<课表小助手>

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2024年5月11日 | <1.0> | 初次编写软件架构文档 | 第8小组 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

目的 4

2. 用例视图 4

3. 逻辑视图 8

3.1 概述 8

3.2 在构架方面具有重要意义的设计包 9

4. 进程视图 14

5. 部署视图 15

6. 实现视图 16

7. 技术视图 18

8. 质量属性的设计 18

软件架构文档

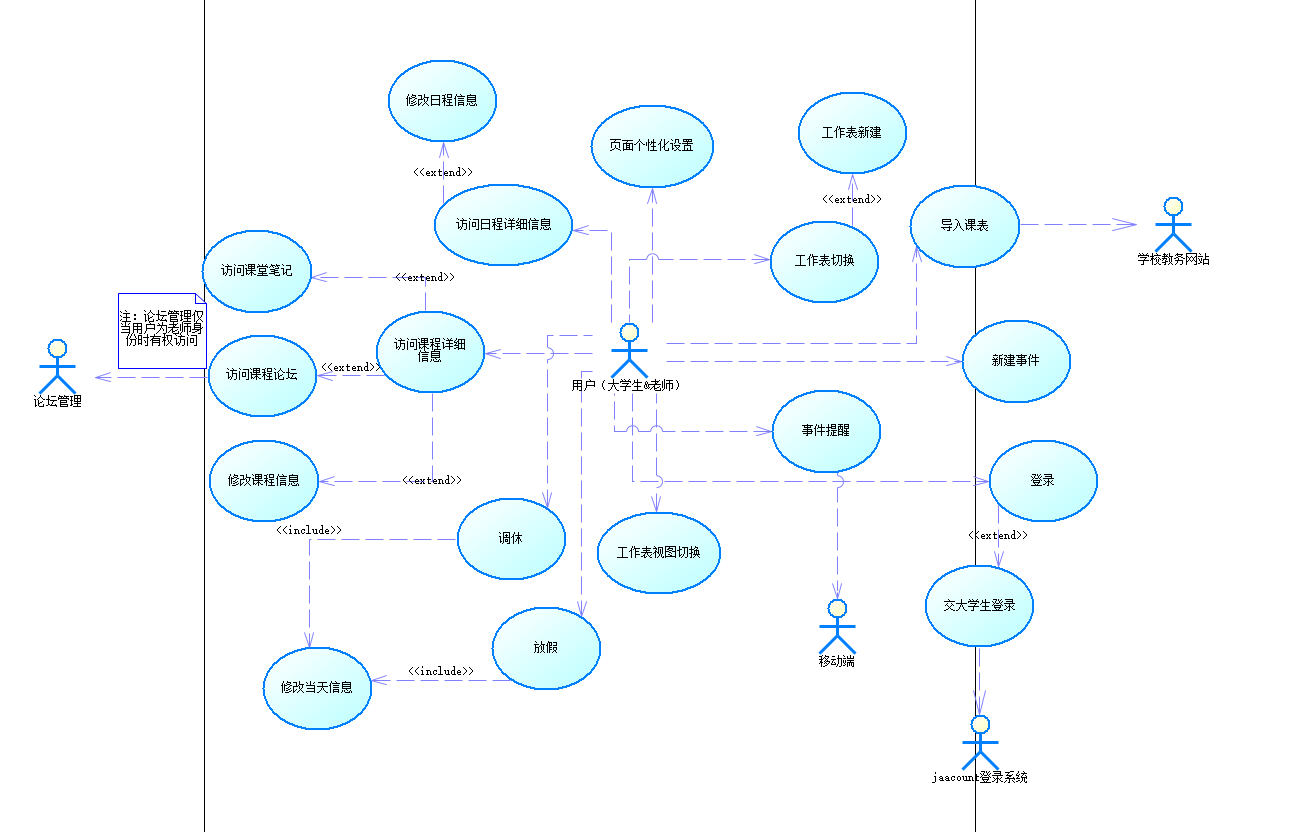
# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

对于课表小助手应用程序，软件架构文档的目的在于为开发团队和相关利益相关者提供关于系统构架的全面理解。这包括了系统的整体结构、各个组件的功能和交互方式、技术选型以及重要决策的记录和解释。特定读者可能包括开发团队的成员（开发人员、设计师、测试人员）、项目经理、产品经理以及其他项目利益相关者。他们可以使用此文档来了解系统的设计原理、决策背景以及各个组件之间的关系，以便更好地参与到开发、测试、部署和维护过程中。

# 用例视图



**基本流：**

当一个学生打开课表小助手时即发生该用例。学生通过课表小助手的按钮实现各种功能：

a.登录

b.导入课表、新建事件、页面个性化设置、工作表切换、工作表的视图切换、查看或更改账户信息

c.事件提醒、调休、访问课程详细信息、访问日程详细信息、导出课表必须在登录执行后再执行b类事件，必须在导入课表或者新建事件之后才能执行c类事件。

导入、新建、切换、调休、导出、个性化、账户按钮均在主界面更多选项中，点击更多会出现下拉框以便用户选择此功能，之后此操作不再赘述。

#### **2.1 登录**

1.学生打开软件后弹出登录界面（有注册按钮，交大学生不用注册，可以直接用jaccount登录）。

2.交大学生点击“使用jaccount登录”。

3.页面跳转到jaccount登陆系统，处理学生登录信息。

4.页面跳转到工作表主页面。登录过程完毕。

#### **2.2 导入课表**

1.学生点击导入课表按钮。

2.页面跳转到 选择要导入该课表的工作表 页面。

3.学生选择要导入课表的工作表。

4.页面跳转到学校教务网站课表界面，学生选择对应学期的课表，提交。

5.系统生成课表并填充至对应工作表。

6.返回“导入成功”对话框。导入课表过程完毕。

#### **2.3 新建事件**

1.学生点击新建按钮。

2.页面跳转至新建事件页面，提供标题、地点输入框，事件类型、要新建事件的工作表、事件是否重复选择框、是否为重要事件勾选框。

3.学生点击新建课程，提供标题、地点、要新建事件的工作表、是否重复、是否为重要事件，选择事件类型（课程/日程）。

4.1.若学生选择事件类型为课程，则需要填写课程代码、日期（以第n周周几计）、时间（以第n节课计）。

4.2.若学生选择时间类型为日程，则需要填写日期（以年月日计）、开始时间、结束时间（以时刻计）。

5.学生点击完成，系统将新事件添加至对应工作表中。新建过程完毕。

#### **2.4 页面个性化设置**

1.学生点击个性化按钮。

2.页面跳转至个性化设置界面，提供修改当前工作表名称的输入框，字体、课程块颜色、日程块颜色、起始时间、持续周数选择框，上课时间修改按钮，上传工作表背景按钮。

3.1.用户点击上课时间修改按钮，页面跳转至上课时间修改界面，提供每一节课的开始时间和结束时间设置，用户可以自行调整第n节课的开始和结束时间。点击保存，系统自动更改该工作表课程时间和布局。

3.2.用户点击上传工作表背景，系统调用手机相册权限，用户选择想要作为背景的图片，系统将该工作表的背景设置成对应图片。

4.用户点击保存，系统保存该修改，并保存本地数据。个性化设置过程完毕。

#### **2.5 工作表切换**

1.用户点击切换工作表按钮。

2.系统弹出切换工作表对话框，提供所有工作表的列表以及新建工作表按钮，当前所在工作表突出显示。

3.1.用户点击另一个工作表，系统切换到另一个工作表界面。工作表切换过程完毕。

3.2.用户点击新建工作表，系统切换到新建事件界面，进行新建事件流。新建事件流进行完毕后，直接切换到新建工作表界面，工作表切换过程完毕。

#### **2.6 工作表的视图切换**

1.用户在工作表界面可以看到月、周、日三个按钮，且工作表当前视图模式按钮突出显示。

2.用户点击另一个视图。

3.系统将工作表页面切换至对应视图模式，并且将该模式的按钮突出显示。工作表视图切换过程完毕。

补充：用户在月视图中点击某一日，会直接跳到该日的日视图。

#### **2.7 查看或更改账户信息**

1.用户点击账户按钮。

2.页面跳转至账户详细信息界面。展示用户圆形图像，以输入框形式展示账户名、昵称，以选择框形式展示性别、属地，并提供修改密码和退出登录按钮。（交大学生的账户名就是jaccount，昵称初始随机）

3.1.用户点击头像，系统调用手机相册权限，用户选择想要作为图像的图片，系统将其处理为圆形并展示。

3.2.用户点击账户名/昵称输入框，即可修改账户名和昵称。

3.3.用户点击性别/属地选择框，选择性别/属地。

3.4.用户点击修改密码，跳转到jaccount修改密码界面，输入用户名、原密码、新密码、确认新密码，点击完成，系统保存新密码并退出当前账号，返回登录界面。

3.5.用户点击退出登录，直接返回登录界面。

4.用户点击保存，系统保存当前修改和数据，并返回工作表界面，查看或更改账户信息过程完毕。

#### **2.8 事件提醒**

1.当用户某个事件还有15分钟开始时。

2.系统调用手机通知权限，在手机上弹出信息通知框，提示用户距事件xxx开始还有15分钟。

3.事件提醒过程完毕。

#### **2.9 调休**

1.当用户点击调休按钮。

2.页面跳转至调休设置界面。提供调整类型选择框（休假/上课）。

3.1.用户选择调整类型为休假，页面提供调整日期选择框，用户选择日期。系统将原日期的课程以背景图覆盖（但是数据并不删除），日程不变。

3.2.用户点击上课，页面提供调整日期和原课程日期选择框，用户选择两个日期，系统将原课程日期的课程数据复制（注意是复制，原日期的课程不变）到调整日期，并生成对应的课程框取代调整日期的课程框，日程不变。

4.用户点击完成，系统弹出提示框显示调整了的上课和休息时间。若系统检测到用户一次只设置了休假或者上课。提示框会增加提醒：您的调休只选择了上课/休假时间，还未设置休假/上课时间哦，是否保存当前修改？

5.1.用户点击返回，则继续（注意是继续，不是重新）进行调休设置。

5.2.用户点击保存，系统更新课程数据，删除原数据，调休过程完毕。

#### **2.10 导出**

1.用户点击导出按钮。

2.系统将课程表导出为pdf保存在app所在文件夹下。

3.导出过程完毕。

#### **2.11 访问课程详细信息**

1.用户在日视图或者周视图中点击对应的课程

2.页面跳到课程详细信息界面，系统提供对应的课程信息（课程名、地点、课程代码等），以及课堂笔记、课堂论坛和编辑按钮

3.1.\*(可能会实现)点击课堂笔记跳转到该课课堂笔记界面，可以上传自己的课堂笔记和访问自己之前上传的笔记

3.2.\*(可能会实现)点击课堂论坛跳转到该课课堂论坛界面，可以在论坛中提问和回答问题。

3.3.点击编辑按钮，跳转到修改信息界面，界面和新建课程相似，用户可修改相应信息并保存。系统保存修改的数据并更新工作表。

4.访问课程详细信息过程完毕。

#### **2.12 访问日程详细信息**

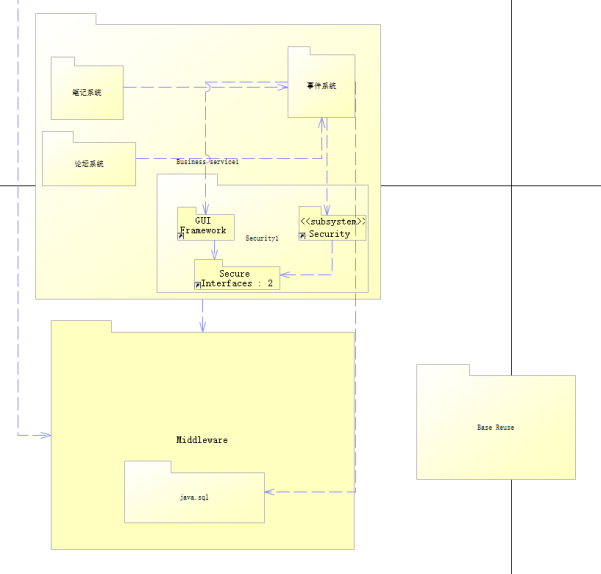
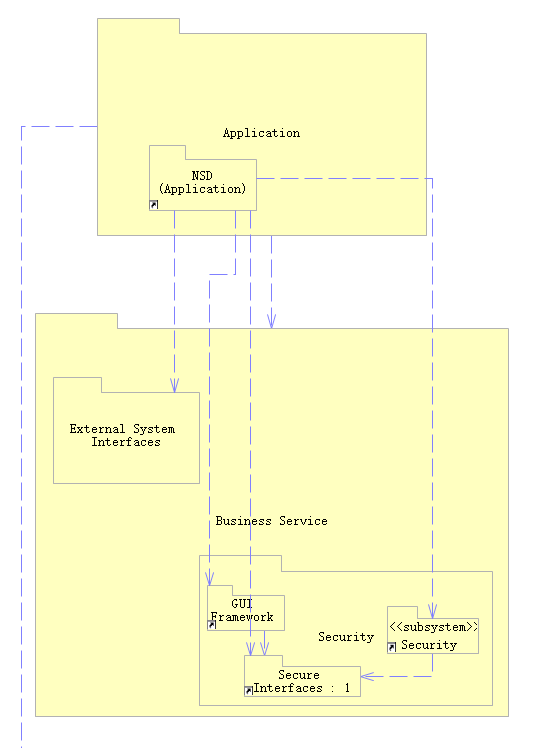
1. 用户在日视图或者周视图中点击对应的日程

2. 页面跳到课程详细信息界面，系统提供对应的日程详细信息和编辑按钮

3.点击编辑按钮，跳转到修改信息界面，界面和新建日程相似，用户可修改相应信息并保存。系统保存修改的数据并更新工作表。

# 逻辑视图

## 概述

****

**有三部分组成：**

Application， Business Services，Middleware

对于课表小助手应用程序，采用了典型的客户端-服务器（PC端）架构风格。在这种架构中，客户端应用（例如移动应用）与服务器（PC端）进行通信，服务器负责处理数据存储、逻辑处理和与客户端的交互。

1. **架构风格：**

- 客户端：NSD（Android应用）作为客户端，负责与用户进行交互，显示课程表信息、接收用户输入并将其发送到服务器。

- 服务器（由PC端执行功能）：后端服务负责处理客户端请求，执行业务逻辑，访问数据库，并将结果返回给客户端。

2. **模块分解和协同情况：**

- **客户端模块：**

- 用户界面模块：负责展示课程表、用户登录、课程管理等界面，并接收用户输入。

- 数据交互模块：负责与服务器进行通信，发送请求并处理响应。

- 本地存储模块：负责将从服务器获取的数据进行本地存储，以便在没有网络连接时也能访问数据。

- **服务器模块：**

- 路由与控制器模块：负责接收来自客户端的请求，并将其分发给相应的处理器进行处理。

- 业务逻辑模块：包含了课程管理、用户认证、提醒功能等业务逻辑的实现。

- 数据访问模块：负责与数据库进行交互，执行数据查询、插入、更新和删除操作。

- **数据存储模块：**

- 数据库：用于存储用户信息、课程信息、提醒设置等数据。

客户端应用通过数据交互模块向服务器发送请求，服务器接收请求后，由路由与控制器模块进行路由分发，然后交由业务逻辑模块处理具体的业务逻辑，包括课程管理、用户认证等。业务逻辑模块可能需要访问数据访问模块来执行数据库操作。处理完成后，服务器将结果返回给客户端，客户端接收并展示数据。

## 在构架方面具有重要意义的设计包

**1. Application**

**简要说明：**

NSD：课表小助手中与用户进行交互的所有功能，和外部接口（jaccount单点登录接口）相连，和GUI框架、安全系统和安全接口相连。也可以跨过Business Service层和中间件直接相连。

**重要类和包图：**

Application

|-- UserInterface: 用户界面相关类和组件

| |-- MainActivity: 应用的主界面，显示课程表和其他功能选项

| |-- CourseFragment: 课程表显示的Fragment

| |-- LoginActivity: 用户登录界面

| |-- ...

|

|-- DataManagement: 数据管理相关类和组件

| |-- DataManager: 数据管理器，负责管理本地数据和与服务器的数据交互

| |-- DatabaseHelper: 本地数据库的帮助类，用于执行数据库操作

| |-- APIClient: 与服务器进行通信的客户端，发送请求并处理响应

| |-- ...

|

|-- Utils: 实用工具类和方法

| |-- DateUtils: 日期时间处理工具类

| |-- NetworkUtils: 网络连接检测工具类

| |-- ...

**重要类和主要职责、操作、属性：**

1. **MainActivity**

- **简要说明：** 应用的主界面，显示课程表和其他功能选项。

- **主要职责：**

- 显示课程表和其他功能选项。

- 处理用户与界面的交互。

- **操作：**

- 加载和显示课程表。

- 响应用户的操作，如点击课程或功能选项。

- **属性：**

- 课程表数据源。

- 功能选项列表。

2. **CourseFragment**

- **简要说明：** 用于显示课程表的Fragment。

- **主要职责：**

- 渲染并显示课程表。

- 响应用户与课程表的交互。

- **操作：**

- 接收课程数据并在界面上展示。

- 提供课程的点击事件处理。

- **属性：**

- 课程数据。

3. **LoginActivity**

- **简要说明：** 用户登录界面。

- **主要职责：**

- 提供用户登录功能。

- 处理用户认证信息。

- **操作：**

- 用户输入认证信息。

- 向认证服务发送认证请求。

- **属性：**

- 用户名和密码输入框。

**2. Business Services**

**简要说明：**

Business service层包含笔记系统、事件系统、论坛系统和安全系统。笔记系统和论坛系统依赖于事件系统。事件系统和GUI框架和安全系统相连，安全系统通过调用安全接口来保证数据安全。同时事件系统还和中间件java sql相连，通过sql调用处理数据库。

**重要类和包图：**

BusinessServices

|-- CourseManagement: 课程管理相关类和组件

| |-- CourseManager: 课程管理器，负责课程的增删改查操作

| |-- Course: 课程类，表示一个课程的基本信息

| |-- ...

|

|-- UserManager: 用户管理相关类和组件

| |-- UserManager: 用户管理器，负责用户认证、注册等操作

| |-- User: 用户类，表示一个应用用户的基本信息

| |-- ...

|

|-- Reminder: 提醒功能相关类和组件

| |-- ReminderManager: 提醒管理器，负责设置和管理课程提醒

| |-- Reminder: 提醒类，表示一个提醒的基本信息

| |-- ...

|

|-- ...

```

**重要类和主要职责、操作、属性：**

1. **CourseManager**

- **简要说明：** 课程管理器，负责课程的增删改查操作。

- **主要职责：**

- 管理课程数据的增删改查。

- 提供课程操作的接口。

- **操作：**

- 添加、删除、修改课程信息。

- 查询课程信息。

- **属性：**

- 课程列表。

2. **UserManager**

- **简要说明：** 用户管理器，负责用户认证、注册等操作。

- **主要职责：**

- 处理用户认证和注册。

- 管理用户信息。

- **操作：**

- 处理用户登录和注册请求。

- 管理用户信息。

- **属性：**

- 用户认证信息。

3. **ReminderManager**

- **简要说明：** 提醒管理器，负责设置和管理课程提醒。

- **主要职责：**

- 管理课程提醒设置。

- 触发提醒事件。

- **操作：**

- 设置课程提醒。

- 响应提醒事件。

- **属性：**

- 提醒设置信息。

**3. Middleware**

**简要说明：**

Middleware包含了课表小助手应用程序的中间层组件，负责处理客户端与服务器之间的通信和数据交互。

**重要类和包图：**

Middleware

|-- Network: 网络通信相关类和组件

| |-- NetworkManager: 网络管理器，负责检测网络连接状态和处理网络请求

| |-- Request: 请求类，表示一个网络请求的基本信息

| |-- Response: 响应类，表示一个网络响应的基本信息

| |-- ...

|

|-- Authentication: 用户认证相关类和组件

| |-- AuthManager: 认证管理器，负责处理用户登录和认证操作

| |-- AuthRequest: 认证请求类，表示一个用户认证请求的基本信息

| |-- AuthResponse: 认证响应类，表示一个用户认证响应的基本信息

| |-- ...

|

|-- DataSerialization: 数据序列化与反序列化相关类和组件

| |-- Serializer: 序列化器，负责将数据对象序列化为字节流

| |-- Deserializer: 反序列化器，负责将字节流反序列化为数据对象

| |-- ...

以上是课表小助手应用程序中三个重要包的简要说明和示意图。每个包都包含了各自的重要类和组件，负责不同的功能模块，以实现应用程序的整体功能。

理解了。我会逐个列出每个重要类的名称、简要说明，并尽可能包括其主要职责、操作和属性的说明。

**重要类和主要职责、操作、属性：**

1. **NetworkManage**

- **简要说明：** 网络管理器，负责检测网络连接状态和处理网络请求。

- **主要职责：**

- 监测和管理网络连接状态。

- 处理网络请求和响应。

- **操作：**

- 检测网络连接状态。

- 发送和接收网络请求。

- **属性：**

- 网络连接状态。

2. **AuthManager**

- **简要说明：** 认证管理器，负责处理用户登录和认证操作。

- **主要职责：**

- 处理用户登录和认证。

- 管理用户身份验证。

- **操作：**

- 处理用户登录请求。

- 验证用户身份。

- **属性：**

- 认证信息。

3. **Serializer**

- **简要说明：** 序列化器，负责将数据对象序列化为字节流。

- **主要职责：**

- 将数据对象转换为字节流。

- 用于数据的传输和存储。

- **操作：**

- 将对象转换为字节流。

- 从字节流反序列化对象。

- **属性：**

- 无。

# 进程视图

1. **轻量级进程：**

- **用户界面进程**：负责与用户进行交互，接收用户输入并显示课程表信息。

- **通信模式**：消息传递，通过事件处理或回调函数与其他进程通信。

- **课程数据处理进程**：负责处理课程数据，包括添加、删除、编辑课程等操作。

- **通信模式**：消息传递，接收用户界面进程发送的操作请求，返回处理结果。

- **提醒服务进程**：负责提醒用户课程时间，可以是定时提醒或者根据用户设置的提醒条件触发。

- **通信模式**：中断，根据课程时间触发提醒事件，发送提醒消息给用户界面进程。

- **数据存储进程**：负责存储课程数据，可以使用数据库或者文件系统进行存储。

- **通信模式**：消息传递，接收课程数据处理进程发送的数据操作请求，进行数据的读写操作。

2. **重量级进程**：

- **课程推荐系统**：根据用户的学习情况和偏好推荐相关课程。

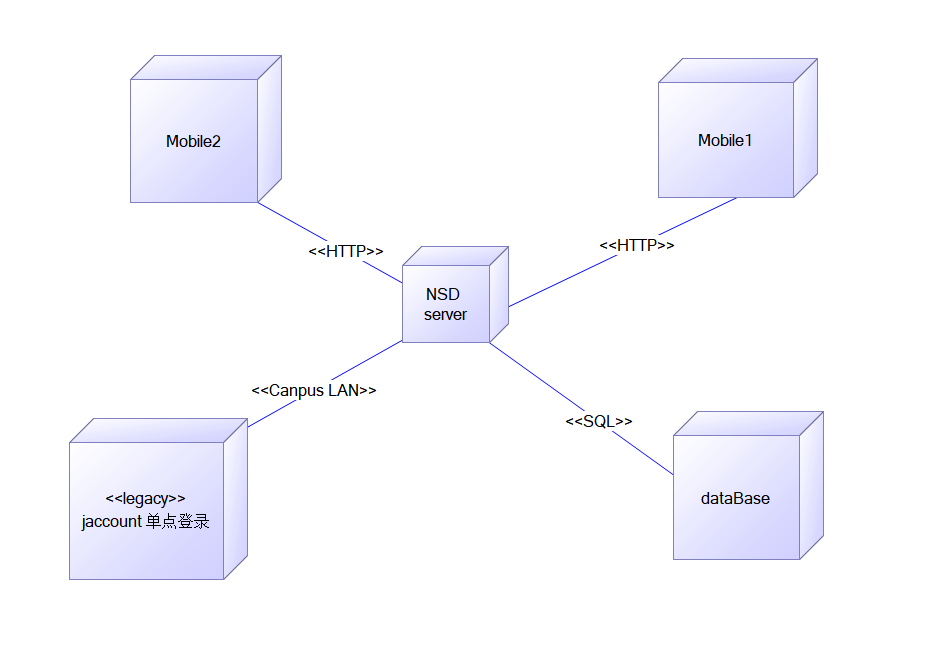
- **通信模式**：会合，收集来自用户界面进程和课程数据处理进程的用户数据和课程信息，进行推荐计算后将推荐结果发送给用户界面进程。

- **数据同步服务**：将课程表数据同步到云端或其他设备上。

- **通信模式**：消息传递，接收课程数据处理进程发送的数据同步请求，将数据同步到指定的目标上。

这种分解方式下，轻量级进程主要负责用户交互和基本的数据处理，而重量级进程则负责一些复杂的计算和跨系统的数据同步等任务。不同进程之间通过消息传递、中断和会合等通信模式进行交互和协作，以完成整个课表小助手系统的功能。

# 部署视图



1. **Mobile1，2**

学生/老师用户的联网手机

1. **NSD Server**

NSD Server是主要的Unix服务器，所有用户都可以通过网络与NSD Server取得连接

1. **dataBase**

dataBase是存在于PC端的数据库，用于存储用户设置的日程表以及个性化设置等数据

1. **jaccount单点登录**

jaccount是已有的登录接口，交大用户通过连接校园网或者VPN登录账号，并获取自己的课程信息

**以下是物理网络配置以及进程到物理节点的映射：**

**1. Mobile1, 2**

- 物理节点：两台联网手机

- 进程映射：Mobile App 进程分布在两台手机上运行

**2. NSD Server**

- 物理节点：Unix 服务器

- 进程映射：NSD Server 进程运行在Unix 服务器上

**3. dataBase**

- 物理节点：PC端的数据库服务器

- 进程映射：数据库进程运行在PC端的数据库服务器上

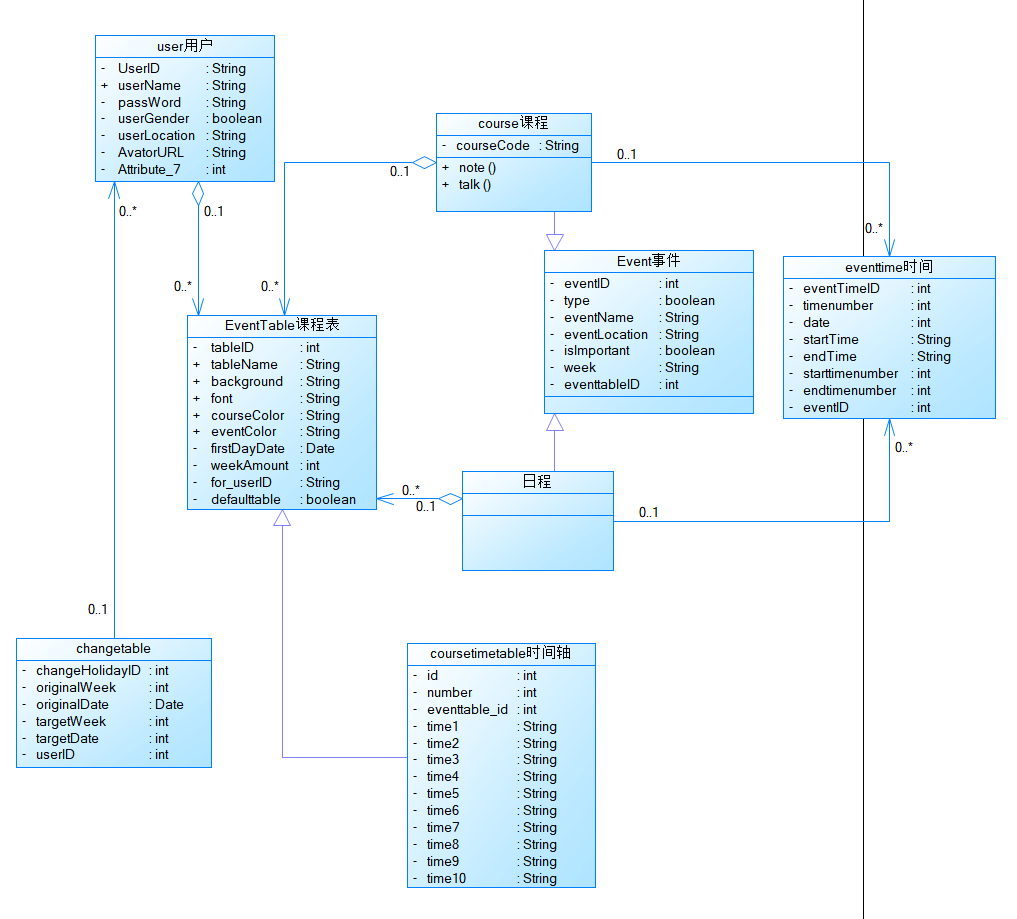
**4. jaccount单点登录**

- 物理节点：jaccount 登录接口所在的服务器

- 进程映射：jaccount 登录接口进程运行在登录接口所在的服务器上

# 实现视图

1. 首先我们小组采用了 template method 模式，我们首先创建了一个 event 类，就是事件类，它是所有课程类和 schedule 日程类的父类。其中包含其通用信息（时间，地点，是否重要等等）。
2. 实现一个event模板类,在其中编码了事件基本的策略、规则、流程。在针对具体问题进行精化时，我们用课程类和日程类继承event模板类，来扩展这个Template中的一些细节，比如课程编码，访问笔记和论坛。
3. 之后我们创建了 user 用户类，以及 EventTable 类来完善该模式的设计，一个用户对应多张课表，所以每张课表中有一个 for\_userID 来对应用户。
4. 为了调休的实现方便，我们专门加入了一个类 changetable 来为用户提供调休的方式



将系统的整体结构、软件分解、以及重要构件进行如下的说明：

**整体结构：**

系统采用了模板方法（Template Method）设计模式。整体结构可以分为以下几个部分：

1.**Event（事件）类**：作为所有课程类和日程类的父类，包含了通用信息（时间、地点、重要性等）。

2.**User（用户）类**：表示系统中的用户。

3.**EventTable（事件表）类**：用于管理用户的课表，一个用户可以拥有多张课表。

4.**ChangeTable（调休表）类**：提供调休功能的实现。

**软件分解为层和子系统情况：**

1. **模板方法层：**包括了模板方法的实现，即Event类作为模板类，定义了通用的事件处理流程、规则和策略。

2. **业务逻辑层：**包括了用户类（User）、事件表类（EventTable）、调休表类（ChangeTable）等，用于管理系统中的用户、课表和调休功能。

3. **数据访问层：**可能包含与数据库的交互，用于存储和检索用户、事件等信息。

**重要构件说明：**

1. **Event类：**作为模板方法的核心构件，定义了事件的通用处理逻辑，是整个系统的基础。

2. **User类：**代表系统中的用户，是系统的核心之一，用于管理用户信息和操作。

3. **EventTable类：**管理用户的课表信息，提供对课表的增删改查等操作，与User类相互关联。

4. **ChangeTable类：**为用户提供调休功能，是系统的重要功能之一，能够影响用户的日程安排和行为。

整体上，系统采用模板方法设计模式，通过将通用逻辑提取到父类中，并通过子类进行细化和定制化，实现了系统的灵活性和可扩展性。同时，用户、课表、调休等功能模块清晰，各自职责明确，便于系统的维护和扩展。

# 技术视图

编程语言：前端react native 使用jsx语言；后端采用java语言

开发工具：前端Android Studio，后端使用IDEA，数据库使用MySQL Workbench

框架：前端使用React native 后端使用 Spring Boot框架

# 质量属性的设计

**1. 性能：**

- **合理的后端数据处理：**设计高效的数据结构和算法，以支持快速的数据查询和更新操作，确保系统能够快速响应用户请求。

- **前端优化**：采用轻量级的前端框架和技术，减少页面加载时间和资源消耗，提升用户的访问体验。

- **缓存机制：**利用缓存技术，如Redis等，缓存频繁访问的数据，减轻数据库的压力，提高系统的响应速度。

**2. 可扩展性：**

- **模块化设计：**将系统拆分为多个独立的模块，每个模块都有清晰的责任和接口，便于新增功能或修改现有功能而不影响其他部分。

- **微服务架构：**采用微服务架构，将系统拆分为多个小型服务，每个服务都可以独立部署和扩展，提高系统的灵活性和可扩展性。

**3. 可靠性：**

- **容错机制：**设计系统时考虑到各种异常情况的处理，实现合理的容错机制，以保证系统在面对异常情况时能够保持稳定运行。

- **监控和日志：**引入监控和日志系统，实时监控系统的运行状态，及时发现和解决问题，提高系统的可靠性和稳定性。

**4. 易用性：**

- **用户界面设计**：设计简洁直观的用户界面，符合用户的使用习惯和期望，提高用户的满意度和使用体验。

- **用户反馈机制**：为用户提供反馈渠道，收集用户的意见和建议，及时调整和改进系统，以提高易用性。

**5. 可移植性：**

- **平台无关性：**尽量减少对特定平台的依赖，采用跨平台的技术和工具，以提高系统的可移植性和适用性。

- **云原生架构：**将系统设计为云原生架构，使用容器化技术和微服务架构，便于在不同的云平台上部署和运行。

特别在安全性和保密性方面：

- **安全机制：**采用身份验证、权限控制等安全机制，保护用户数据的安全性和完整性，防止未经授权的访问和攻击。

- **数据加密**：对用户敏感数据进行加密存储和传输，确保用户数据的保密性，在数据传输和存储过程中防止数据泄露。

课表小助手的软件架构应该兼顾性能、可扩展性、可靠性、易用性和可移植性等方面，同时在安全性和保密性方面也要有所考虑，以保证系统的稳定运行和用户信息的安全。