接口参见软件体系结构描述文档表 9

Userbl 模块的设计如表 1 所示：

表 1： user 业务逻辑层详细设计的上下文

输入	需求	参见需求规格说明文档 V1.1 3.3.1
	体系结构	<div>//被 presentation 层调用的接口</div> <div>public interface UserControllerService {</div> <div> //登陆</div> <div> public int login(int id, char[] password);</div> <div> //修改密码</div> <div> public boolean changePassword(char[] old, char[] newPassword);</div> <div> //获取用户信息</div> <div> public PO getInformation();</div> <div>}</div> <div>//调用 dataservice 的接口，方法均定义在父类 DatabaseService 中</div> <div>//学生数据服务，标红为此业务逻辑会用到的方法接口。</div>

		<pre>public interface StudentDataService extends DatabaseService{ public boolean insert(PO po) throws RemoteException; public boolean delete(int id) throws RemoteException; public boolean update(PO po) throws RemoteException; public PO find(int id) throws RemoteException; public ArrayList<PO> find(int condition, int id) throws RemoteException; public ArrayList<PO> findAll() throws RemoteException; } //教师数据服务 public interface TeacherDataService extends DatabaseService{ public boolean insert(PO po) throws RemoteException; public boolean delete(int id) throws RemoteException; public boolean update(PO po) throws RemoteException; public PO find(int id) throws RemoteException; public ArrayList<PO> find(int condition, int id) throws RemoteException; public ArrayList<PO> findAll() throws RemoteException; }</pre>
输出	类图	图 1

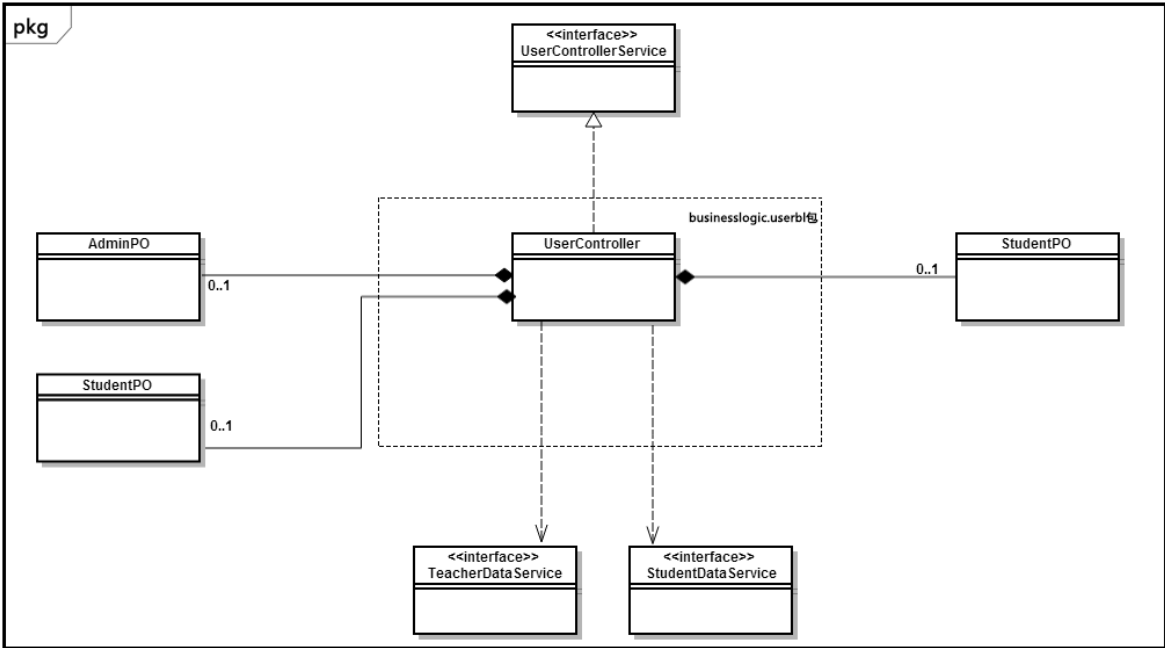


图 1： userbl 模块类图

(2) 模块内部类的接口规范

表 2： userbl 模块各个类的职责

类	职责
UserController	负责用户的登陆（判断用户身份）以及修改

	密码
--	----

表 3: UserController 的接口规范

提供的服务(供接口)		
UserController.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
UserController.change Password	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
UserController.getInfo mation	语法	public PO getInfomation();
	前置条件	系统验证用户登陆成功
	后置条件	系统返回用户的个人信息
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
TeacherDataService.find(int id)		从教师表查找该 id 的用户
StudentDataService.find(int id)		从学生表查找该 id 的用户
StudentDataService.update(PO student)		更新指定学生表的信息
TeacherDataService.update(PO teacher)		更新指定教师表的信息

(3) 业务逻辑层的动态模型

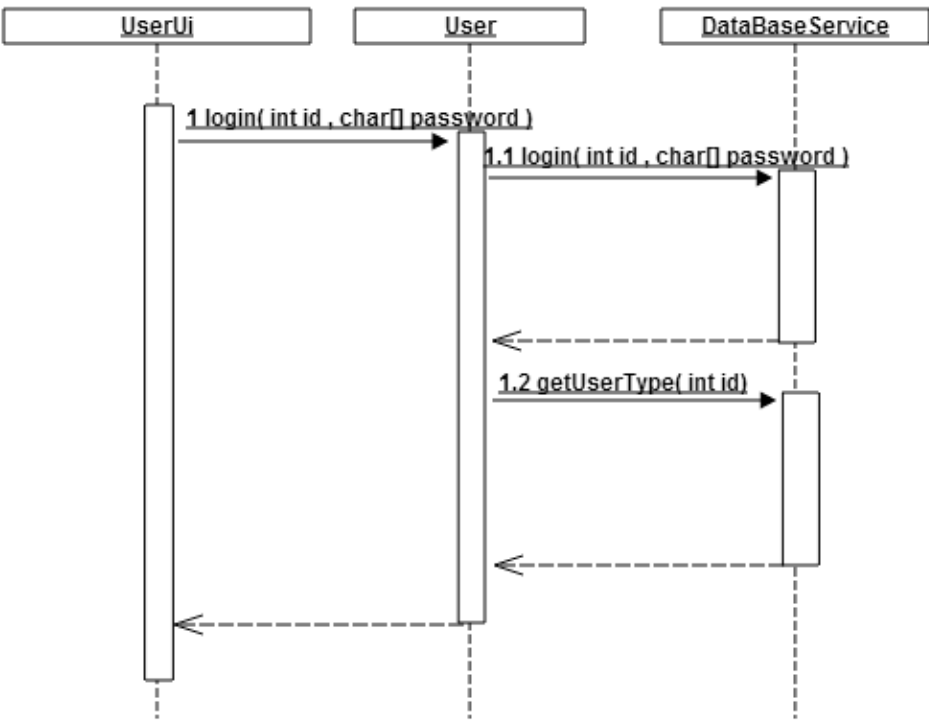


图 2：用户登陆顺序图

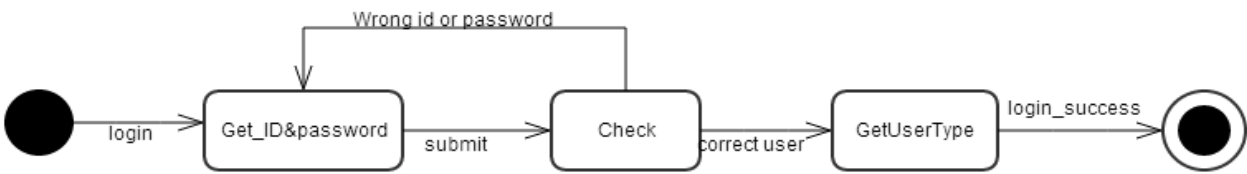


图 3：UserController 对象状态图

4.2.2 adminbl 模块

表 4：adminbl 业务逻辑层详细设计的上下文

输入	需求	参见需求规格说明文档 V1.1 3.2.1-3.2.1 3.3.3（易用性） 需求用例文档用例 1-用例 2
	体系结构	//被 presentation 层调用的接口 public interface AdminblService {


```

public boolean addStudent(StudentPO student);//添加学生
public boolean addTeacher(TeacherPO teacher);//添加老师
public boolean deleteStudent(int id);//删除学生
public boolean deleteTeacher(int id);//删除老师
// public boolean addStudent();
// public boolean addTeacher();
// public boolean deleteTeacher();
// public boolean deleteStudent();
//（此处阴影部分为支持批量导入的方法，先如此设计，而后确定可行性后再决定是否采用，需要依靠 AdminMassManager 来进行实现。
public boolean modifyStudent(StudentPO student);//修改学生
public boolean modifyTeacher(TeacherPO teacher);//修改教师
public StudentPO showStudent(int Stu_id);//搜索指定学号学生
public ArrayList<StudentPO> showStudentofins(int ins_id , int grade);//搜索指定院系学生列表
public TeacherPO showTeacher(int Tea_id);//搜索指定工号老师
public ArrayList<TeacherPO> showTeacherofins(int ins_id);//搜索指定院系老师列表
}
//调用 displayblservice 的接口
public interface StudentInfoDisplayService {
    //根据学号获取学生个人信息
    public StudentPO getStudent(int stu_id);
    //根据院系号获取学生列表
    public ArrayList<StudentPO> getStudentList(int ins_id , int grade);
    //根据课程号获取该课程的所有学生列表
    public ArrayList<StudentPO> getStudentListByLesson(int les_id);
}
public interface TeacherInfoDisplayService {
    public TeacherPO getTeacher(int tea_id);//根据工号获取教师
    public ArrayList<TeacherPO> getTeacherOfIns(int ins_id);//根据院系编号获取该院系教师名单
}
//调用 dataservice 的接口，方法均定义在父类 DatabaseSService 中
//学生数据服务，标红为此业务逻辑会用到的方法接口。
public interface StudentDataService extends DatabaseService{
    public boolean insert(PO po) throws RemoteException;
    public boolean delete(int id) throws RemoteException;
    public boolean update(PO po) throws RemoteException;
    public PO find(int id) throws RemoteException;
    public ArrayList<PO> find(int condition, int id) throws RemoteException;
    public ArrayList<PO> findAll() throws RemoteException;
}
//教师数据服务

```

		<pre>public interface TeacherDataService extends DatabaseService{ public boolean insert(PO po) throws RemoteException; public boolean delete(int id) throws RemoteException; public boolean update(PO po) throws RemoteException; public PO find(int id) throws RemoteException; public ArrayList<PO> find(int condition, int id) throws RemoteException; public ArrayList<PO> findAll() throws RemoteException; }</pre>
输出	类图	见图 2

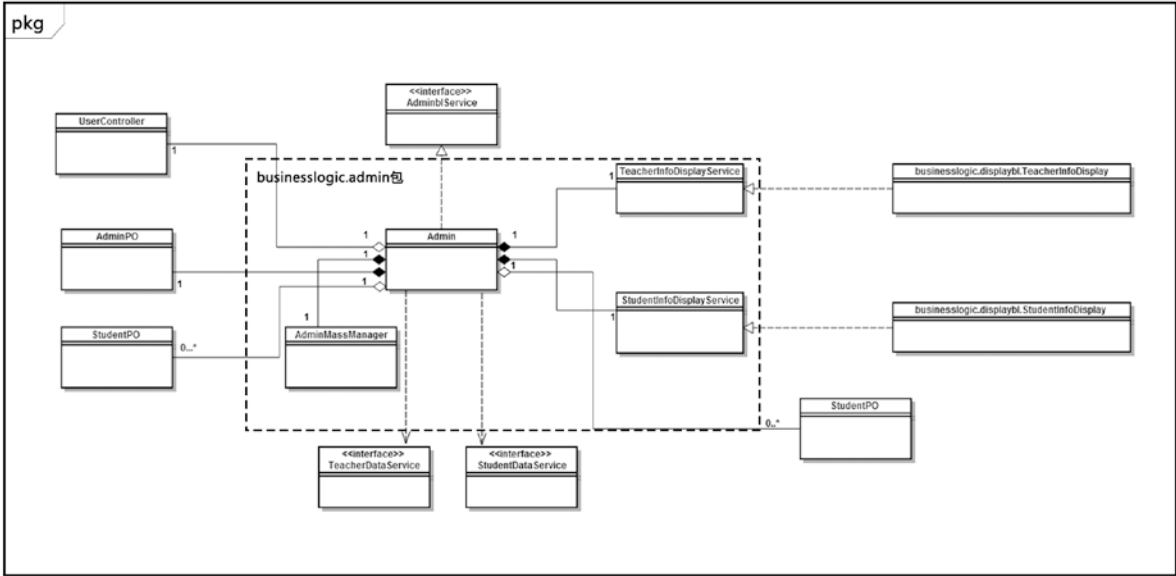


图 4： adminbl 模块类图

表 5： adminbl 模块各个类的职责

类	职责
Admin	负责管理员的功能实现，包括增删查改用户
AdminMassManager	负责用户的批量导入与删除

表 6 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); public boolean addStudent();
	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 参数为空代表调用 AdminMassManager 的批量导入服务
Admin.addTeacher	语法	public boolean addTeacher(TeacherPO teacher); public boolean addTeacher()

	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 参数为空代表调用 AdminMassManager 的批量导入服务
Admin.deleteStudent	语法	public boolean deleteStudent(int id); public boolean deleteStudent();
	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 参数为空代表调用 AdminMassManager 的批量删除服务
Admin.deleteTeacher	语法	public boolean deleteTeacher(int id); public boolean deleteTeacher()
	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 参数为空代表调用 AdminMassManager 的批量删除服务
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
Admin.showStudent	语法	public StudentPO showStudent(int Stu_id);
	前置条件	存在该学号的学生
	后置条件	系统根据学生 id 搜索学生并返回该学生的 PO
Admin.showStudentofins	语法	public StudentPO[] showStudentofins(int ins_id,int grade);
	前置条件	存在该编号的院系,年级号合法
	后置条件	系统根据院系编号以及年级搜索该院系学生并返回这些学生的 PO
Admin.showTeacher	语法	public TeacherPO showTeacher(int Tea_id);
	前置条件	存在该学号的教师
	后置条件	系统根据教师 id 搜索教师并返回该教师的 PO
Admin.showTeacherofins	语法	public TeacherPO[] showTeacherofins(int ins_id);
	前置条件	存在该编号的院系
	后置条件	系统根据院系编号搜索该院系教师并返回这些教师的 PO
想要的服务（需接口）		
服务名		服务
TeacherDisplayService.getTeacher		得到指定教师的数据
StudentDisplayService.getStudent		得到指定学生的数据
StudentDisplayService.getStudentList		得到指定院系学生列表的数据
TeacherDisplayService.getTeacherList		得到指定院系教师列表的数据

AdminMassManager.addStudent	批量添加学生
AdminMassManager.addTeacher	批量添加教师
AdminMassManager.deleteStudent	批量删除学生
AdminMassManager.deleteTeacher	批量删除教师
AdminMassManager.setTeacherList	批量导入教师信息
AdminMassManager.setStudentList	批量导入学生信息
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
StudentDataService.find(int id)	从学生表查找该 id 的学生
TeacherDataService.find(int id)	从教师表查找该 id 的教师
TeacherDataService.update(PO teacher)	更新教师表的信息
StudentDataService.update(PO student)	更新学生表的信息
StudentDataService.insert(PO student)	在学生表中插入一学生条目
TeacherDataService.insert(PO teacher)	在教师表中插入一教师条目
TeacherDataService.delete(int id)	在指定表中删除指定编号的教师
StudentDataService.delete(int id)	在指定表中删除指定编号的学生

表 7 AdminMassManager 的接口规范

提供的服务(供接口)		
AdminMassManager.addStudent	语法	public boolean addStudent(File studentData);
	前置条件	AdminMassManager 中已导入学生信息
	后置条件	批量添加学生
AdminMassManager.addTeacher	语法	public boolean addTeacher(File teacherData)
	前置条件	AdminMassManager 中已导入教师信息
	后置条件	批量添加教师
AdminMassManager.deleteStudent	语法	public boolean deleteStudent();
	前置条件	
	后置条件	批量删除学生
AdminMassManager.deleteTeacher	语法	public boolean deleteTeacher()
	前置条件	
	后置条件	批量删除教师
AdminMassManager.setStudentList	语法	public boolean setStudentList(ArrayList<StudentPO> studentList)
	前置条件	
	后置条件	导入学生数据
AdminMassManager.setTeacherList	语法	public boolean setTeacherList(ArrayList<TeacherPO> teacherList)
	前置条件	
	后置条件	导入教师数据
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

StudentDataService.insert(PO student)	在学生表中插入一学生条目
TeacherDataService.insert(PO teacher)	在教师表中插入一教师条目
TeacherDataService.delete(int id)	在指定表中删除指定编号的教师
StudentDataService.delete(int id)	在指定表中删除指定编号的学生

(3) 业务逻辑层的动态模型

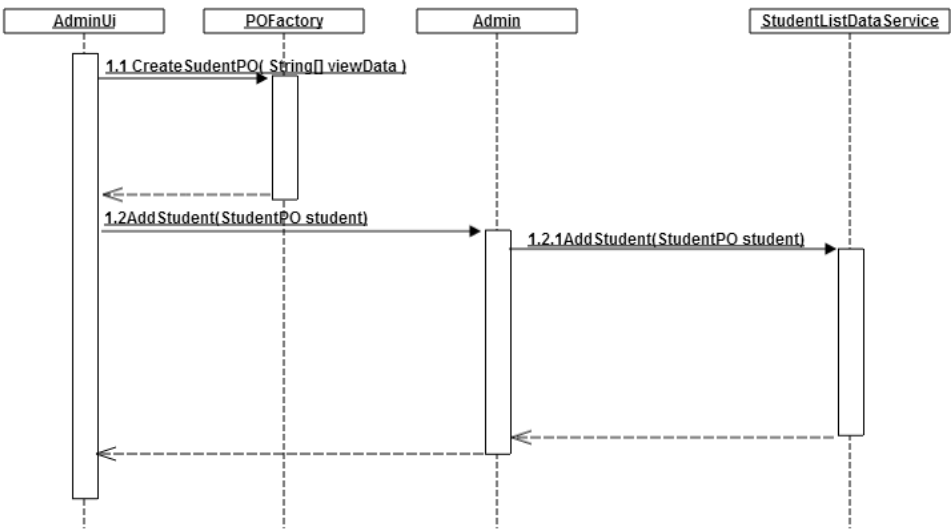


图 5 ： Admin 注册单个学生的顺序图

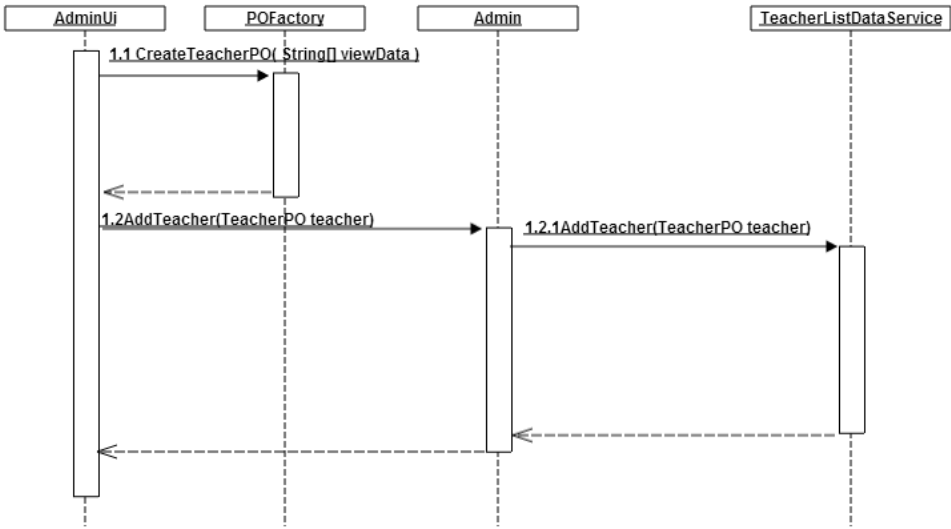


图 6: Admin 注册单个教师的顺序图

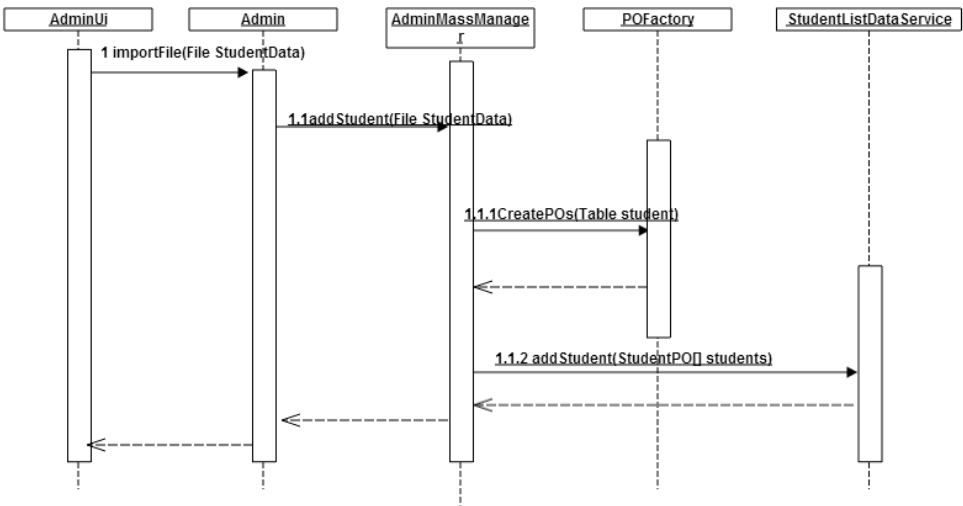


图 7: 管理员批量导入学生的顺序图

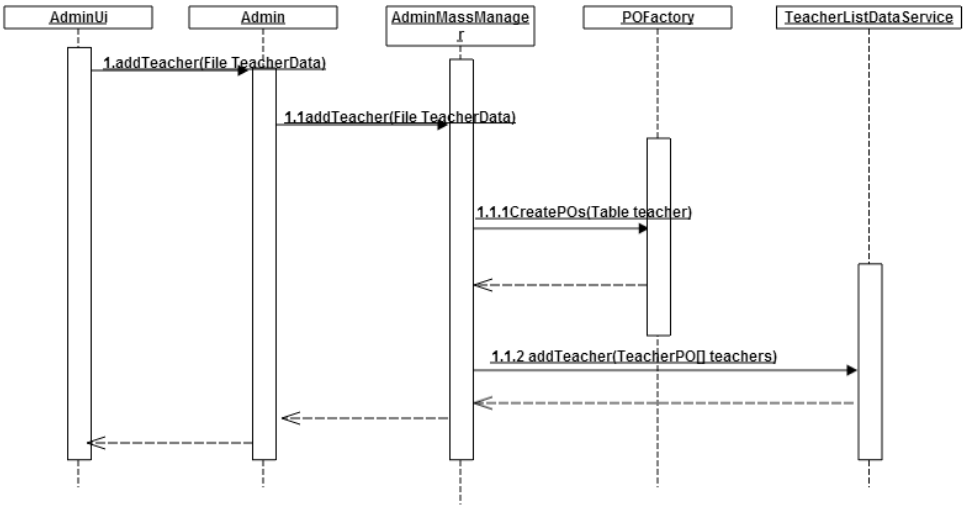


图 8: 管理员批量导入教师的顺序图

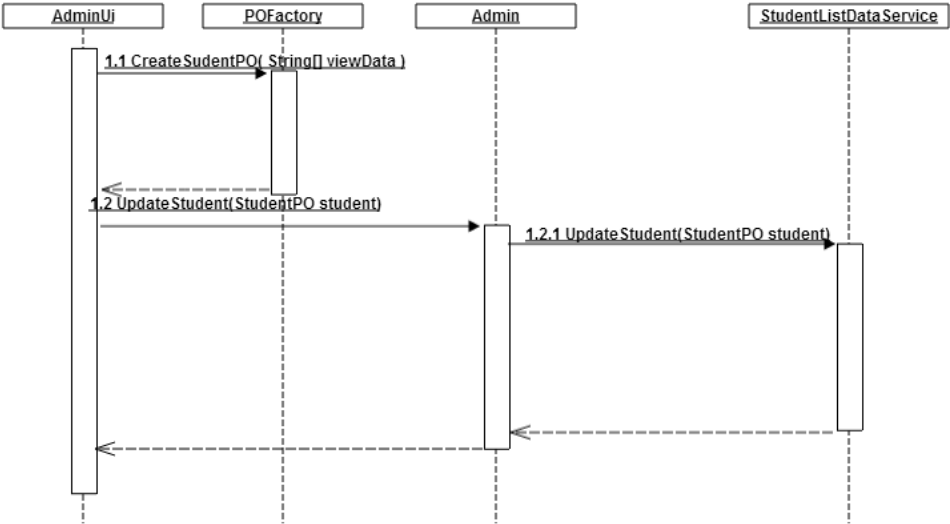


图 9 管理员修改学生信息的顺序图

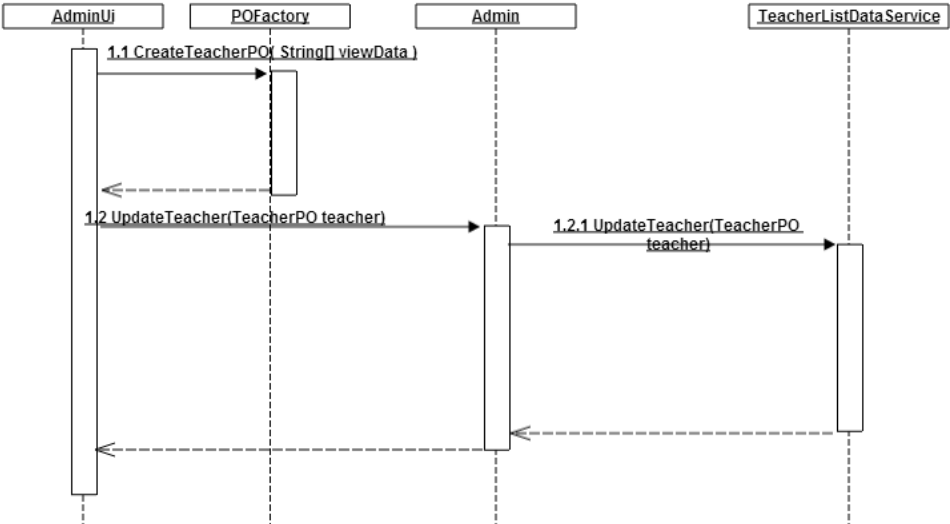


图 10: 管理员修改教师信息的顺序图

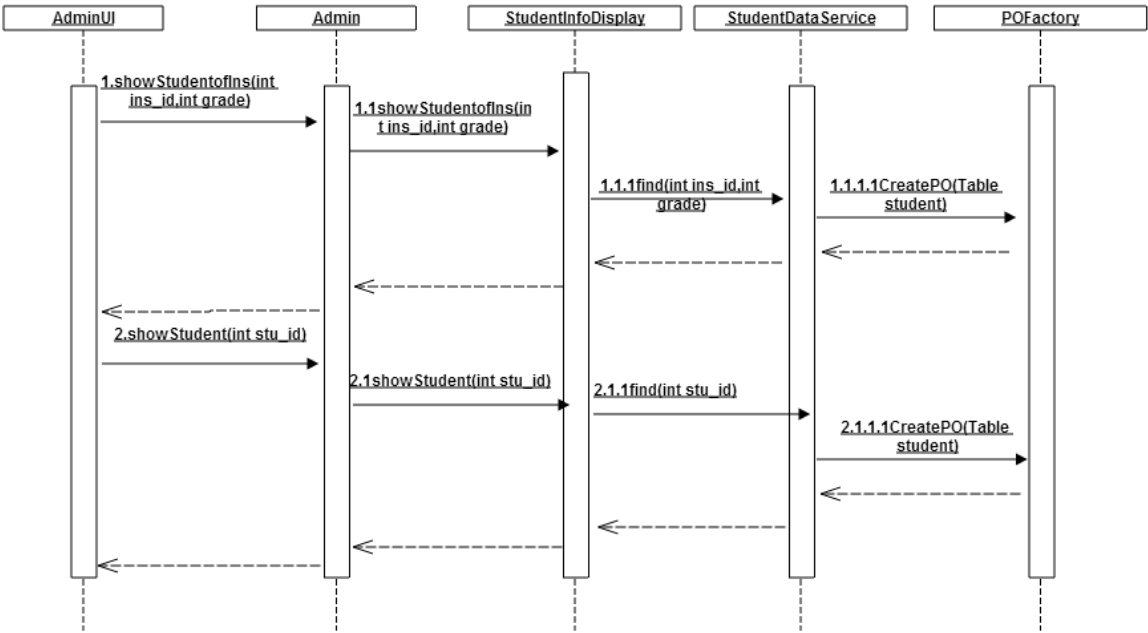


图 11：管理员查看学生列表的顺序图

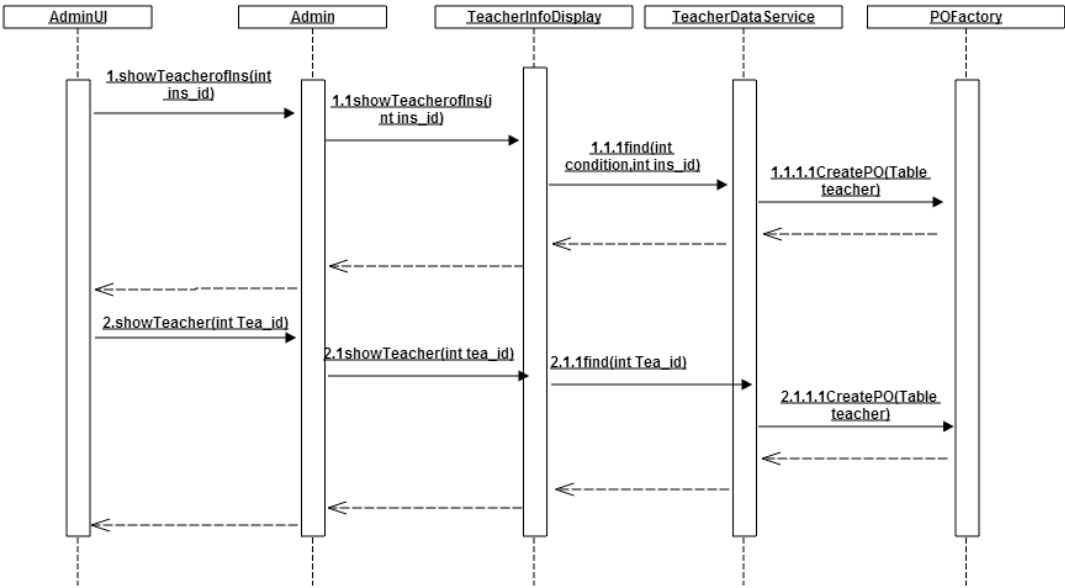


图 12：管理员查看老师顺序图

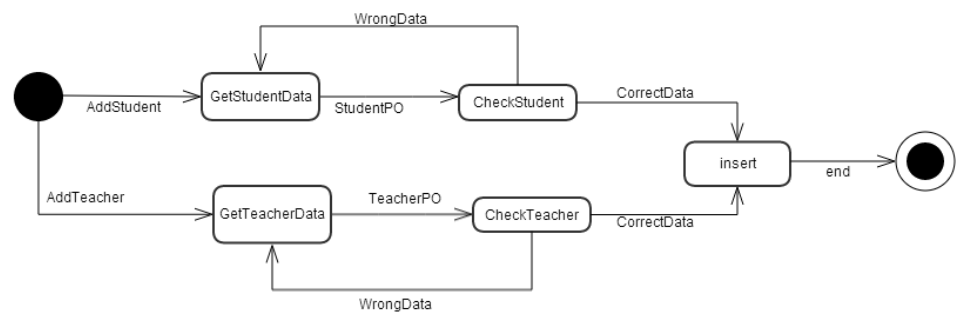


图 13： 管理员添加单个用户的对象状态图

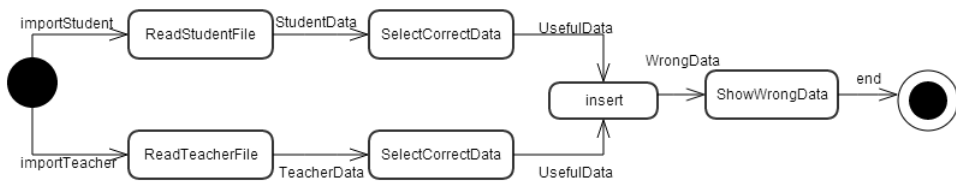


图 14： 管理员批量导入用户的对象状态图

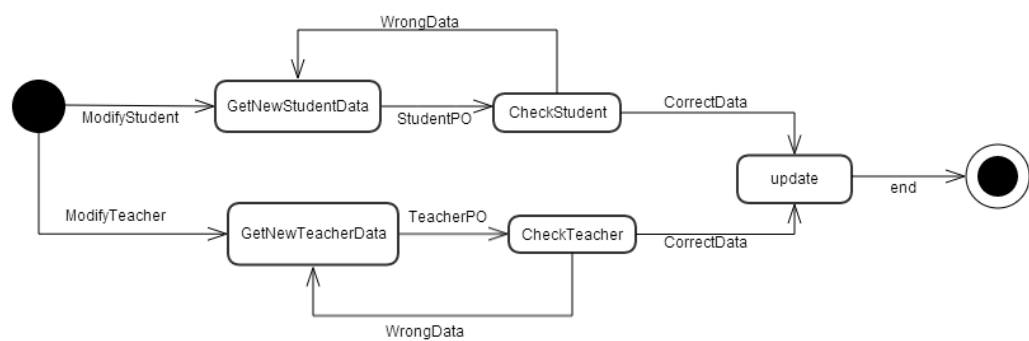


图 15： 管理员修改用户信息的对象状态图

另：查看信息的状态图略去。

4.2.3 studentbl 模块

表 8： studentbl 业务逻辑层详细设计的上下文

输入	需求	参见需求规格说明文档 V1.1 3.2.16-3.2.19 需求用例文档用例 16-用例 19
	体系结构	//被 presentation 层调用的接口 public interface StudentBIService { public boolean select(int Les_id);//选课 public boolean cancel(int Les_id);//退选 public boolean by_select(int Les_id);//补选

		<pre> public boolean by_cancel(int Les_id);//退课 public ArrayList<Lesson_uniquePO> show_myLesson();//查看我的课表 public ArrayList<Lesson_uniquePO> show_mySelection();//查看我的选课 public boolean changePassword(char[] old, char[] newPassword);//修改密码 public ArrayList<Les_RecordPO> show_myScore();//查看我的成绩 public void setSelectHelper(int lessonType) ;//设置选课类型 public ArrayList<Lesson_uniquePO> showBySelection();//显示可补选课程 } //调用 userbl 的接口 public interface UserControllerService { public int login(int id, char[] password); public boolean changePassword(char[] old, char[] newPassword); public PO getInformation(); } //调用 displayblservice 的接口 public interface LessonRecordDisplayService { //获取该学生的课程记录 public ArrayList<Les_RecordPO> getLessonRecord(int stu_id); public ArrayList<StudentPO> getStudentOfLesson(int les_id); //获取该学生的课程详细信息 public ArrayList <Lesson_uniquePO> getLessonOfStudent(int stu_id); } public interface SelectRecordDisplayService { //根据学号以及类别获取该学生该类别的所有选课 public ArrayList<Lesson_uniquePO> getChooseList(int stu_id, int type); } //调用 dataservice 的接口，方法均定义在父类 DatabaseSService 中 //选课记录数据服务，标红为此业务逻辑会用到的方法接口。 public interface SelectRecordDataService extends DatabaseService{ public boolean insert(PO po) throws RemoteException; public boolean delete(int id) throws RemoteException; public boolean update(PO po) throws RemoteException; public PO find(int id) throws RemoteException; public ArrayList<PO> find(int condition, int id) throws RemoteException; public ArrayList<PO> findAll() throws RemoteException; } //课程记录数据服务 public interface LessonRecordDataService extends DatabaseService{ public boolean insert(PO po) throws RemoteException; public boolean delete(int id) throws RemoteException; public boolean update(PO po) throws RemoteException; public PO find(int id) throws RemoteException; public ArrayList<PO> find(int condition, int id) throws RemoteException; public ArrayList<PO> findAll() throws RemoteException; </pre>
--	--	---

		}
输出	类图	图 16

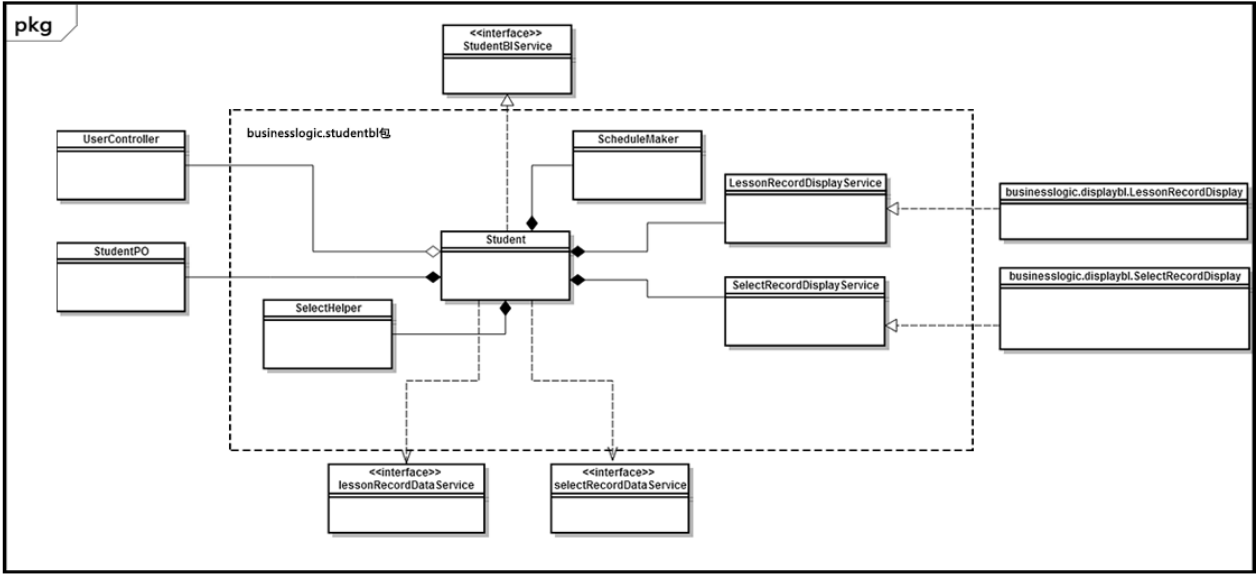


图 16: studentbl 模块类图

表 9: studentbl 模块各个类的职责

类	职责
Student	负责学生的各种功能实现，包括选课，退选、查看课表、查看成绩
SelectHelper	辅助学生进行各种课程的选课
SchduleMaker	辅助生成课表

(2) 模块内部类的接口规范

表 10 Student 的接口规范

提供的服务(供接口)		
Student.select	语法	public boolean select(int les_id);
	前置条件	学生登录、选课开放
	后置条件	系统生成选课记录，加入数据库，更新数据
Student.cancel	语法	public boolean cancel(int les_id);
	前置条件	学生登录、该学生选择了该编号的课程
	后置条件	系统从数据库删除该选课记录，更新数据
Student.by_select	语法	public boolean by_select(int les_id);
	前置条件	学生登录、补选开放、指定编号课程允许补选
	后置条件	系统直接生成一条课程记录（包括课程编号、学生学号以及成绩），加入数据库
Student.by_cancel	语法	public boolean by_cancel(int les_id);
	前置条件	学生登录、学生选到了该编号的课程

	后置条件	系统删除改课程记录，更新数据库
Student.showMyLesson	语法	public LessonPO[] showMyLesson()
	前置条件	无
	后置条件	系统返回该学生的所有课程
Student.showMyScore	语法	public LessonRecordPO[] showMyScore()
	前置条件	无
	后置条件	系统返回该学生的课程以及对应的分数
Student.showMySelection	语法	public LessonPO[] showMySelection(int type)
	前置条件	选课已开放
	后置条件	根据类别 系统返回该学生的选课
Student.showSelection	语法	public LessonPO[] showSelection(int lesson_type)
	前置条件	选课已开放
	后置条件	type 为课程类别，系统返回符合该条件的可选课程
Student.showBySelection	语法	public LessonPO[] showBySelection(int type,int Ins_id)
	前置条件	补选已开放
	后置条件	type 为课程类别,Ins_id 为院系编号，系统返回符合该条件的可补选课程
Student.changePassword	语法	public boolean changePassword(char[] old, char[] newpassword)
	前置条件	学生已登录，旧密码正确且新密码合法
	后置条件	系统更新学生信息，提示修改密码成功
Student.setSelectHelper	语法	public void setSelectHelper(int lessonType)
	前置条件	学生已登录，选课开放
	后置条件	系统选择选课类别
想要的服务（需接口）		
服务名		服务
UserControllerService.changePassword		用户修改密码
SelectRecordDisplayService.getChooseList		得到该学生制定课程类别的选课列表
SelectRecordDisplayService.getChooseLesson		根据课程类别与院系编号得到指定可选课程列表
LessonRecordDisplayService.getByChooseLesson		根据课程类别与院系编号得到指定可补选课程列表
LessonRecordDisplay.getLessonofStudent		得到该学生的所有课程信息
ScheduleMaker.makeSchdule		生成课程表
SelectionHelper.setMySelection		读入用户选择的课程
SelectionHelper.submit		提交选课
SelectionHelper.deleteSelection		删除选课
DatabaseFactory.getLessonRecordDatabase		得到 LessonRecord 数据库的服务的引用
DatabaseFactory.getChooseDatabase		得到 Choose 数据库服务的的引用
LessonRecordDataService.find(int les_id,int stu_id)		从课程记录查找符合学生 id 与课程编号对应的记录。
ChooseDataService.find(int les_id,int stu_id)		从选课记录查找符合学生 id 与课程编号对应的记录。

LessonRecordDataService.insert(PO lessonrecord)	从课程记录中插入一条新纪录
ChooseDataService.insert(PO Choose)	从选课记录中插入一条新纪录
LessonRecordDataService.delete(int les_id,int stu_id)	在课程记录中删除指定编号的学生和课程对应的记录
LessonRecordDataService.delete(int les_id,int stu_id)	在选课记录中删除指定编号的学生和课程对应的记录

表 11 SelectHelper 的接口规范

提供的服务(供接口)		
SelectHelper.setMySelection	语法	public void setMySelection(ArrayList<Lesson_uniquePO> mySelection)
	前置条件	选课开放
	后置条件	导入已选课程
SelectHelper.submit	语法	public void submit()
	前置条件	选课开放
	后置条件	提交选课列表
SelectHelper.deleteSelection	语法	public void deleteSelection(Lesson_uniquePO selection)
	前置条件	选课开放
	后置条件	退选课程
想要的服务 (需接口)		
服务名		服务
ChooseDataService.insert(PO Choose)		从选课记录中插入一条新纪录
LessonRecordDataService.delete(int les_id,int stu_id)		在选课记录中删除指定编号的学生和课程对应的记录

表 12 ScheduleMaker 的接口规范

提供的服务(供接口)		
ScheduleMaker.makeSchedule	语法	public ArrayList<Lesson_uniquePO> makeSchedule();
	前置条件	
	后置条件	对已有课程进行排序操作 生成课表
想要的服务 (需接口)		
服务名		服务
无		无

(3) 业务逻辑层的动态模型

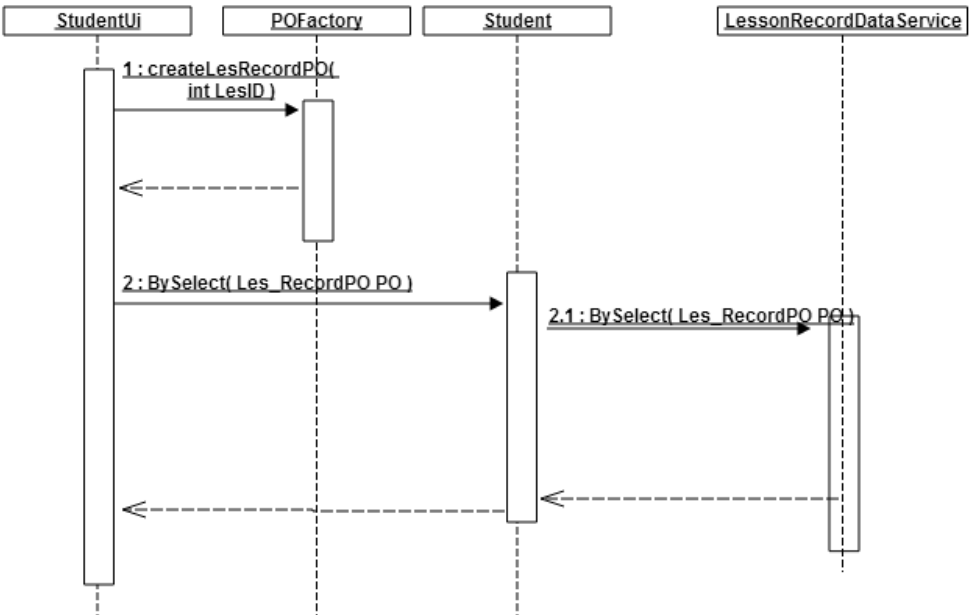


图 17： 学生补选的顺序图

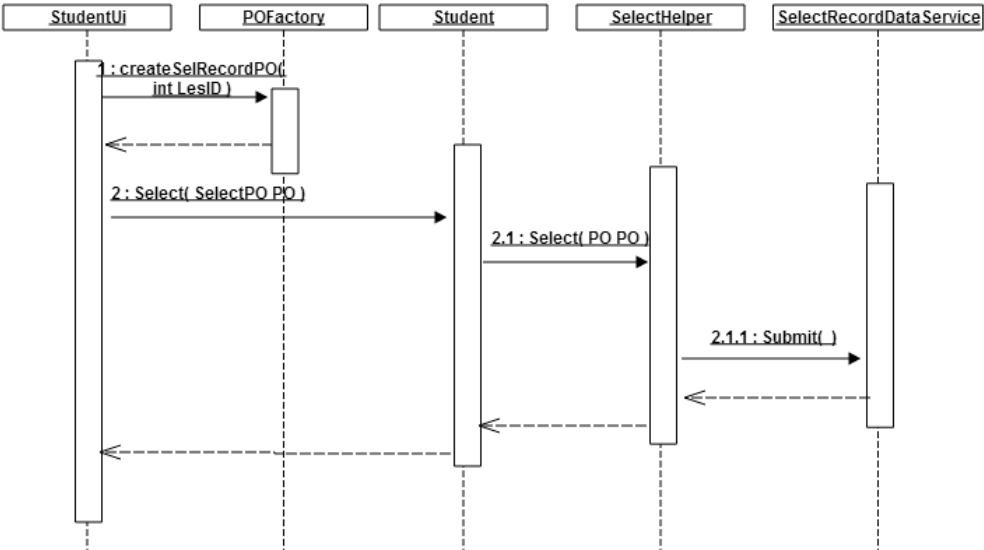


图 18： 学生选课的顺序图

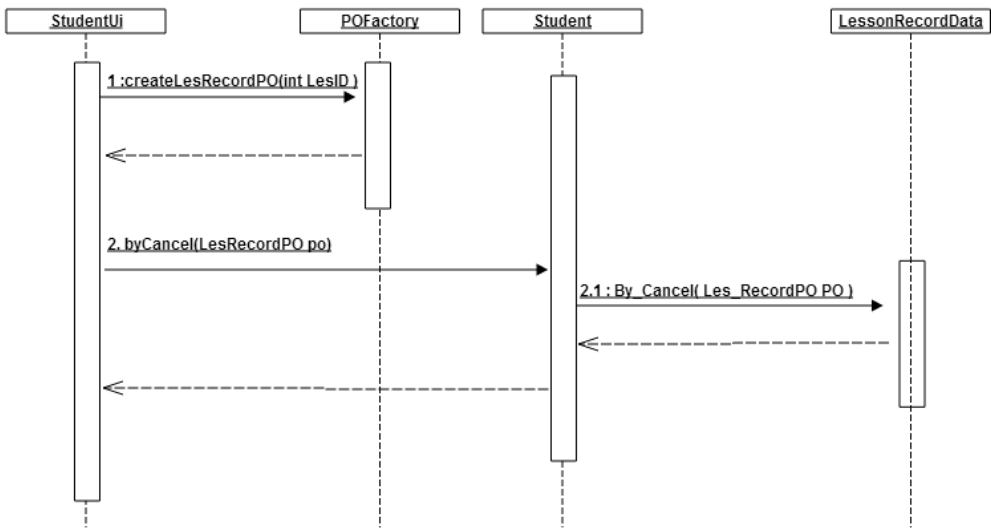


图 19: 学生退课的顺序图

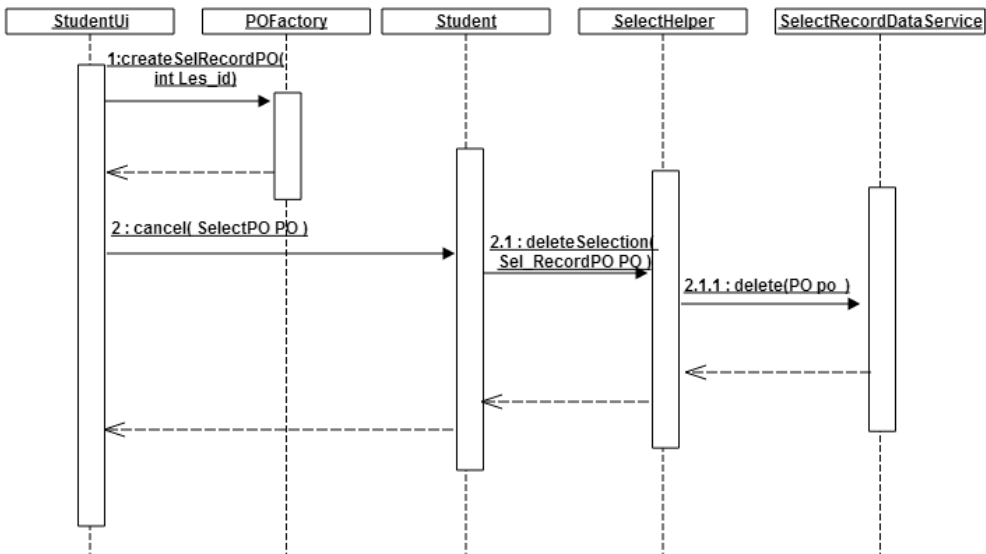


图 20:学生退选的顺序图

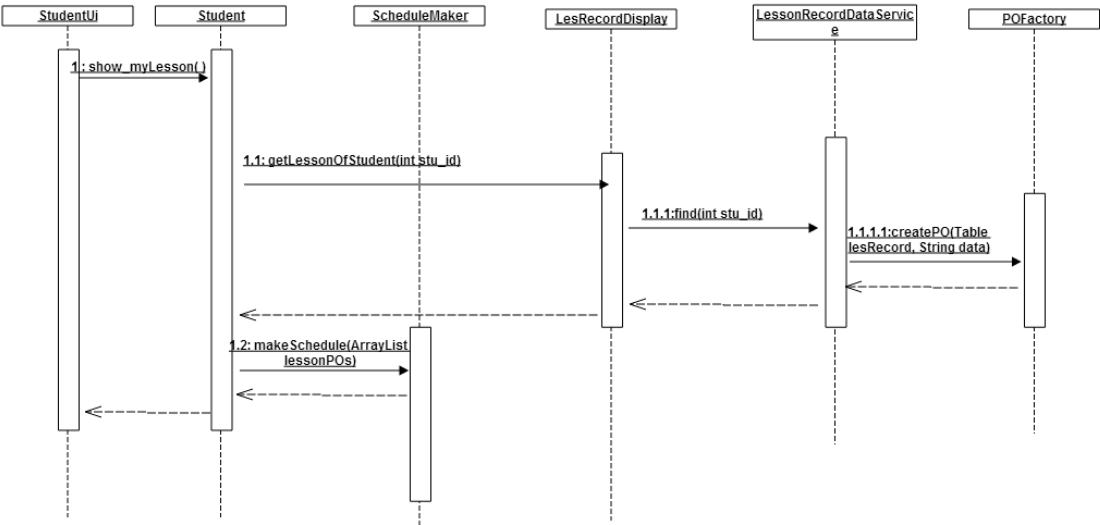


图 21： 学生查看课表的顺序图

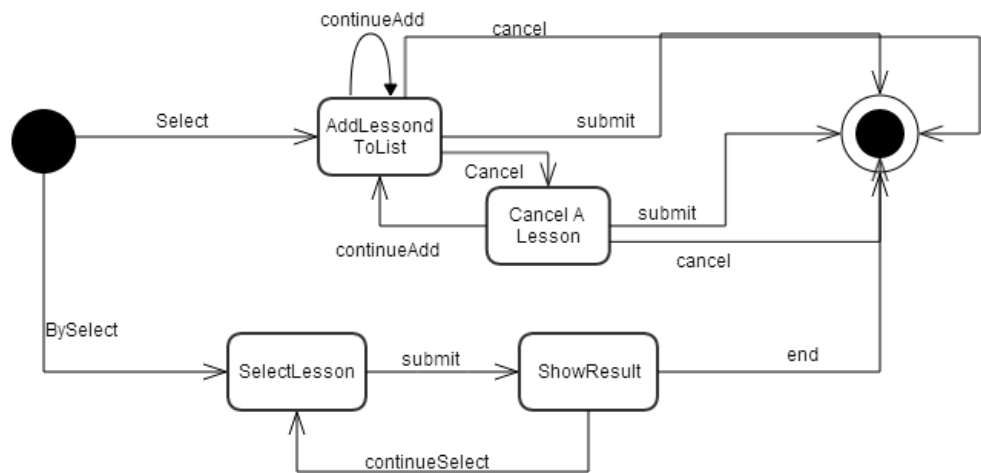


图 22 ： 学生选课补选的对象状态图

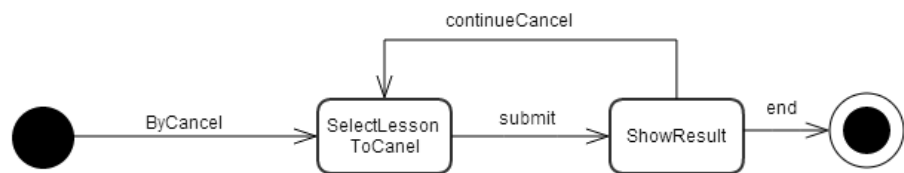


图 23: 学生退课的对象状态图

4.2.4 teacherbl 模块

表 13: teacherbl 业务逻辑层详细设计的上下文

输入	需求	参见需求规格说明文档 V1.1 3.2.12-3.2.14 3.3.3 (易用性) 需求用例文档用例 12-用例 14
	体系结构	<p>//被 presentation 层调用的接口</p> <pre> public interface TeacherBIService { public boolean recordScore();//批量导入成绩 public boolean editLesInfo(Lesson_uniquePO lesson);//编辑课程信息 public boolean recordScore(Les_RecordPO record);//单独录入成绩 public ArrayList<Lesson_uniquePO> showMyLesson();//查看我的课程 public ArrayList<StudentPO> showMyStudent(int les_id);//查看课程的学生 public boolean changePassword(char[] old, char[] newPassword);//修改密码 public void chooseLesson(int les_id) public void addScore(int stu_id,int score) } </pre> <p>//调用 userbl 的接口</p> <pre> public interface UserControllerService { public int login(int id, char[] password); public boolean changePassword(char[] old, char[] newPassword); public PO getInformation(); } </pre> <p>//调用 displayblservice 的接口</p> <pre> public interface LessonRecordDisplayService { //获取该学生的课程记录 public ArrayList<Les_RecordPO> getLessonRecord(int stu_id); public ArrayList<StudentPO> getStudentOfLesson(int les_id); //获取该学生的课程详细信息 public ArrayList <Lesson_uniquePO> getLessonOfStudent(int stu_id); } </pre> <pre> public interface LessonDisplayService { public ArrayList<Lesson_uniquePO> getByChooseLesson(int type, int ins_id); public ArrayList<Lesson_uniquePO> getLessonsOfIns(int ins_id); //获取教师课程 public ArrayList<Lesson_uniquePO> getLessonOfTeacher(int tea_id); //获取课程详细信息 public Lesson_uniquePO getLessonInfo(int Les_id); } </pre> <p>//调用 dataservice 的接口，方法均定义在父类 DatabaseSService 中</p>

	<pre>//选课记录数据服务，标红为此业务逻辑会用到的方法接口。 public interface LessonUnRecordDataService extends DatabaseService{ public boolean insert(PO po) throws RemoteException; public boolean delete(int id) throws RemoteException; public boolean update(PO po) throws RemoteException; public PO find(int id) throws RemoteException; public ArrayList<PO> find(int condition, int id) throws RemoteException; public ArrayList<PO> findAll() throws RemoteException; } //课程记录数据服务 public interface LessonRecordDataService extends DatabaseService{ public boolean insert(PO po) throws RemoteException; public boolean delete(int id) throws RemoteException; public boolean update(PO po) throws RemoteException; public PO find(int id) throws RemoteException; public ArrayList<PO> find(int condition, int id) throws RemoteException; public ArrayList<PO> findAll() throws RemoteException; }</pre>
输出	类图 <div>图 24</div>

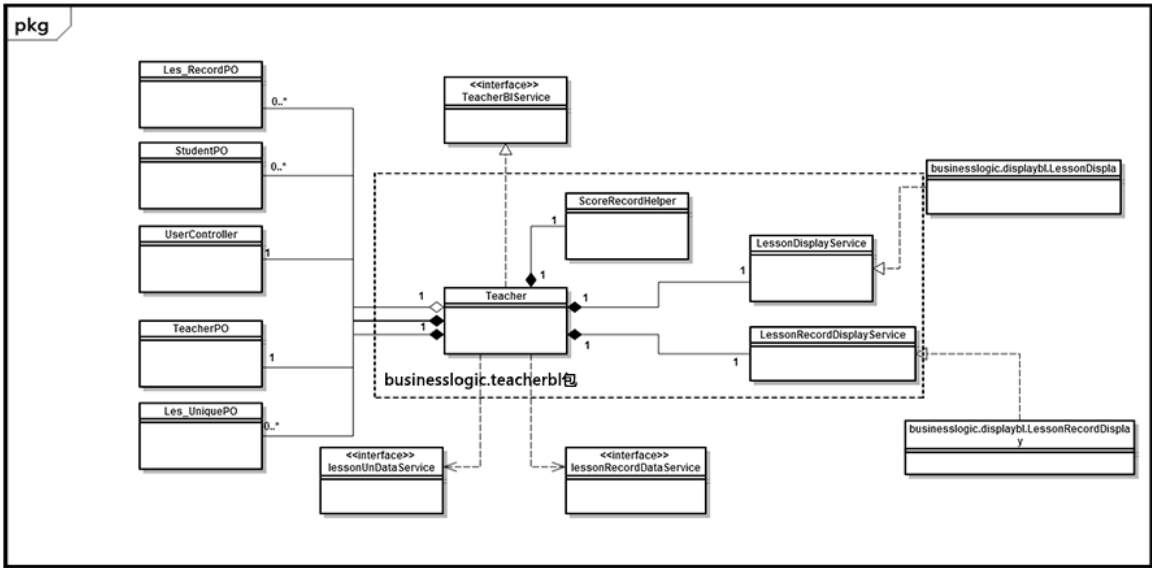


图 24:teacherbl 模块类图

表 14: teacherbl 模块各个类的职责

模块	职责
Teacher	负责教师的各种功能实现，包括编辑课程信息、录入成绩、查看我的学生、查看我的课程
ScoreRecordHelper	负责协助教师进行批量地成绩录入

(2) 模块内部类的接口规范

表 15 teacher 模块的接口规范

提供的服务(供接口)		
Teacher.editLesInfo	语法	public boolean editLesInfo(LessonPO lesson);
	前置条件	教师登陆, 课程已发布
	后置条件	系统更新课程信息
Teacher.recordScore	语法	public boolean recordScore(LessonRecordPO record); public boolean recordScore()
	前置条件	任课老师登陆、存在原课程记录
	后置条件	系统更新该课程记录, 添加学生成绩 参数为空代表调用 ScoreRecordHelper 进行批量导入
Teacher.showMyLesson	语法	public LessonPO[] showMyLesson()
	前置条件	任课老师登陆, 任课老师有课程
	后置条件	系统返回该任课老师的所有课程
Teacher.showMyStudent	语法	public StudentPO[] showMyStudent(int les_id)
	前置条件	任课老师登陆, 任课老师有该课程
	后置条件	系统返回该课程的学生名单
Teacher.changePassword	语法	public boolean changePassword(char[] old, char[] newpassword)
	前置条件	任课老师已登录, 旧密码正确且新密码合法
	后置条件	系统更新任课老师信息, 提示修改密码成功
Teacher.chooseLesson	语法	public void chooseLesson(int les_id)
	前置条件	任课老师已登录, 有该课程编号的课程
	后置条件	系统做好登记成绩的准备
想要的服务 (需接口)		
服务名		服务
UserControllerService.changePassword		用户修改密码
ScoreRecordHelper.recordScore		批量导入成绩
LessonRecordDisplayService.getStudentofLesson		得到该课程的所有学生
LessonDisplayService.getLessonListofTea		得到该教师的所有课程
DatabaseFactory.getLessonRecordDatabase		得到 LessonRecord 数据库的服务的引用
DatabaseFactory.getLessonDatabase		得到 Lesson 数据库服务的的引用
LessonRecordDataService.update(PO lessonrecord)		更新课程记录中一条数据
LessonDataService.update(PO Lesson)		更新具体课程表中一条数据

表 16 ScoreRecordHelper 的接口规范

提供的服务(供接口)		
ScoreRecordHelper.recordScore	语法	public void recordScore()
	前置条件	教师选择导入文件
	后置条件	导入成绩信息
想要的服务 (需接口)		
服务名		服务
LesRecordDataService.upda		更新课程记录

te(PO lesRecord)	
------------------	--

(3) 业务逻辑层的动态模型

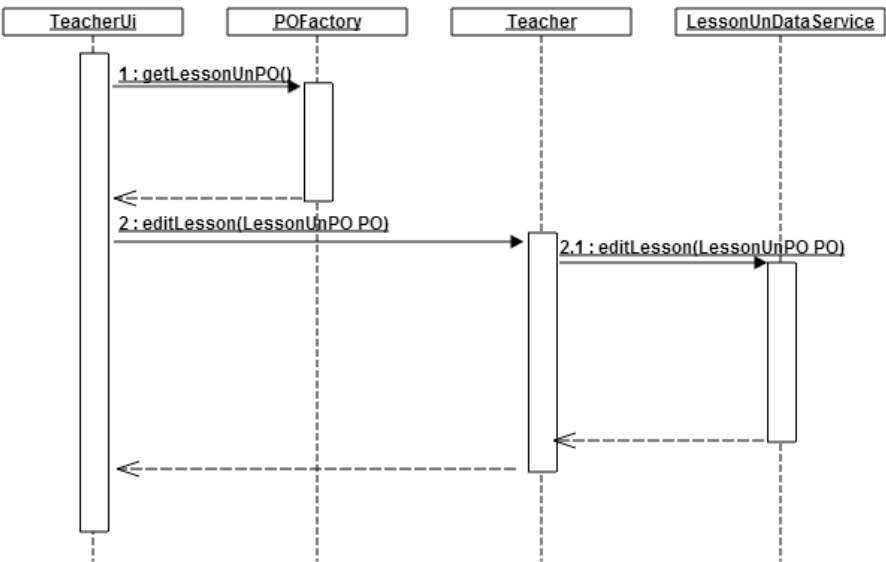


图 25:教师编辑课程信息的顺序图

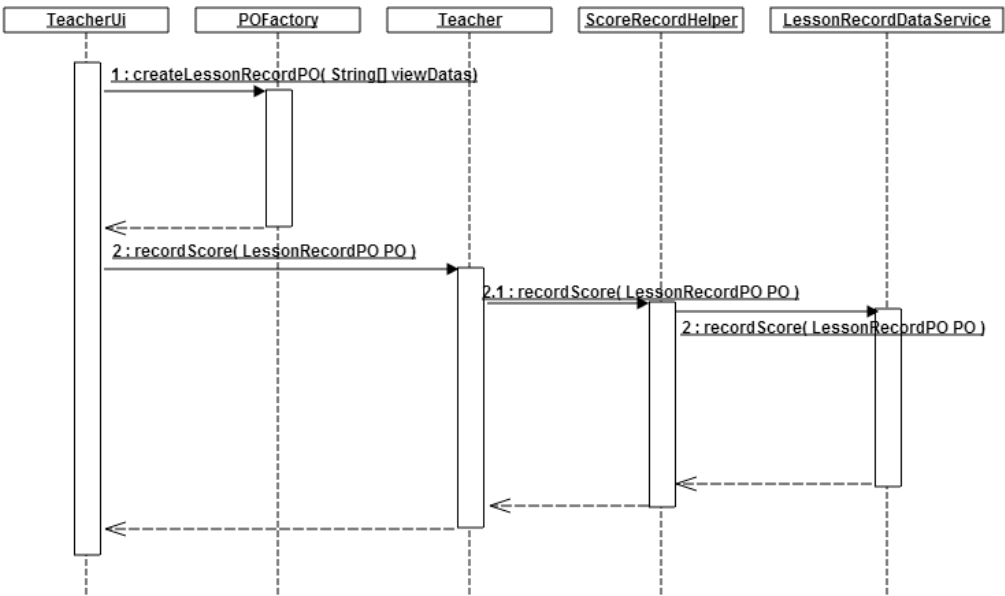


图 26: 教师登记分数的顺序图

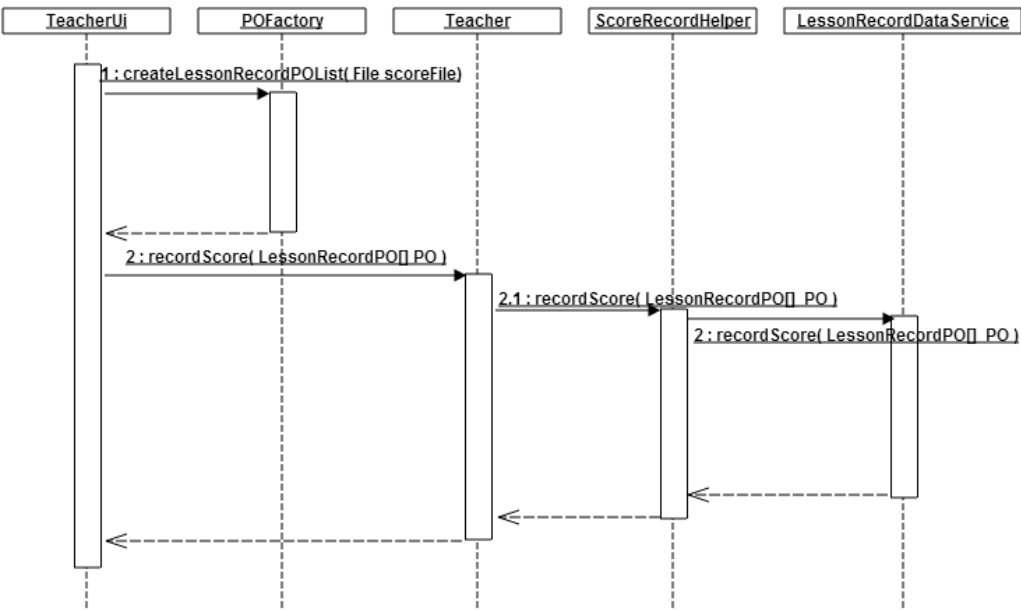


图 27:教师批量导入学生成绩的顺序图

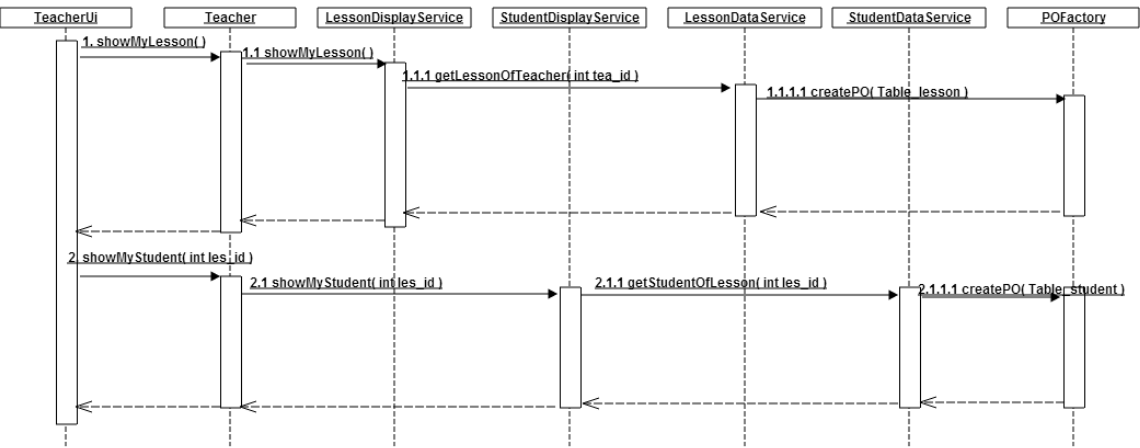


图 28:教师查看课程以及课程学生的顺序图

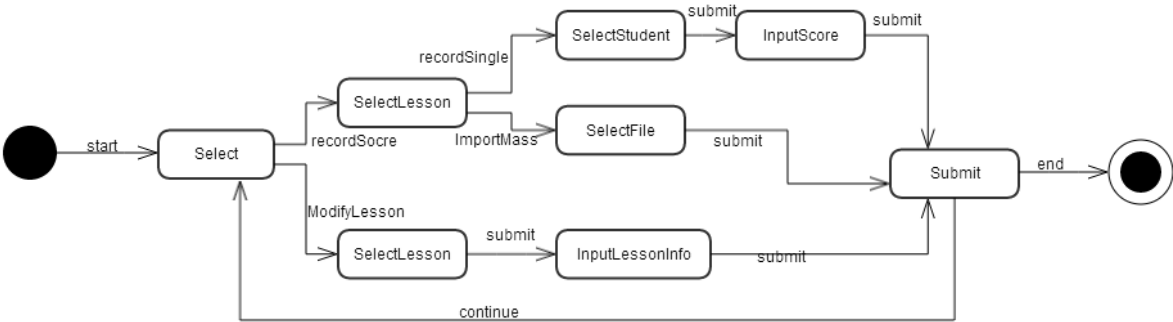


图 29:任课老师登录成绩的对象状态图

4.2.5 insteacherbl 模块

表 17: insteacherbl 业务逻辑层详细设计的上下文

	需求	参见需求规格说明文档 V1.1 3.2.8-3.2.11 3.3.3（易用性） 需求用例文档用例 8-用例 11
输入	体系结构	<div>//被 presentation 层调用的接口</div> <div>public interface InsTeacherBIService { public boolean addLesson(Lesson_abstractPO lesson_abstractPO);//计划——>添加课程 public boolean publishLesson(Lesson_abstractPO lesson_abstractPO, Lesson_uniquePO newLesson);//发布计划中的课程 public boolean modifyPlan(Lesson_abstractPO lesson_abstractPO);//修改教学计划（即计划中的课程）; public boolean modifyLesson(Lesson_uniquePO lesson_uniquePO);//修改已发布课程 public boolean addStudent(Les_RecordPO les_RecordPO); //在某门课中添加学生 public boolean addStudent(int ins_id,int grade);//在她们课中添加某院系整个年级的学生 public boolean addStudent();//批量外部导入 public boolean deleteStudent(Les_RecordPO les_RecordPO); //在某门课中删除学生 public ArrayList<Lesson_uniquePO> showLesson();//显示该院系的所有课程 public ArrayList<StudentPO> showStudentofLesson(int les_id); public boolean changePassword(char[] old, char[] newPassword);//修改密码 public PlanVO showPlan(); } //调用 userbl 的接口</div> <div>public interface UserControllerService { public int login(int id, char[] password); public boolean changePassword(char[] old, char[] newPassword); public PO getInformation(); }</div> <div>//调用 displayblservice 的接口</div> <div>public interface LessonRecordDisplayService { //获取该学生的课程记录 public ArrayList<Les_RecordPO> getLessonRecord(int stu_id); public ArrayList<StudentPO> getStudentOfLesson(int les_id); //获取该学生的课程详细信息 public ArrayList <Lesson_uniquePO> getLessonOfStudent(int stu_id); }</div>

输出	类图	<pre> public interface LessonDisplayService { public ArrayList<Lesson_uniquePO> getByChooseLesson(int type, int ins_id); public ArrayList<Lesson_uniquePO> getLessonsOfIns(int ins_id); //获取教师课程 public ArrayList<Lesson_uniquePO> getLessonOfTeacher(int tea_id); //获取课程详细信息 public Lesson_uniquePO getLessonInfo(int Les_id); } //调用 dataservice 的接口，方法均定义在父类 DatabaseSService 中 //选课记录数据服务，标红为此业务逻辑会用到的方法接口。 public interface LessonUnRecordDataService extends DatabaseService{ public boolean insert(PO po) throws RemoteException; public boolean delete(int id) throws RemoteException; public boolean update(PO po) throws RemoteException; public PO find(int id) throws RemoteException; public ArrayList<PO> find(int condition, int id) throws RemoteException; public ArrayList<PO> findAll() throws RemoteException; } //课程记录数据服务 public interface LessonRecordDataService extends DatabaseService{ public boolean insert(PO po) throws RemoteException; public boolean delete(int id) throws RemoteException; public boolean update(PO po) throws RemoteException; public PO find(int id) throws RemoteException; public ArrayList<PO> find(int condition, int id) throws RemoteException; public ArrayList<PO> findAll() throws RemoteException; } </pre>
图 30		

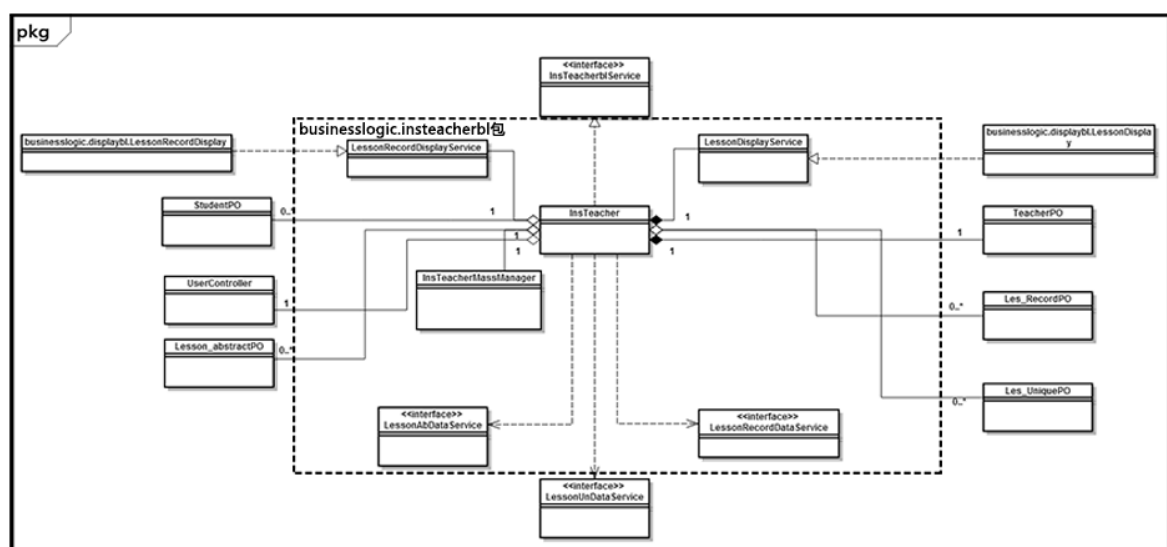


图 30:insteacherbl 模块类图

表 18: insteacherbl 模块各个类的职责

模块	职责
InsTeacher	负责院系教务老师各种功能的实现，包括编辑教学计划、发布课程、修改已发布的课程、查看课程学生、编辑课程学生。
InsTeacherMassManager	负责协助院系教务老师进行课程学生的批量处理
PlanEditor	负责协助院系教务老师进行计划编辑

(2) 模块内部类的接口规范

表 19 Insteacher 类的接口规范

提供的服务(供接口)		
InsTeacher.addLesson	语法	public boolean addLesson(Lesson_abstractPO Lesson);
	前置条件	院系教务老师登陆，教学框架已制定
	后置条件	系统添加一条抽象课程，根据所述课程类别更新该院系教学计划
InsTeacher.publishLesson	语法	public boolean publish(Lesson_abstractPO lesson, Lesson_uniquePO newLesson);
	前置条件	院系教务老师登陆，教学计划内含该课程
	后置条件	系统添加一条具体课程，该课程列入到已发布课程列表中
InsTeacher.modifyPlan	语法	public boolean modifyLesson(Lesson_abstractPO Lesson);
	前置条件	院系教务老师登陆，教学计划内含该抽象课程
	后置条件	系统更新该抽象课程的信息
InsTeacher.modifyLesson	语法	public boolean modifyLesson(Lesson_uniquePO Lesson);
	前置条件	院系教务老师登陆，该课程已发布
	后置条件	系统更行该已发布课程的信息
InsTeacher.addStudent	语法	public boolean addStudent(LessonRecordPO, lessonRecord)
	前置条件	院系教务老师登陆，该课程已发布，该课程不包含此学生
	后置条件	系统添加该课程记录，包含该学生学号以及该课程编号。
InsTeacher.deleteStudent	语法	public boolean deleteStudent(LessonRecordPO, lessonRecord)
	前置条件	院系教务已登录，该课程内有该学生名单
	后置条件	系统删除对应的课程记录
InsTeacher.showLessons	语法	public Lesson_uniquePO[] ShowLessons()
	前置条件	院系教务老师已登录

	后置条件	返回该院系的所有课程
InsTeacher.showStudentofLesson	语法	public StudentPO[] ShowStudentofLesson(int les_id)
	前置条件	院系教务老师已登录
	后置条件	返回该指定课程编号课程的学生
InsTeacher.changePassword	语法	public boolean changePassword(char[] old, char[] newpassword)
	前置条件	院系教务老师已登录, 旧密码正确且新密码合法
	后置条件	系统更新院系教务老师信息, 提示修改密码成功
InsTeacher.showPlan	语法	public PlanVO ShowPlan()
	前置条件	院系教务老师已登录
	后置条件	返回该院系的教学计划
想要的服务 (需接口)		
服务名		服务
UserblService.changePassword		用户修改密码
LessonDisplayblService.getLessonListofIns		系统返回指定院系已发布的所有课程
LessonRecordDisplayblService.getStudentofLes		系统返回指定编号课程的学生
PlanDisplayblService.getPlan		系统返回该院系的教学计划
PlanEditor.addLesson		增加计划中的抽象课程
PlanEditor.modifyPlan		修改教学计划 (修改抽象课程)
PlanEditor.checkPlan		检验计划是否符合要求
PlanEditor.showPlan		显示已编辑好的计划
InsTeacherMassManager.addStudent		批量增加学生
DatabaseFactory.getLessonRecordDatabase		得到 LessonRecord 数据库的服务的引用
DatabaseFactory.getPlanDatabase		得到 Plan 数据库服务的的引用
DatabaseFactory.getLessonDatabase		得到 Lesson 数据库服务的引用
LessonRecordDataService.insert(PO lessonrecord)		从课程记录中插入一条新纪录
LessonRecordDataService.delete(int les_id,int stu_id)		在课程记录中删除指定编号的学生和课程对应的记录
LessonAbDataService.update(PO Plan)		在抽象课程表中更新对应课程的信息
LessonAbDataService.delete(int id)		在抽象课程表中删除对应抽象课程
LessonAbDataService.insert(PO Plan)		在抽象课程表中插入一条新的记录
LessonUnDataService.update(PO Lesson)		在具体课程表中更新对应课程的信息
LessonUnDataService.insert(PO Lesson)		在具体课程表中插入一条新的记录

表 20 InsteacherMassManager 的接口规范

提供的服务(供接口)		
InsteacherMassManager.addStudent	语法	public boolean addStudent(int les_id,int ins_id, int grade); public boolean addStudent(int les_id,int fileName);
	前置条件	课程已发布
	后置条件	3 个参数代表导入该院系该年级的所有学生 2 个参数代表根据外部文件导入
想要的服务 (需接口)		

服务名	服务
LesRecordDataService.add(PO lesRecord)	添加课程记录

表 21 PlanEditor 的接口规范

提供的服务(供接口)		
PlanEditor.addLesson	语法	public boolean addLesson(Lesson_abstractPO lesson_abstractPO);
	前置条件	教学框架已制定
	后置条件	增加抽象课程一条
PlanEditor.modifyPlan	语法	public boolean modifyPlan(Lesson_abstractPO lesson_abstractPO);
	前置条件	计划已制定
	后置条件	修改已制定计划中的指定一条抽象课程
PlanEditor.checkPlan	语法	public boolean checkPlan(ArrayList<Lesson_abstractPO> list);
	前置条件	计划已制定
	后置条件	检查计划是否符合框架学分要求
PlanEditor.showPlan	语法	public PlanVO showPlan(ArrayList<Lesson_abstractPO> list);
	前置条件	计划已制定
	后置条件	生成教学计划
想要的服务（需接口）		
服务名	服务	
LessonAbDataService.update(PO Plan)	在抽象课程表中更新对应课程的信息	
LessonAbDataService.delete(int id)	在抽象课程表中删除对应抽象课程	
LessonAbDataService.insert(PO Plan)	在抽象课程表中插入一条新的记录	

(3)业务逻辑层的动态模型

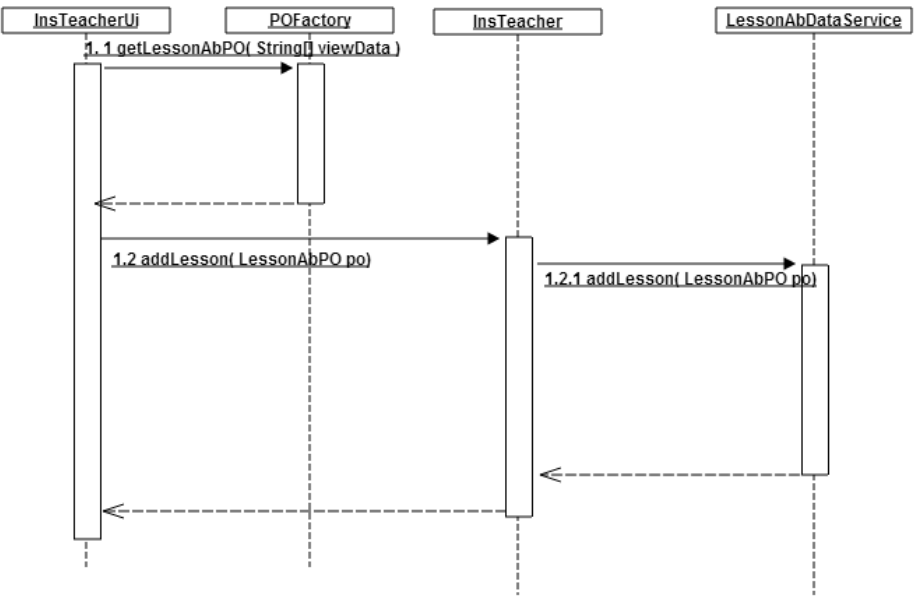


图 31： 院系教务老师添加课程顺序图

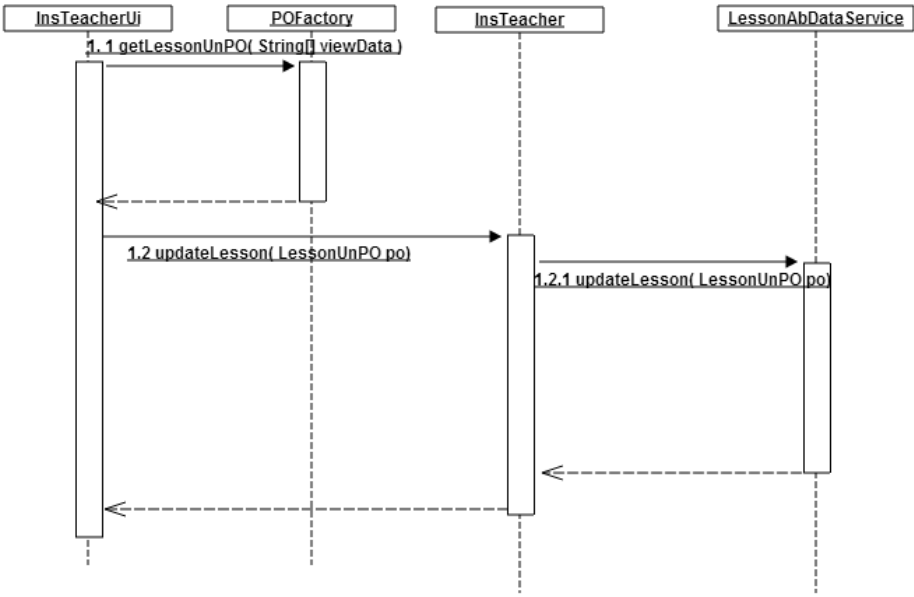


图 32： 院系教务老师修改已发布课程顺序图

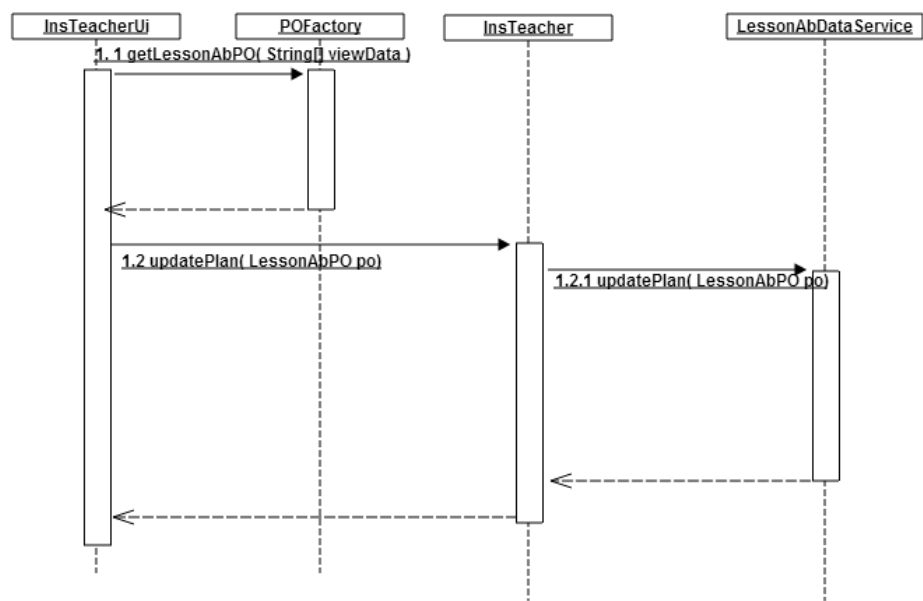


图 33：院系教务老师修改计划顺序图

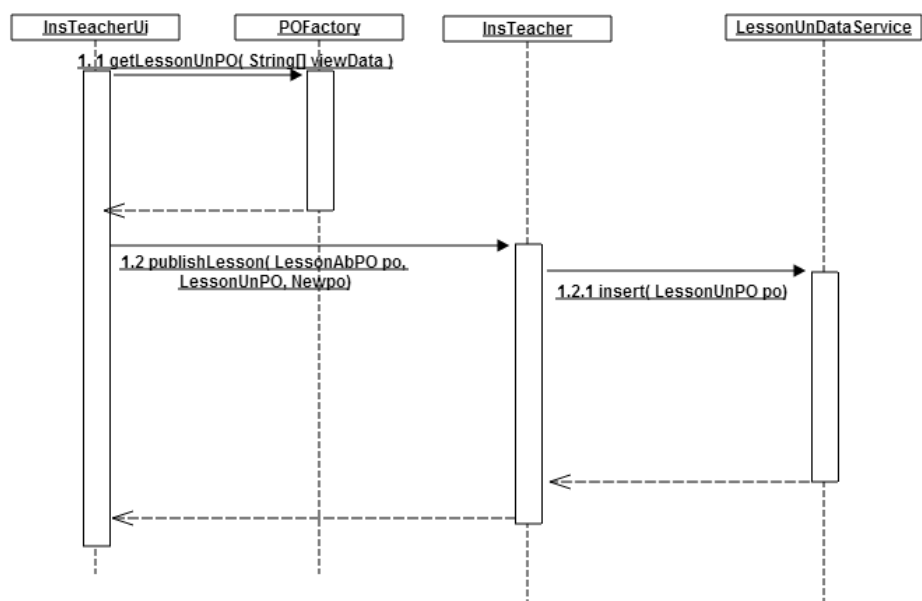


图 34：院系教务老师发布课程顺序图

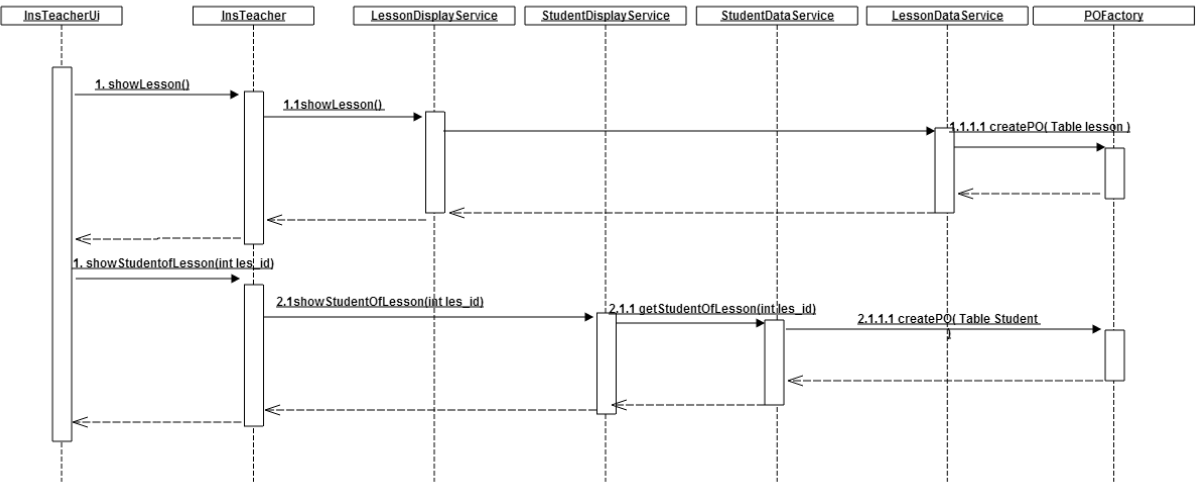


图 35:查看课程以及课程学生的顺序图

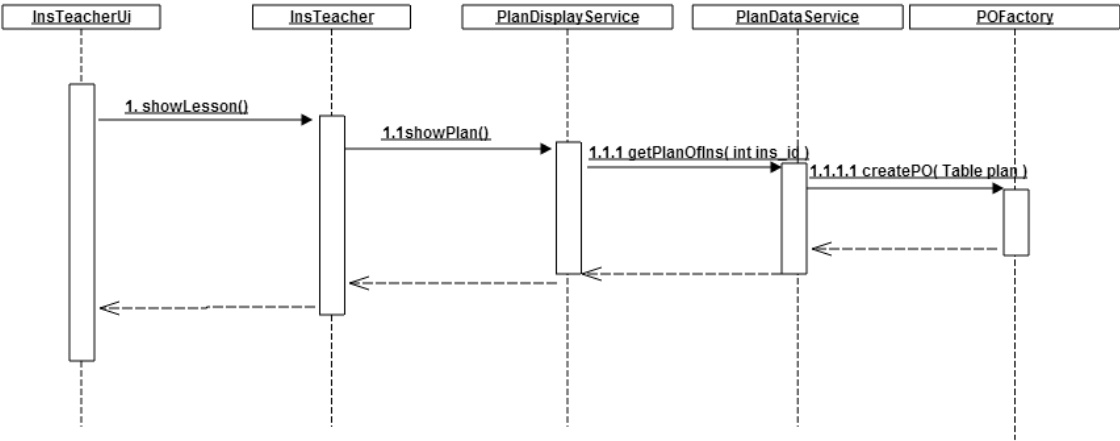


图 35:查看院系教学计划的顺序图

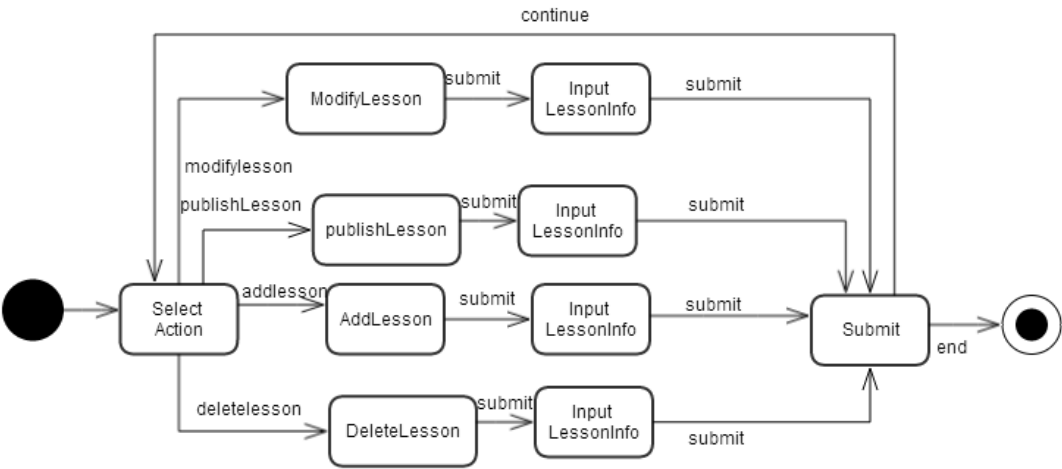


图 36: 院系教务老师编辑教学计划的对象状态图

4.2.6 schteacherbl 模块

表 22: insteacherbl 业务逻辑层详细设计的上下文

	需求	参见需求规格说明文档 V1.1 3.2.8-3.2.11 3.3.3 (易用性) 需求用例文档用例 8-用例 11
	输入 体系结构	<pre> //被 presentation 层调用的接口 public interface SchTeacherBIService { //查看指定院系教学计划 public PlanVO showPlan(int ins_id); //增加抽象课程 (教学计划中) public boolean addLesson(Lesson_abstractPO lesson); //增加课程类型 public boolean addType(TypePO typePO); //增加课程模块 public boolean addModule(ModulePO modulePO); //修改抽象课程 public boolean modifyLesson(Lesson_abstractPO lesson_abstractPO); //修改课程类型 public boolean modifyType(TypePO typePO); //修改课程模块 public boolean modifyModule(ModulePO modulePO); //删除类型 public boolean deleteType(int id); //删除模块 public boolean deleteModule(int id); //删除课程 public boolean deleteLesson(int id); //获取指定院系指定年级学生名单 public ArrayList<StudentPO> showStudentList(int ins_id, int grade); //获取指定院系老师名单 public ArrayList<TeacherPO> showTeacherList(int ins_id); //获取指定院系抽象课程 public ArrayList<Lesson_abstractPO> showLessonList(int ins_id); //修改密码 public boolean changePassword(char[] old, char[] newPassword); } //调用 displayblservice 的接口 public interface PlanDisplayService { public Lesson_abstractPO getPlan(int LesAb_id); public ArrayList<Lesson_abstractPO> getPlanofIns(int ins_id);} public interface StudentInfoDisplayService { //根据学号获取学生个人信息 </pre>

```

    public StudentPO getStudent(int stu_id);
    //根据院系号获取学生列表
    public ArrayList<StudentPO> getStudentList(int ins_id , int grade);
    //根据课程号获取该课程的所有学生列表
    public ArrayList<StudentPO> getStudentListByLesson(int les_id);
}

public interface TeacherInfoDisplayService {
    public TeacherPO getTeacher(int tea_id);
    public ArrayList<TeacherPO> getTeacherOfIns(int ins_id);
}

public interface FrameInfoDisplayService {
    public TypePO getType(int id);
    public ArrayList<TypePO> getType();
    public ModulePO getModule(int id);
    public ArrayList<ModulePO> getModule();
}

//调用 dataservice 的接口，方法均定义在父类 DatabaseSService 中
//模块记录数据服务，标红为此业务逻辑会用到的方法接口。
public interface ModuleDataService extends DatabaseService{
    public boolean insert(PO po) throws RemoteException;
    public boolean delete(int id) throws RemoteException;
    public boolean update(PO po) throws RemoteException;
    public PO find(int id) throws RemoteException;
    public ArrayList<PO> find(int condition, int id) throws RemoteException;
    public ArrayList<PO> findAll() throws RemoteException;
}

//课程类别记录数据服务
public interface TypeDataService extends DatabaseService{
    public boolean insert(PO po) throws RemoteException;
    public boolean delete(int id) throws RemoteException;
    public boolean update(PO po) throws RemoteException;
    public PO find(int id) throws RemoteException;
    public ArrayList<PO> find(int condition, int id) throws RemoteException;
    public ArrayList<PO> findAll() throws RemoteException;
}

//抽象课程记录数据服务
public interface Lesson_abstractDataService extends DatabaseService{
    public boolean insert(PO po) throws RemoteException;
    public boolean delete(int id) throws RemoteException;
    public boolean update(PO po) throws RemoteException;
    public PO find(int id) throws RemoteException;
    public ArrayList<PO> find(int condition, int id) throws RemoteException;
    public ArrayList<PO> findAll() throws RemoteException;
}

```

输出

类图

图 37

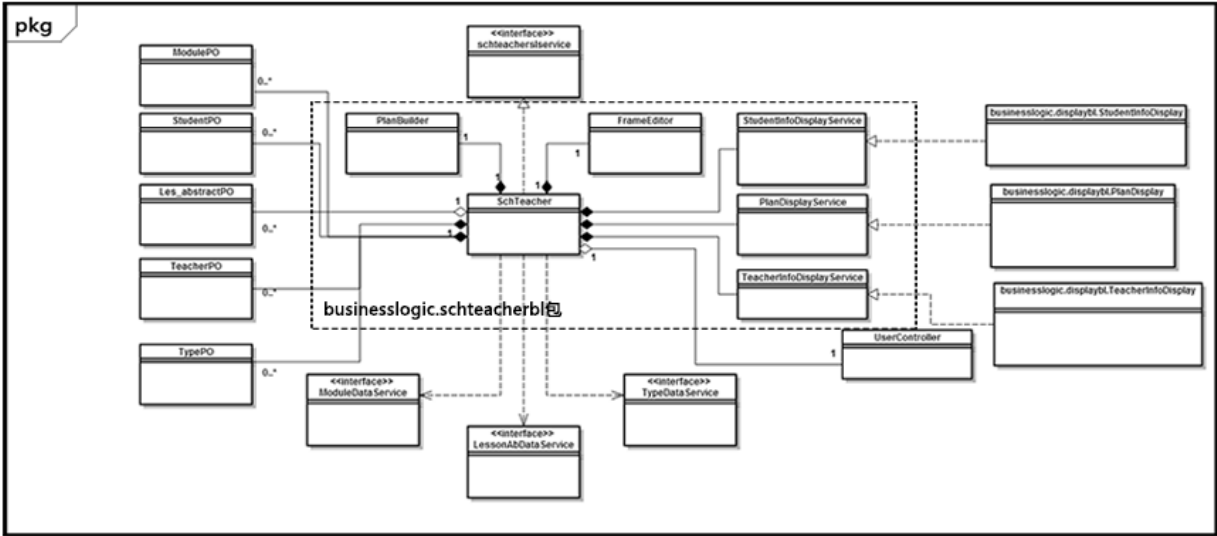


图 37:schteacher 模块类图

表 23: schteacherbl 模块各个类的职责

模块	职责
SchTeacher	负责院系教务老师的各种功能实现，包括制定教学计划（模块、类别、课程的增删改，查看学生列表、教师列表、各院系教学计划）
FrameEditor	协助学校教务老师编辑教学计划
PlanBuilder	根据抽象课程生成教学计划

（2）模块内部类的接口规范

表 24 Schteacher 类的接口规范

提供的服务(供接口)		
SchTeacher.addLesson	语法	public boolean addLesson(Lesson_abstractPO lesson);
	前置条件	学校教务老师登陆，课程所属类别已添加
	后置条件	系统添加一条抽象课程，根据所述课程类别更新教学框架
SchTeacher.addType	语法	public boolean addType(TypePO type);
	前置条件	学校教务老师已登录，该课程类别未添加
	后置条件	系统添加一条课程类别，在教学框架中对应课程模块下显示
SchTeacher.addModule	语法	public boolean addModule(ModulePO module);
	前置条件	学校教务老师已登录，该课程模块未添加
	后置条件	系统添加一条课程模块，在教学框架中显示

SchTeacher.modifyLesson	语法	public boolean modifyLesson(Lesson_abstractPO lesson);
	前置条件	学校教务老师登陆，该课程已添加
	后置条件	系统更新该抽象课程信息
SchTeacher.modifyType	语法	public boolean modifyType(TypePO type);
	前置条件	学校教务老师已登录，该课程类别已添加
	后置条件	系统更新该课程类别
SchTeacher.modifyModule	语法	public boolean modifyModule(ModulePO module);
	前置条件	学校教务老师已登录，该课程模块已添加
	后置条件	系统更新该课程模块
SchTeacher.deleteType	语法	public boolean deleteType(int id);
	前置条件	学校教务老师已登录，该课程类别已添加
	后置条件	删除该课程类别，数据库根据关联自动删除所有该类别下的具体课程与抽象课程
SchTeacher.deleteModule	语法	public boolean deleteModule(int id);
	前置条件	学校教务老师已登录，该课程模块已添加
	后置条件	删除该课程模块
SchTeacher.showStudentList	语法	public StudentPO[] showStudentList(int ins_id);
	前置条件	学校教务老师已登录，存在该 id 的院系
	后置条件	返回该院系学生列表
SchTeacher.showTeacherList	语法	public TeacherPO[] showTeacherList(int ins_id);
	前置条件	学校教务老师已登录，存在该 id 的院系
	后置条件	返回该院系教师列表
SchTeacher.showLessonList	语法	public LessonPO[] showLessonList(int ins_id);
	前置条件	学校教务老师已登录，存在该 id 的院系
	后置条件	返回该院系课程列表
SchTeacher.changePassword	语法	public boolean changePassword(char[] old, char[] newpassword)
	前置条件	学校教务老师已登录，旧密码正确且新密码合法
	后置条件	系统更新学校教务老师信息，提示修改密码成功
想要的服务（需接口）		
服务名		服务
UserblService.changePassword		用户修改密码
LessonDisplay.getLessonListofIns		系统得到该院系的所有课程
StudentInfoDisplay.getStudentListofins		系统得到该院系的所有学生
TeacherInfoDisplay.getTeacherList		系统得到该院系的所有教师
FrameEditor.addLesson		添加抽象课程
FrameEditor.deleteLesson		删除抽象课程
FrameEditor.modifyLesson		修改抽象课程
FrameEditor.addType		添加课程类别
FrameEditor.deleteType		删除课程类别
FrameEditor.modifyType		修改课程类别

FrameEditor.addModule	增加课程模块
FrameEditor.deleteModule	删除课程模块
FrameEditor.modifyModule	更新课程模块
PlanBuilder.buildPlan	构建教学计划
PlanBuilder.checkPlan	检验教学计划是否符合要求
DatabaseFactory.getPlanDatabase	得到 Plan 数据库服务的引用
DatabaseFactory.getTypeDatabase	得到 Type 数据库服务的引用
DatabaseFactory.getModuleDatabase	得到 Module 数据库服务的引用

表 25 FrameEditor 的接口规范

提供的服务(供接口)		
FrameEditor.addLesson	语法	public boolean addLesson(Lesson_abstractPO lesson_abstractPO);
	前置条件	教学框架已制定
	后置条件	增加抽象课程一条
FrameEditor.modifyLesson	语法	public boolean modifyLesson(Lesson_abstractPO lesson_abstractPO);
	前置条件	计划已制定, 该课程已添加
	后置条件	修改已制定计划中的指定一条抽象课程
FrameEditor.deleteLesson	语法	public boolean modifyLesson(Lesson_abstractPO lesson_abstractPO);
	前置条件	框架已制定
	后置条件	更新框架中的一条抽象课程
FrameEditor.addType	语法	public boolean addType(TypePO typePO);
	前置条件	无
	后置条件	增加一条课程类型
FrameEditor.modifyType	语法	public boolean modifyType(TypePO typePO)
	前置条件	计划已制定, 该课程类别已添加
	后置条件	修改一条教学计划
FrameEditor.deleteType	语法	public boolean deleteType(int id);
	前置条件	计划已制定, 该课程类别已添加
	后置条件	删除该课程类别, 数据库根据关联自动删除所有该类别下的具体课程与抽象课程
FrameEditor.addModule	语法	public boolean addModule(ModulePO module);
	前置条件	该课程模块未添加
	后置条件	系统添加一条课程模块, 在教学框架中显示
FrameEditor.deleteModule	语法	public boolean deleteModule(int id);
	前置条件	学校教务老师已登录, 该课程模块已添加
	后置条件	删除该课程模块
FrameEditor.modifyM	语法	public boolean modifyModule(ModulePO module);

odule	前置条件	学校教务老师已登录，该课程模块已添加
	后置条件	系统更新该课程模块
想要的服务（需接口）		
服务名		服务
LessonAbDataService.update(PO Plan)		在抽象课程表中更新对应课程的信息
LessonAbDataService.delete(int id)		在抽象课程表中删除对应抽象课程
LessonAbDataService.insert(PO Plan)		在抽象课程表中插入一条新的记录
TypeDataService.insert(PO type)		在课程类型表中插入一条新的记录
TypeDataService.update(PO type)		在课程类型表中更新一条记录
TypeDataService.delete(int id)		在课程类型表中删除一条记录
ModuleDataService.insert(PO Module)		在课程模块表中插入一条新的记录
ModuleDataService.update(PO Module)		在课程模块表中更新一条记录
ModuleDataService.delete(int id)		在课程模块表中删除一条记录

表 26 PlanBuilder 的接口规范

提供的服务(供接口)		
PlanBuilder.buildPlan	语法	public PlanVO buildPlan(ArrayList<Lesson_abstractPO> list)
	前置条件	无
	后置条件	返回构建好的计划
PlanBuilder.checkPlan	语法	public boolean checkPlan(ArrayList<Lesson_abstractPO> list)
	前置条件	无
	后置条件	返回计划是否符合教学框架学分标准
想要的服务（需接口）		
无		

(3)业务逻辑层的动态模型

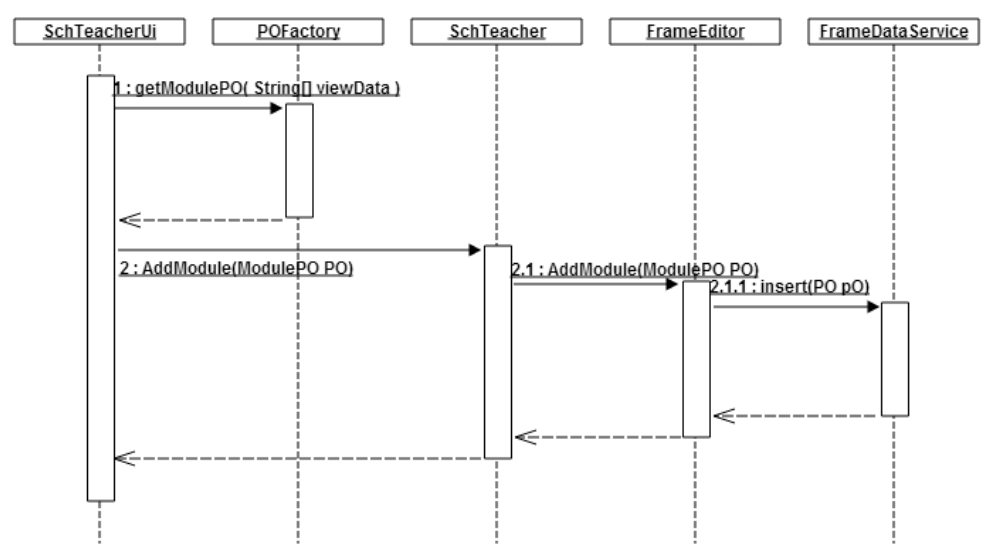


图 38：学校教务老师添加课程模块顺序图

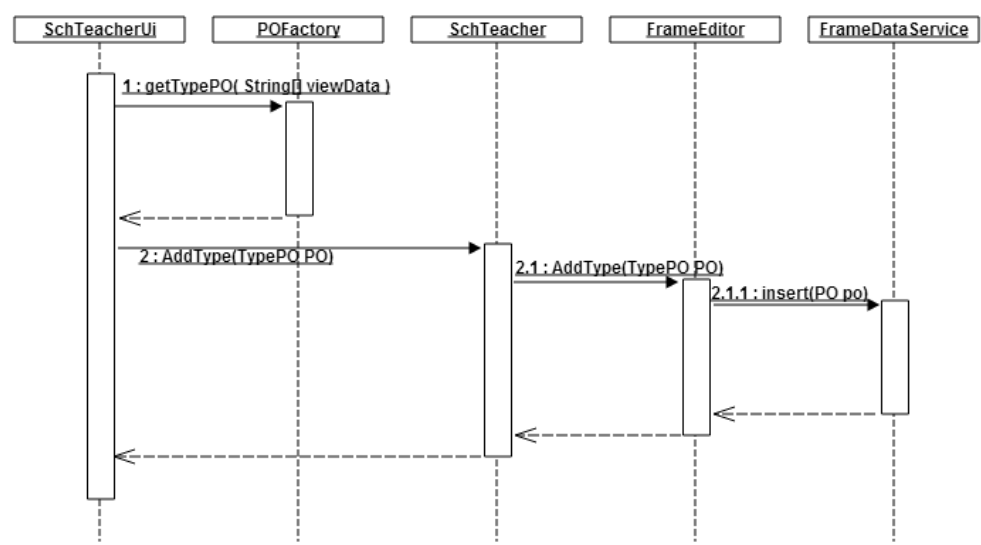


图 39：学校教务老师添加课程类别顺序图

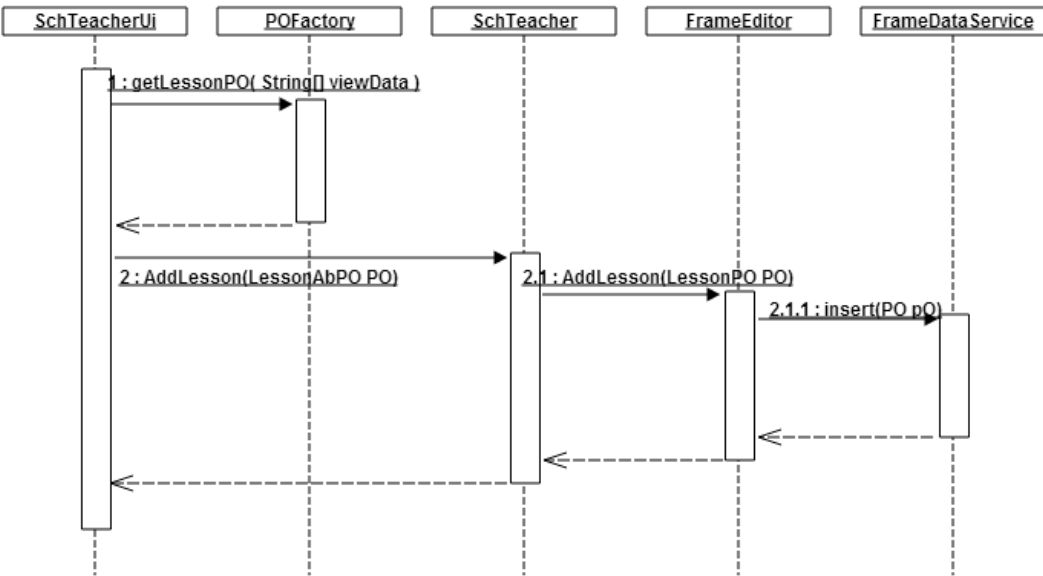


图 40： 学校教务老师添加抽象课程顺序图

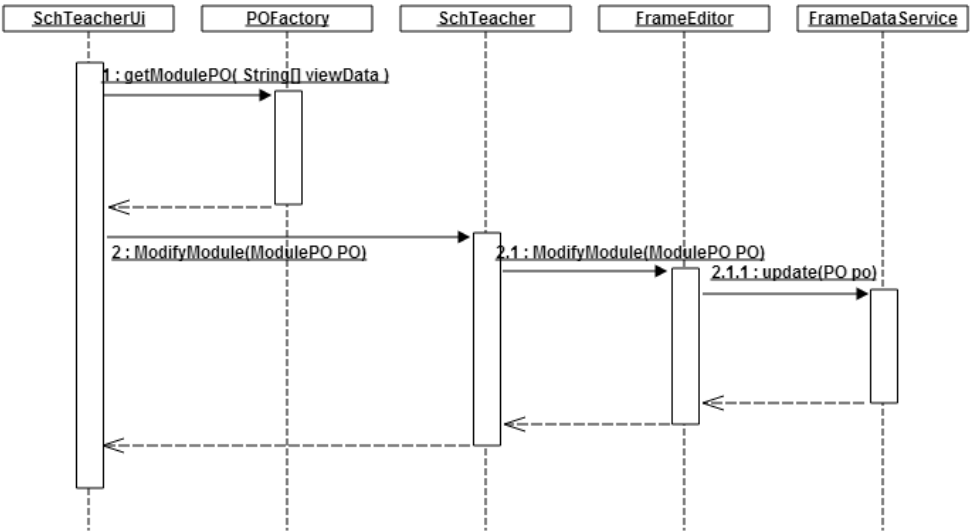


图 41： 学校教务老师修改课程模块顺序图

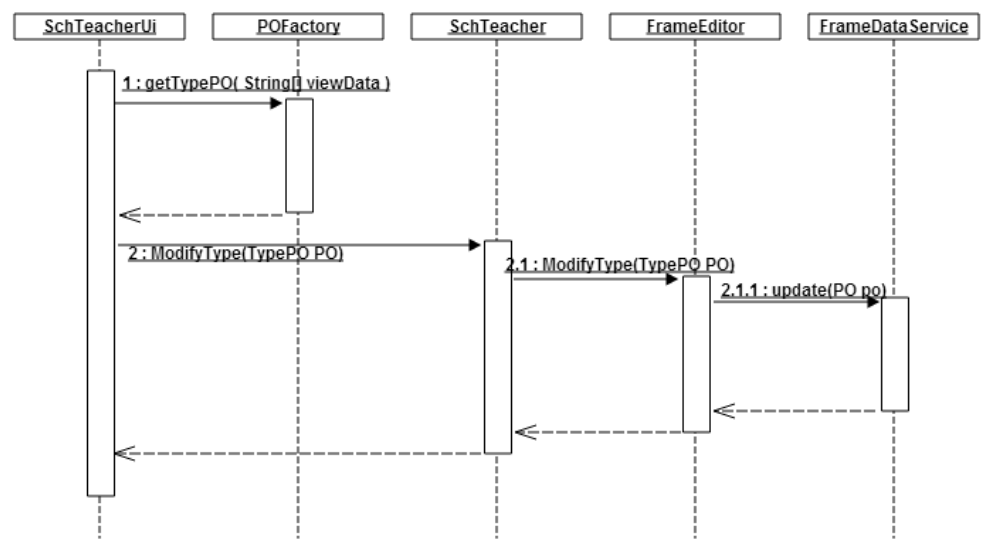


图 45：学校教务老师修改课程类别顺序图

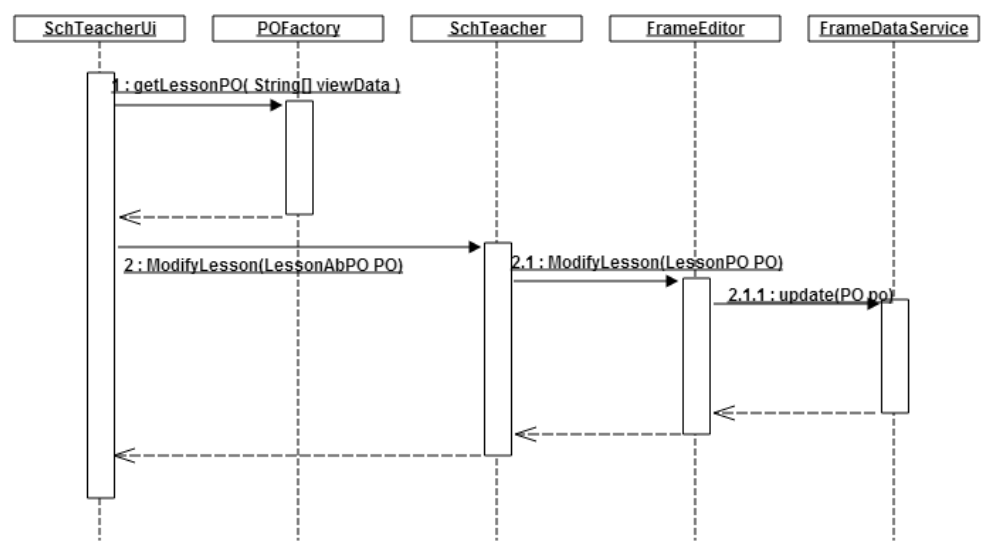


图 46：学校教务老师修改抽象课程顺序图

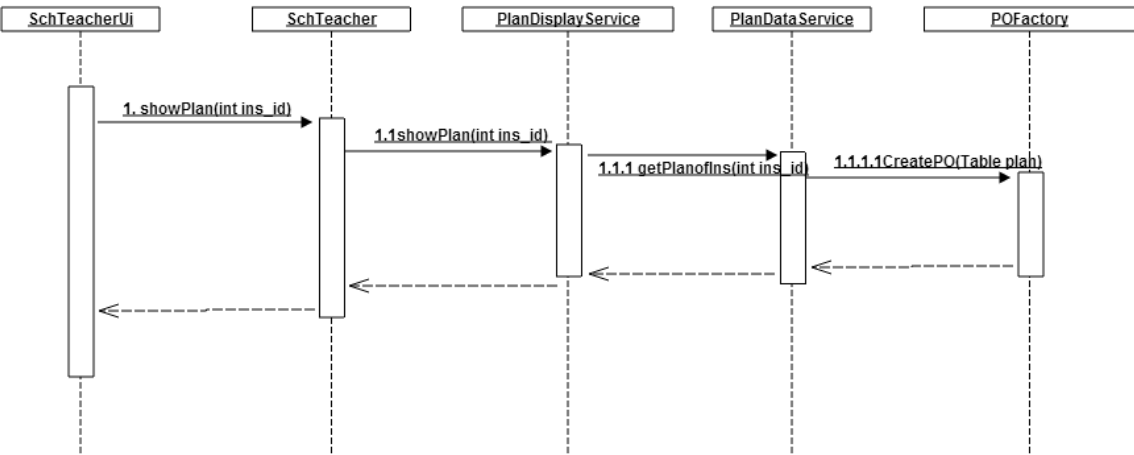


图 47: 学校教务老师查看院系教学计划顺序图

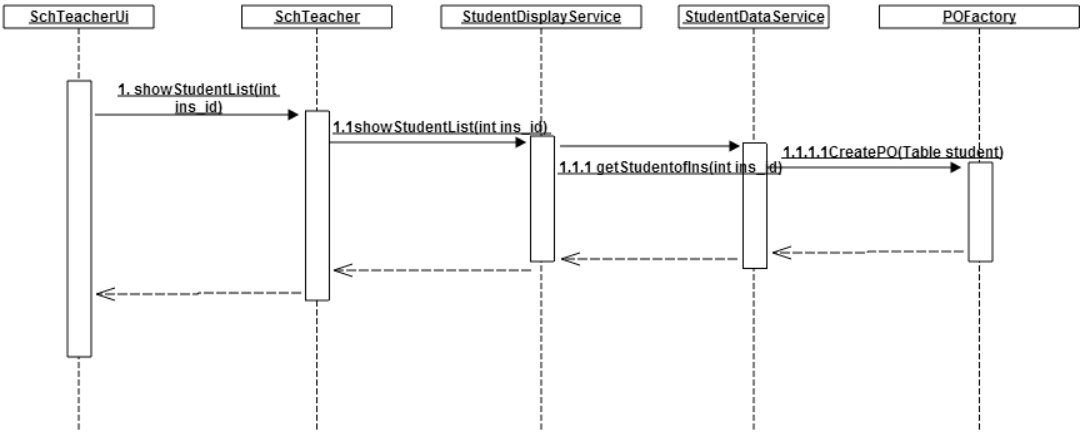


图 48:学校教务老师查看院系学生列表顺序图

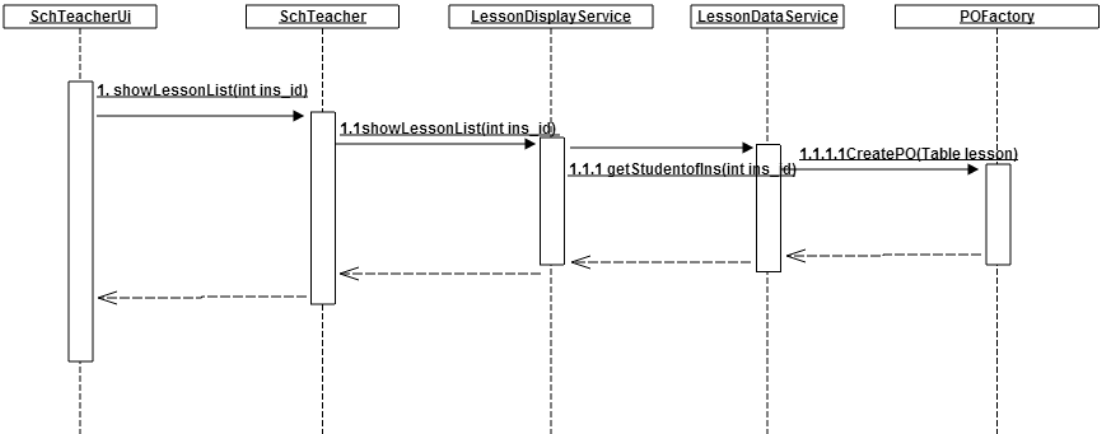


图 49:学校教务老师查看院系课程列表顺序图

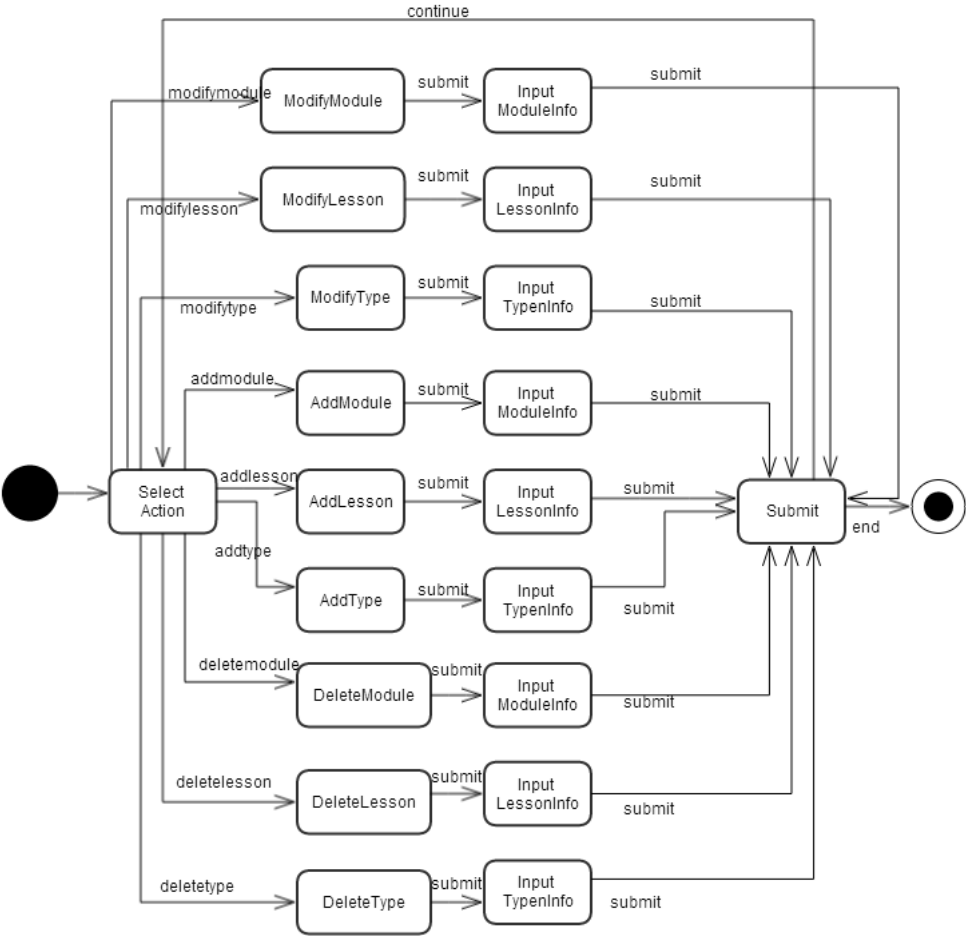


图 50: 学校教务老师编辑教学框架的对象状态图

4.3 数据层的整体结构说明

数据层实体分布在服务器端，如图 51 所示

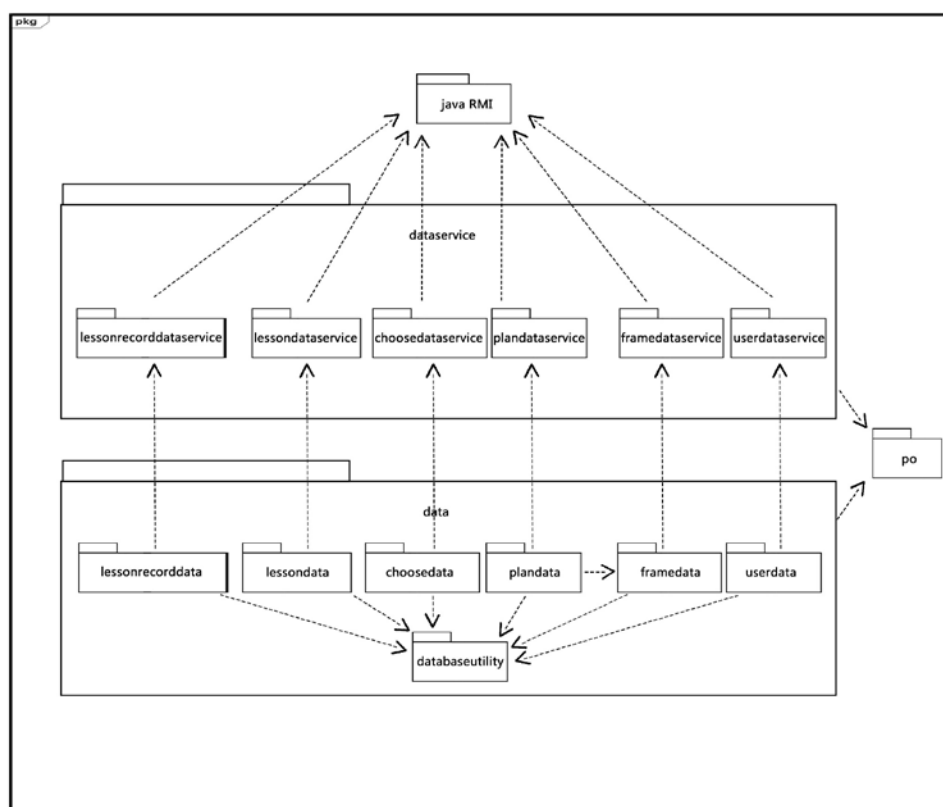


图 51: 数据层框架

其中所有的 data 都实现 DatabaseService 接口图 52 所示

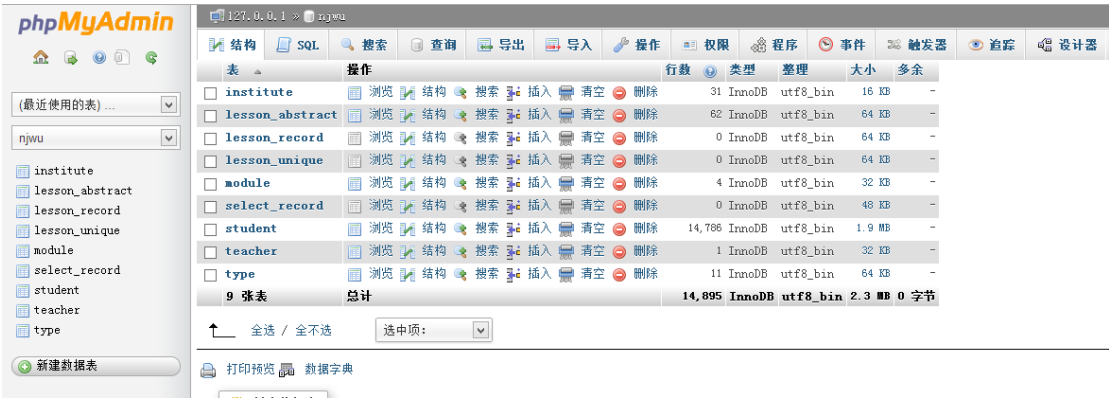
```
public interface DatabaseService {
    public boolean insert(PO po);
    public boolean delete(int id);
    public boolean update(PO po);
    public PO find(int id);
    public ArrayList<PO> find(int condition, int id);
    public ArrayList<PO> findAll();
}
```

图 52 数据层的接口

实现所有关于数据库操作。

4.4 数据库的详细设计

njwu 库总体概况如下组图所示



The screenshot shows the phpMyAdmin interface for the 'njwu' database. The 'Table' tab is selected, displaying a list of tables: institute, lesson_abstract, lesson_record, lesson_unique, module, select_record, student, teacher, and type. Each table entry includes icons for browsing, structure, SQL, search, insert, export, import, and operations, along with statistics for rows, type, engine, and size. A summary row at the bottom indicates 9 tables with a total of 14,895 rows.

表	操作	行数	类型	整理	大小	多余
<input type="checkbox"/> institute		31	InnoDB	utf8_bin	16 KB	-
<input type="checkbox"/> lesson_abstract		62	InnoDB	utf8_bin	64 KB	-
<input type="checkbox"/> lesson_record		0	InnoDB	utf8_bin	64 KB	-
<input type="checkbox"/> lesson_unique		0	InnoDB	utf8_bin	64 KB	-
<input type="checkbox"/> module		4	InnoDB	utf8_bin	32 KB	-
<input type="checkbox"/> select_record		0	InnoDB	utf8_bin	48 KB	-
<input type="checkbox"/> student		14,786	InnoDB	utf8_bin	1.9 MB	-
<input type="checkbox"/> teacher		1	InnoDB	utf8_bin	32 KB	-
<input type="checkbox"/> type		11	InnoDB	utf8_bin	64 KB	-
9 张表	总计	14,895	InnoDB	utf8_bin	2.3 MB	0 字节

图 52 数据库表列表

说明：每个表与每个 PO 一一对应，Lesson_unique 为 Lesson_abstract 的抽象化，前者受后者约束，因为同一门课程（课程号相同）在不同院系可能会有不同的信息（包括时间、地点、学生、老师）



The screenshot shows the 'Structure' tab for the 'institute' table. It lists two fields: 'Ins_Id' (int(11)) and 'name' (varchar(30)). Both fields are marked as 'PRIMARY' and 'UNIQUE' with associated icons for index, space, and full-text search.

#	名字	类型	整理	属性	空	默认	额外	操作
<input type="checkbox"/> 1	Ins_Id	int(11)		否	无			
<input type="checkbox"/> 2	name	varchar(30)	utf8_bin	否	无			

图 53:表 institute 结构



The screenshot shows the 'Structure' tab for the 'module' table. It lists four fields: 'Module_Id' (int(11)), 'Name' (varchar(10)), 'Min_Credit' (int(3)), and 'Max_Credit' (int(3)). 'Module_Id' is the primary key. 'Name' and 'Min_Credit' are indexed.

#	名字	类型	整理	属性	空	默认	额外	操作
<input type="checkbox"/> 1	Module_Id	int(11)		否	无	AUTO_INCREMENT		
<input type="checkbox"/> 2	Name	varchar(10)	utf8_bin	否	无			
<input type="checkbox"/> 3	Min_Credit	int(3)		否	无			
<input type="checkbox"/> 4	Max_Credit	int(3)		否	无			

图 54:表 module 结构



The screenshot shows the 'Structure' tab for the 'type' table. It lists eight fields: 'Type_Id' (int(11)), 'Module_Id' (int(11)), 'name' (varchar(32)), 'compulsory' (int(2)), 'term_start' (int(2)), 'term_end' (int(2)), 'min_credit' (int(3)), and 'max_credit' (int(11)). 'Type_Id' is the primary key. 'Module_Id' is indexed. 'name', 'compulsory', 'term_start', 'term_end', 'min_credit', and 'max_credit' are indexed.

#	名字	类型	整理	属性	空	默认	额外	操作
<input type="checkbox"/> 1	Type_Id	int(11)		否	无	AUTO_INCREMENT		
<input type="checkbox"/> 2	Module_Id	int(11)		是	NULL			
<input type="checkbox"/> 3	name	varchar(32)	utf8_bin	是				
<input type="checkbox"/> 4	compulsory	int(2)		否	无			
<input type="checkbox"/> 5	term_start	int(2)		否	无			
<input type="checkbox"/> 6	term_end	int(2)		否	无			
<input type="checkbox"/> 7	min_credit	int(3)		否	无			
<input type="checkbox"/> 8	max_credit	int(11)		否	无			

图 55:表 type 结构



The screenshot shows the 'Structure' tab for the 'lesson_abstract' table. It lists eight fields: 'Ins_Id' (int(11)), 'Les_Id_Ab' (int(20)), 'Type_Id' (int(11)), 'name' (varchar(128)), 'min_credit' (int(2)), 'max_credit' (int(2)), 'term_start' (int(2)), and 'term_end' (int(2)). 'Ins_Id' and 'Les_Id_Ab' are primary keys. 'Type_Id' is indexed. 'name', 'min_credit', 'max_credit', 'term_start', and 'term_end' are indexed.

#	名字	类型	整理	属性	空	默认	额外	操作
<input type="checkbox"/> 1	Ins_Id	int(11)		否	无			
<input type="checkbox"/> 2	Les_Id_Ab	int(20)		否	无			
<input type="checkbox"/> 3	Type_Id	int(11)		否	无			
<input type="checkbox"/> 4	name	varchar(128)	utf8_bin	否	无			
<input type="checkbox"/> 5	min_credit	int(2)		否	无			
<input type="checkbox"/> 6	max_credit	int(2)		否	无			
<input type="checkbox"/> 7	term_start	int(2)		否	无			
<input type="checkbox"/> 8	term_end	int(2)		否	无			

图 56:表 lesson_abstract 结构

127.0.0.1 » njwu » lesson_unique

浏览

结构

SQL

搜索

插入

导出

导入

操作

追踪

触发器

MySQL 返回的查询结果为空 (即零行)。(查询花费 0.0014 秒)

SELECT *
FROM `lesson_unique`
LIMIT 0 30

概要

[快速编辑]

[编辑]

[解释 SQL]

[创建 PHP 代码]

[刷新]

#	名字	类型	整理	属性	空	默认	额外	操作						
<input type="checkbox"/>	1 Les_Id	int(11)		否	无		AUTO_INCREMENT	<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	2 location	varchar(20)	utf8_bin	否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	3 tea	int(2)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	4 max_stu_num	int(10)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	5 cur_stu_num	int(10)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	6 state	int(2)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	7 Tea_Id	int(11)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	8 Ins_Id	int(11)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	9 day	varchar(8)	utf8_bin	否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	10 start	int(2)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	11 end	int(2)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	12 book	text	utf8_bin	否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	13 introduction	text	utf8_bin	否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	14 outline	text	utf8_bin	否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	15 Les_name	varchar(128)	utf8_bin	否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	16 Les_Id_Ab	int(20)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>

图 57: 表 lesson_unique 结构

127.0.0.1 » njwu » lesson_record

浏览

结构

SQL

搜索

插入

导出

导入

操作

追踪

触发器

MySQL 返回的查询结果为空（即零行）。（ 查询花费 0.0013 秒 ）

SELECT *
FROM `lesson_record`
LIMIT 0 30

☐概要 [快速编辑] [编辑] [解释 SQL] [创建 PHP 代码]

#	名字	类型	整理	属性	空	默认	额外	操作						
<input type="checkbox"/>	1 id	int(11)		否	无		AUTO_INCREMENT	<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	2 Les_Id	int(11)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	3 Stu_Id	int(11)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	4 score	int(4)		否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	5 Les_name	varchar(128)	utf8_bin	否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>
<input type="checkbox"/>	6 Stu_name	varchar(128)	utf8_bin	否	无			<div>修改 删除</div>	<div>浏览非重复值 (DISTINCT)</div>	<div>主键</div>	<div>唯一</div>	<div>索引</div>	<div>空间</div>	<div>全文搜索</div>

图 58: 表 lesson_record 结构

127.0.0.1 » njwu » select_record														
浏览 结构 SQL 搜索 插入 导出 导入 操作 追踪 触发器														
#	名字	类型	整理	属性	空	默认	额外	操作						
<input type="checkbox"/>	1 id	int(11)		否	无	AUTO_INCREMENT	修改 删除	浏览非重复值 (DISTINCT)	主键	唯一	索引	空间	全文搜索	
<input type="checkbox"/>	2 Stu_Id	int(11)		否	无		修改 删除	浏览非重复值 (DISTINCT)	主键	唯一	索引	空间	全文搜索	
<input type="checkbox"/>	3 Les_Id	int(11)		否	无		修改 删除	浏览非重复值 (DISTINCT)	主键	唯一	索引	空间	全文搜索	
<input type="checkbox"/>	4 Type	int(3)		否	无		修改 删除	浏览非重复值 (DISTINCT)	主键	唯一	索引	空间	全文搜索	

图 59: 表 select_record 结构



#	名字	类型	整理	属性	空	默认	额外	操作
1	Stu_Id	int (11)		否	无			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索
2	Ins_Id	int (11)		是	NULL			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索
3	name	varchar (128) utf8_bin		是	NULL			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索
4	gender	varchar (4) utf8_bin		否	无			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索
5	grade	int (2)		否	无			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索
6	password	varchar (20) utf8_bin		否	无			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索

图 60：表 student 结构



#	名字	类型	整理	属性	空	默认	额外	操作
1	Iea_Id	int (11)		否	无			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索
2	Ins_Id	int (11)		是	NULL			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索
3	name	varchar (20) utf8_bin		是				修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索
4	password	varchar (20) utf8_bin		否	无			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索
5	type	int (2)		否	无			修改 删除 浏览非重复值 (DISTINCT) 主键 U 唯一 索引 空间 T 全文搜索

图 61：表 teacher 结构

5.依赖视角

图 62 和图 63 是客户端和服务端各自的包之间的依赖关系

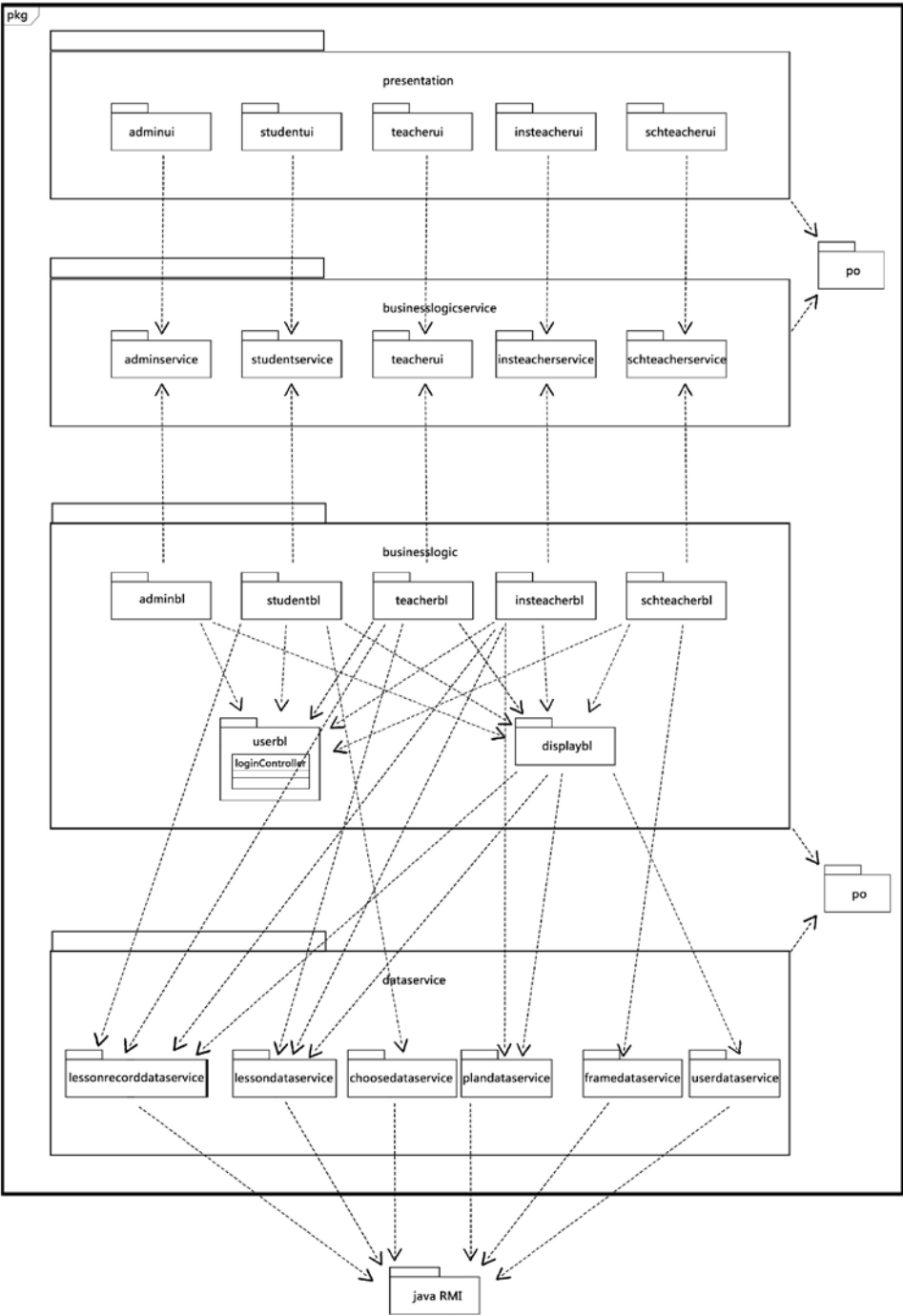


图 62：客户端各包之间的依赖关系

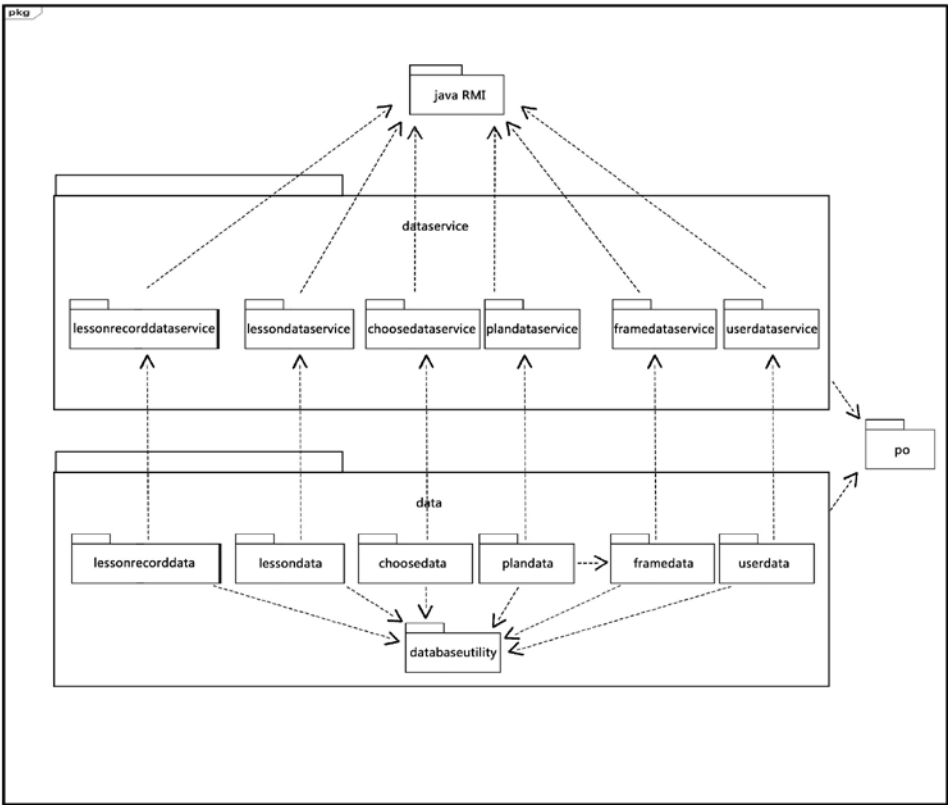


图 63：服务端各包之间的依赖关系