第2次作业模版-概要设计

自助计费式中央温控系统系统概要设计

基于UML 的面向对象建模方法

小组：304a-1

组长：张梓良

组员1：杨晨

组员2：苗雨

组员3：吉奥博

组员4：朱馨妍

组员5：魏陈正树

日期：2024/5/26

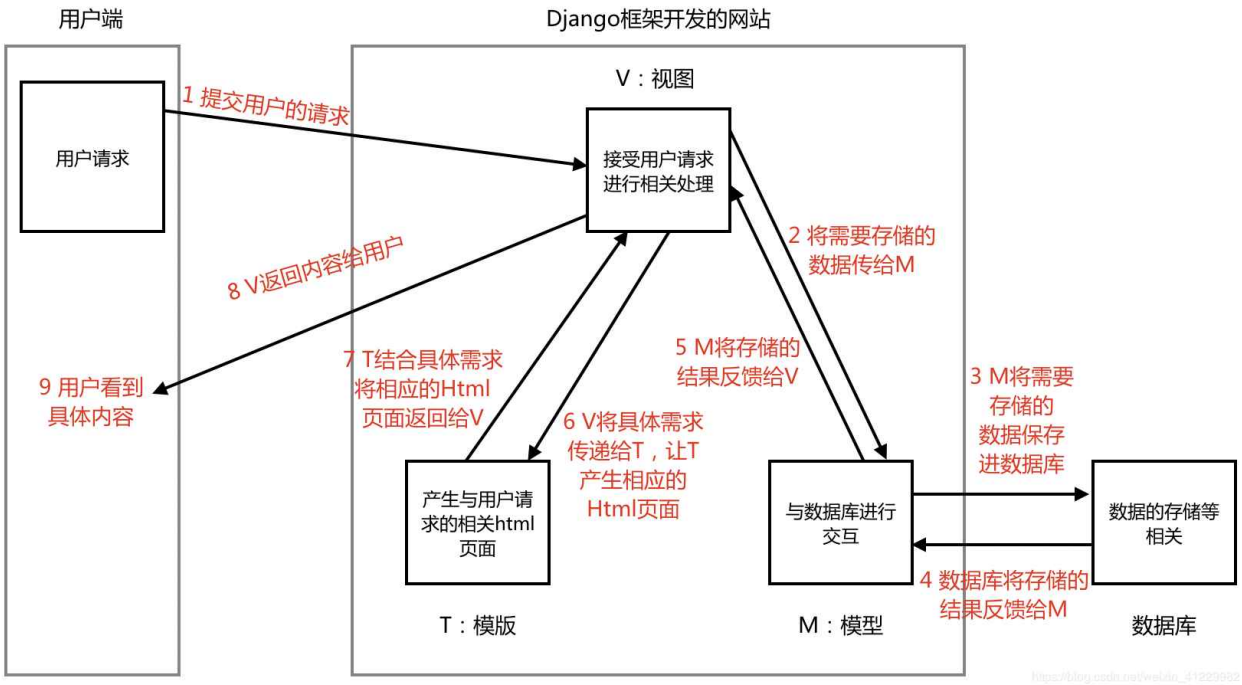
---

1. 软件架构

`Django` 架构

- **M（Model层）**：与MVC中的M功能相同，负责和数据库交互，进行数据处理。
- **V（View层，业务逻辑层）**：接收请求，进行业务处理，返回应答。
- **url.py（Controller，控制器层）**，与MVC中的C功能相同，负责接受用户请求并分发对应的视图。
  - 当然这个url层是我们自己定义的，我们认为url.py充当的是一个路由的角色，书本上对于控制器层的标准定义是：主要用于接受界面层提交的请求并管理和分析这些请求业务的类型，进而转发给业务逻辑层的业务对象。
- **T（Template，用户界面GUI层）**：与MVC中的V功能相同，负责封装构造要返回的html。

1.1 软件架构示意图



1.2 合理性解释

系统整体软件架构示意图的说明

该系统采用了经典的MVC（Model-View-Controller）设计模式，并结合了Django框架的特点。以下是对该架构的合理性及其运行机制的详细说明：

架构合理性

1. **分层设计**：
   - 采用MVC模式将系统分为三层：视图层（V：视图）、模型层（M：模型）和模板层（T：模板）。这种分层设计有助于模块化开发，提高代码的可维护性和可扩展性。
   - 用户端、视图层、模型层和数据库各司其职，职责分明。
2. **低耦合高内聚**：
   - 各个模块之间的依赖关系较低，提高了系统的内聚性。
   - 视图层负责接受用户请求并进行处理，模型层负责与数据库进行交互，模板层负责生成HTML页面。
3. **扩展性和可维护性**：
   - 各个层次可以独立修改和扩展，不会影响其他层次的功能。
   - 视图层、模型层和模板层可以根据需求单独进行优化和调整。

运行机制

1. **用户请求提交（步骤1）**：
   - 用户通过浏览器向服务器发送请求。

2. **视图层处理请求（步骤2）**：
    - 视图层接收到用户请求后，首先对请求进行相关处理，并将需要存储的数据传递给模型层。
3. **模型层存储数据（步骤3、4）**：
    - 模型层接收到数据后，将数据保存到数据库中。
    - 数据库完成存储操作后，将存储结果反馈给模型层。
4. **模型层反馈结果（步骤5）**：
    - 模型层将存储结果传递回视图层。
5. **视图层生成响应内容（步骤6）**：
    - 视图层根据具体需求，将相应的处理结果传递给模板层，以生成相应的HTML页面。
6. **模板层生成HTML页面（步骤7）**：
    - 模板层结合具体需求，生成相应的HTML页面，并返回给视图层。
7. **视图层返回内容给用户（步骤8）**：
    - 视图层将生成的HTML页面内容返回给用户。
8. **用户查看具体内容（步骤9）**：
    - 用户在浏览器中查看具体的响应内容。

总结

整个Django框架下的网站架构流程合理，步骤清晰，符合Web开发的常见模式。该架构利用了MVC模式的优点，使得系统具有良好的扩展性和可维护性。通过这种架构设计，开发者可以更方便地对各个模块进行独立开发和测试，同时保证了系统的稳定性和性能。

2. 分层结构说明

在 `Django` 框架的项目中，各个部分有明确的职责分工。以下是对不同对象职责的详细解释：

1. 哪些对象负责接收前端发过来的请求；

负责接收前端发过来的请求的对象是**视图（Views）**。

- 在 `Django` 中，视图是一个函数或类方法，用于处理HTTP请求并返回HTTP响应。
- 视图接收到请求后，根据请求的URL和请求的方法（如GET、POST等）进行处理。

示例：

```
# 在 air_condition/views.py 中
from django.shortcuts import render
from django.http import HttpResponse
```

```
4
5 def log_in(request):  # 用户登录界面
6     # 处理请求
7     return render(request, 'log-in.html')
```

2. 哪些对象负责创建对象的实例；

负责创建对象实例的对象是**模型（Models）**。

- 模型定义了数据库的结构，包括字段和行为。
- 创建对象实例通常是在视图中调用模型的方法来实现的。

示例：

```
1 # 在 air_condition/models.py 中
2 from django.db import models
3
4 class Scheduler(models.Model):
5     # 存储5个房间,房间开始时的状态都是3--“SHUTDOWN”关机状态
6     rooms = models.ManyToManyField('Room', related_name='scheduler')
7     # Room和Scheduler是多对多关系
8
9 # 在 air_condition/views.py 中
10 from air_condition.models import Scheduler
11
12 scheduler = Scheduler.objects.using('default').create()
```

3. 哪些对象负责处理请求的具体要求；

负责处理请求的具体要求的对象是**视图（Views）**。

- 视图根据请求类型（例如，GET、POST）和参数，执行具体的业务逻辑。
- 视图可以调用模型的方法来获取或修改数据，并调用模板来生成响应。

示例：

```
1 # 在 air_condition/views.py 中
2 from django.shortcuts import render
3 from air_condition.models import Scheduler
4
5 scheduler = Scheduler.objects.using('default').create()
6 def mon_submit(request):
```

```
    scheduler.set_para(high, low, default, fee_h, fee_l, fee_m, mode)
    scheduler.power_on()
    scheduler.start_up()
    return HttpResponseRedirect('/monitor')
```

4. 哪些对象负责数据的同步和存储。

负责数据同步和存储的对象是**模型（Models）**和**数据库（Database）**。

- 模型通过定义字段和方法，与数据库进行交互。
- `Django`的ORM（对象关系映射）将模型对象与数据库表关联，负责数据的增删改查。

示例：

```
# 在 air_condition/models.py 中
from django.db import models


# 数据的创建和存储
temp_room = Room.objects.using('default').create(request_id=self.request_id,
mode=self.mode,

target_temp=self.default_target_temp,
                                                 fee_rate=self.fee_rate_m,
                                                 init_temp=32 if self.mode == 2
 else 8)

# 数据的更新
temp_room.save()

# 数据的删除
Room.objects.all().delete()
```

总结

- **接收前端请求**：视图（Views）
- **创建对象实例**：模型（Models）
- **处理请求具体要求**：视图（Views）
- **数据同步和存储**：模型（Models）和数据库（Database）

`Django` 框架通过这种分工，保证了代码的清晰性和可维护性。视图负责业务逻辑和请求处理，模型负责数据定义和数据库交互，模板负责呈现数据给用户。

---

3. 系统的界面设计

正文：本次作业仅要求各小组给出以下环节的界面设计：

a. 顾客使用空调的控制面板；

b. 前台营业员办理入住和结账的界面；

c. 系统管理员监控空调使用的状态界面（可选，Bonus 10%）；

说明：界面设计重点在于展示布局以及局部的各功能模块的位置（如果已经有界面，可以截图展示并说明）；

这里给出了可参考的界面设计：

a. https://www.toutiao.com/article/7280050808686133795/?log_from=3e14959a31fe5_1701671301403；

b. https://www.toutiao.com/article/7308202020518085160/?log_from=6c208ce5cc433_1701671447806；

c. https://www.toutiao.com/article/7307879808015581730/?log_from=5342047fcdddd_1701671592111。

3.1 登录界面设计

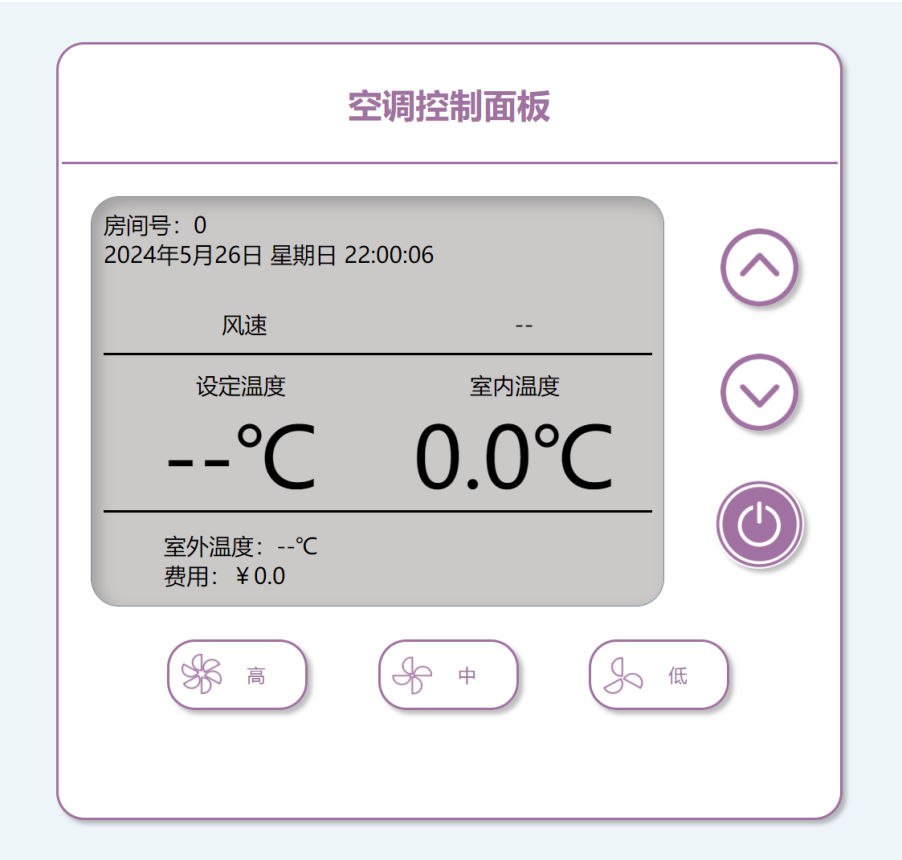
用户，前台营业员，空调管理员可以通过输入账号和密码后，点击登录提交POST请求，实现相对应的界面的跳转

- 用户跳转到空调的控制面板
  - 用户名：room+房间号
  - 密码：123
- 前台营业员跳转到打印账单及详单页面
  - 用户名：qiantai
  - 密码：123
- 空调管理员跳转到监控空调运行状态界面
  - 用户名：admin
  - 密码：123

3.2 空调的控制面板设计

3.2.1 首先进入登录界面，输入用户办理入住的房间名作为账号名，格式为“room+房间号”，输入密码“123”即可成功登录空调的控制面板

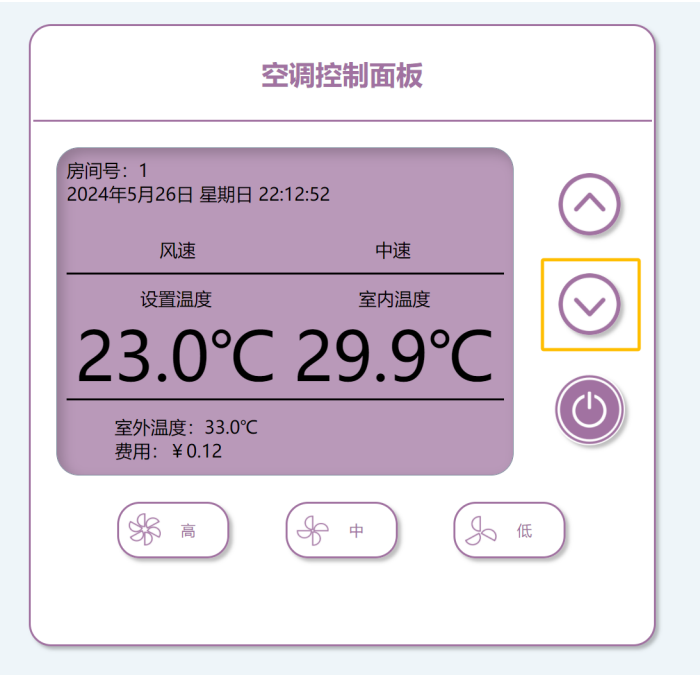
3.2.2 空调控制面板的设计模拟了现实生活中常见的空调遥控器的面板，以大屏幕显示空调温度，环境温度以及费用等信息。用户认证成功进入控制面板后，空调仍未开启，屏幕处于灰色未亮起的状态



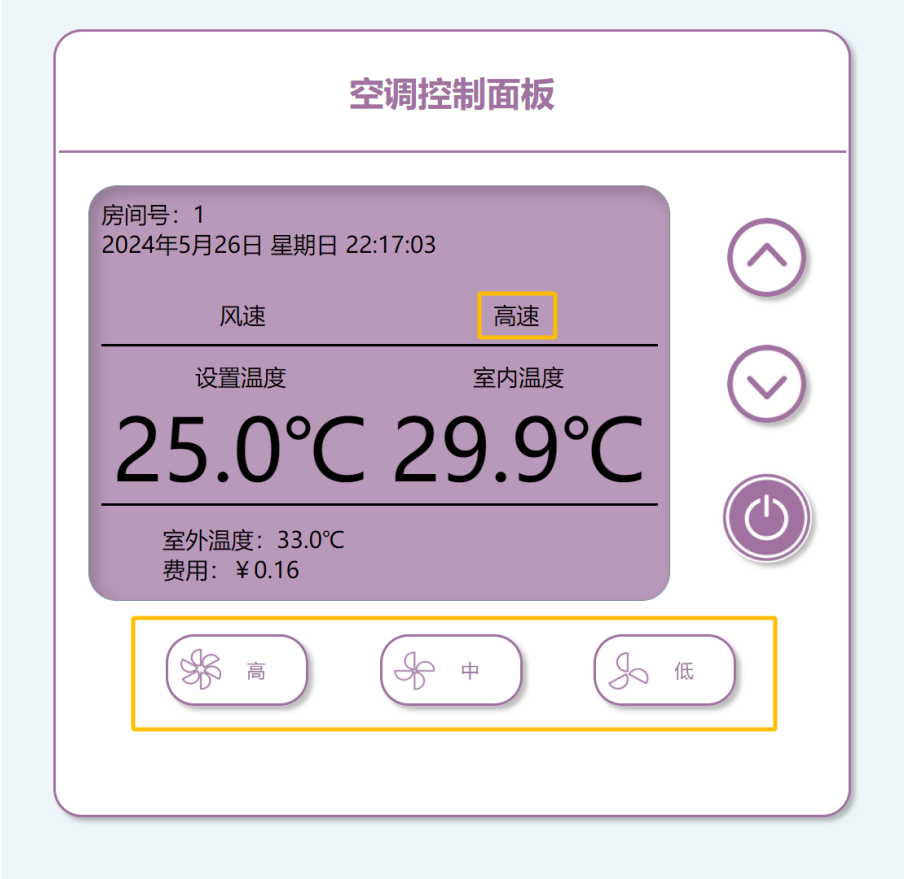
3.2.3 点击电源按钮，实现空调的启动，空调会显示当前时间，默认空调设置温度和风速，并实时更新使用空调产生的费用和室内温度的变化

一段时间后：

3.2.4 点击一次“升高温度”按钮，设置的温度会调高一度；点击一次“降低温度”按钮，设置的温度会调低一度



3.2.5 通过点击“高”，“中”，“低”三个按钮来调整空调的风速，空调的费用也随之改变

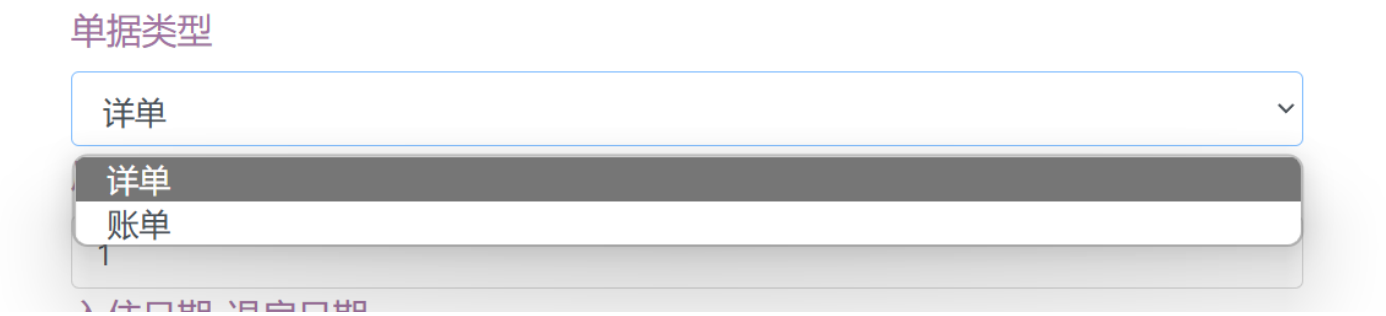


3.3 前台营业员打印账单及详单页面设计

3.3.1首先进入登录界面，输入管理员账号名“qiantai”，输入密码“123”即可成功登录前台营业员界面

3.3.2前台营业员操控界面如下：

单据类型可以选择详单or账单：



输入需要查询的房间号，并提供客户的入住日期和退房日期以供查询客户这段时期的花费：

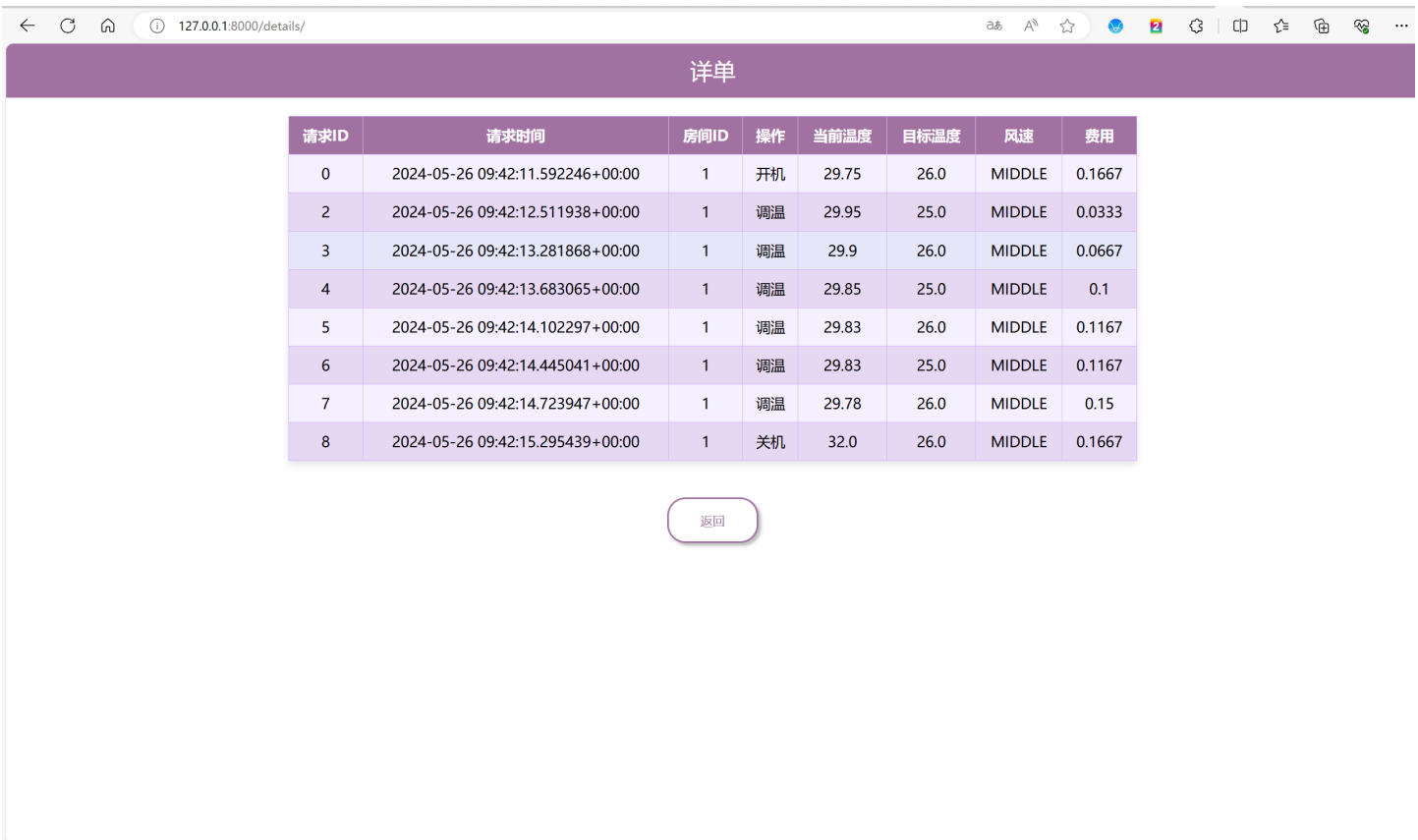
3.3.3打印账单

单据类型选择账单，输入房间号，选择入住日期和退房日期，并点击打印按钮，界面跳转，显示账单信息，

包含操作员、单位、房间号和总费用:

3.3.3打印详单

单据类型选择详单，输入房间号，选择入住日期和退房日期，并点击打印按钮，界面跳转，显示详单信息,显示用户的所用使用记录，包括请求ID、请求时间、房间ID、操作类型、温度、风速和费用：



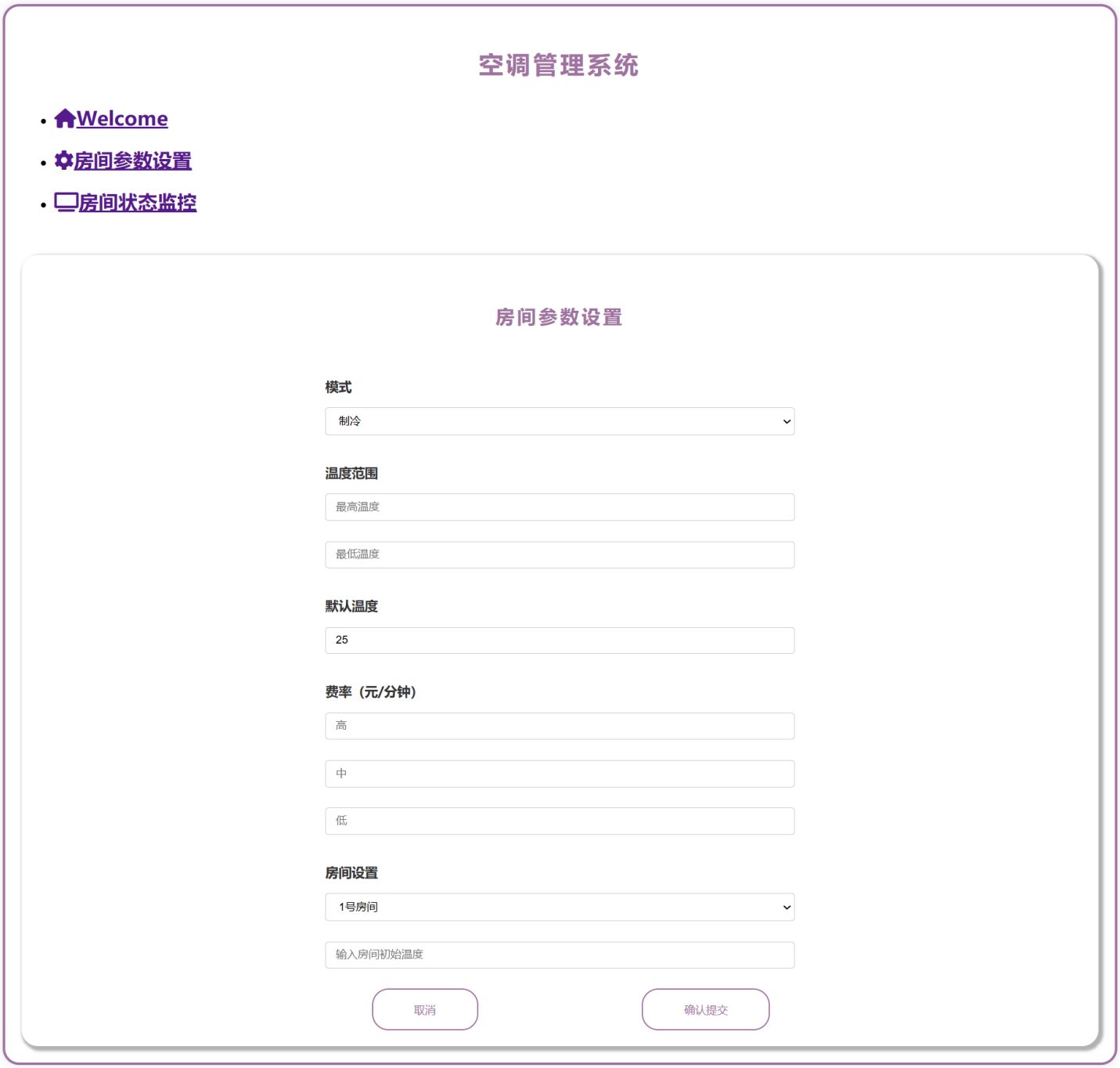
请求ID	请求时间	房间ID	操作	当前温度	目标温度	风速	费用
0	2024-05-26 09:42:11.592246+00:00	1	开机	29.75	26.0	MIDDLE	0.1667
2	2024-05-26 09:42:12.511938+00:00	1	调温	29.95	25.0	MIDDLE	0.0333
3	2024-05-26 09:42:13.281868+00:00	1	调温	29.9	26.0	MIDDLE	0.0667
4	2024-05-26 09:42:13.683065+00:00	1	调温	29.85	25.0	MIDDLE	0.1
5	2024-05-26 09:42:14.102297+00:00	1	调温	29.83	26.0	MIDDLE	0.1167
6	2024-05-26 09:42:14.445041+00:00	1	调温	29.83	25.0	MIDDLE	0.1167
7	2024-05-26 09:42:14.723947+00:00	1	调温	29.78	26.0	MIDDLE	0.15
8	2024-05-26 09:42:15.295439+00:00	1	关机	32.0	26.0	MIDDLE	0.1667

3.4 监控空调运行状态界面设计（可选，10%Bonus）

3.4.1 首先进入登录界面，输入管理员账号名“admin”，输入密码“123”即可成功登录管理员界面

3.4.2 管理员界面如下，界面左上方是管理员操作的菜单栏，进入管理员界面之后默认展示如下欢迎界面：

3.4.3 通过选择菜单栏中的房间参数设置选项，管理员可以进入房间参数设置界面，设置每个房间的空调参数，界面如下：

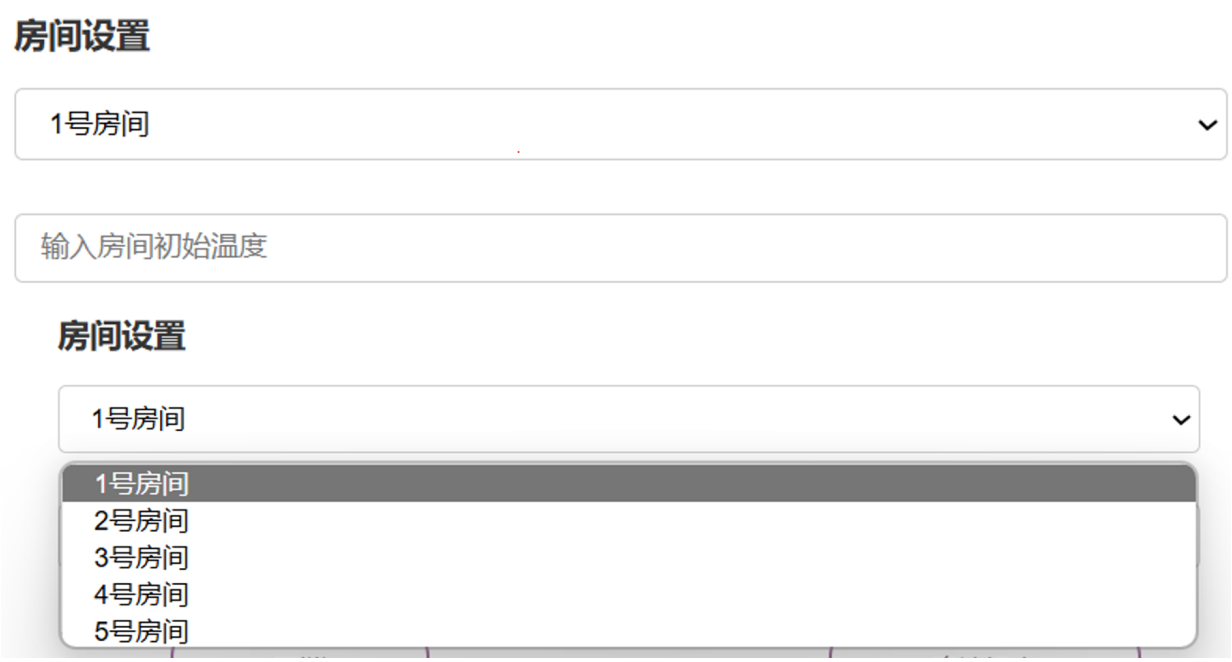


空调运行模式可以选择制冷模式和制热模式：

可以设置房间空调温度范围，例：16-35摄氏度

可以设置房间空调开启时的默认温度，同时，可以设置用户选择的高中低三档风速对应的费率(元/分钟)。例：默认温度25摄氏度，高、中、低三档费率分别对应：1、0.5、0.3元/分钟

可以分别设置每个房间打开空调时的初始温度：



确认提交使得参数配置生效，取消则不做改动：

3.4.4 通过选择菜单栏中的房间状态监控选项，管理员可以进入房间状态监控界面，查看每个房间的空调的运行状态，包括具体的参数，界面如下：

---

4. 系统动态结构设计

4.1 用例: 用户使用空调

4.1.1 已知条件

用例消息及对应返回值，操作契约

消息名称	参数列表	消息含义	返回值	操作契约描述
TurnOn	RoomID	打开空调	Temperature, FanSpeed, CostPerHour	打开指定房间的空调，返回当前温度、风速和每小时费用
change_target_temp	RoomID, Temperature	设置温度	Temperature, FanSpeed, CostPerHour	设置指定房间的空调温度，返回当前温度、风速和每小时费用
Set_fan_Speed	RoomID, FanSpeed	设置风扇风速	Temperature, FanSpeed, CostPerHour	设置指定房间的空调风速，返回当前温度、风速和每小时费用
InqueryCost	RoomID	查询当前总费用	TotalCost	查询指定房间空调的当前总费用
TurnOff	RoomID	关闭空调	OK	关闭指定房间的空调

4.1.2 对象设计：TurnOn(RoomID)

操作契约

- **前置条件**：用户已经登录系统，并且已经按下空调的开机键。
- **后置条件**：
  a. 空调开机，空调开关状态被修改。
  b. 计费状态被修改。
  c. 服务记录被更新。
  d. 首次启动空调时需要调用 poweron 激活中控机 Scheduler。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户启动空调的操作中，哪个对象应首先接收这个请求？ **解决方案**：Server 作为首先接收 TurnOn(RoomID) 消息的软件对象，主要负责处理来自用户的启动请求并协调后续的激活流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的空调激活操作，包括中控机的首次启动。

3. **对象间的关联关系**

Server 与 Scheduler 建立关联，Scheduler 负责实际的空调启动和状态管理。

Scheduler 进一步与 StatisticController 关联，后者负责记录启动事件和更新服务记录。

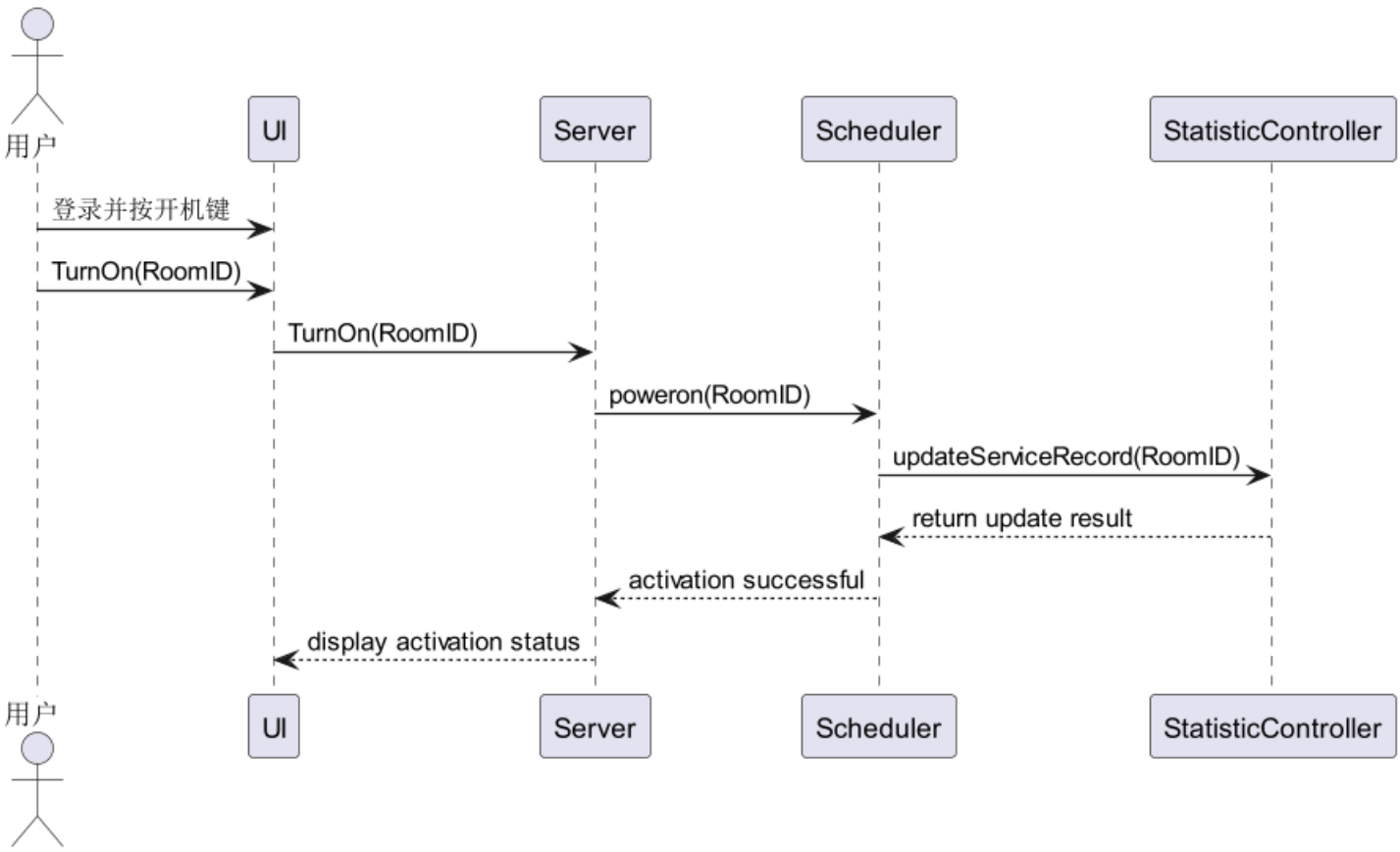
4. **需要修改或初始化的对象属性**

StatisticController 负责记录空调的启动状态、计费状态的修改以及服务记录的更新。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化空调的状态和服务记录，确保在系统重启后能够恢复最后的服务状态。

6. **Sequence Diagram**：



序列图说明

- 用户登录系统并按下空调的开机键，满足前置条件。
- 用户通过用户界面发送启动空调的请求。
- Server 接收请求并指示 Scheduler 进行空调的激活，包括首次启动时的 poweron 调用。

- Scheduler 请求 StatisticController 更新服务记录和计费状态。
- StatisticController 处理请求并将更新结果返回给 Scheduler。
- Scheduler 将激活成功的确认信息反馈给 Server。
- Server 将激活状态显示在UI上，让用户知道空调已成功开启。

4.1.3 对象设计：change_target_temp(RoomID, Temperature)

操作契约

- **前置条件**：用户已经登录系统，并且已经按下空调的开机键。
- **后置条件**：
  a. 空调的目标温度被修改至用户指定的温度。
  b. 系统返回当前设置的温度给用户。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户修改目标温度的操作中，哪个对象应首先接收这个请求？

解决方案：Server 作为首先接收 change_target_temp(RoomID, Temperature) 消息的软件对象，主要负责处理来自用户界面的温度调整请求并协调后续的处理流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的温度调整操作。

3. **对象间的关联关系**

- Server 与 Scheduler 建立关联，Scheduler 负责具体的温度调整。
- Scheduler 进一步与 StatisticController 关联，后者负责记录温度更改历史和相关数据。

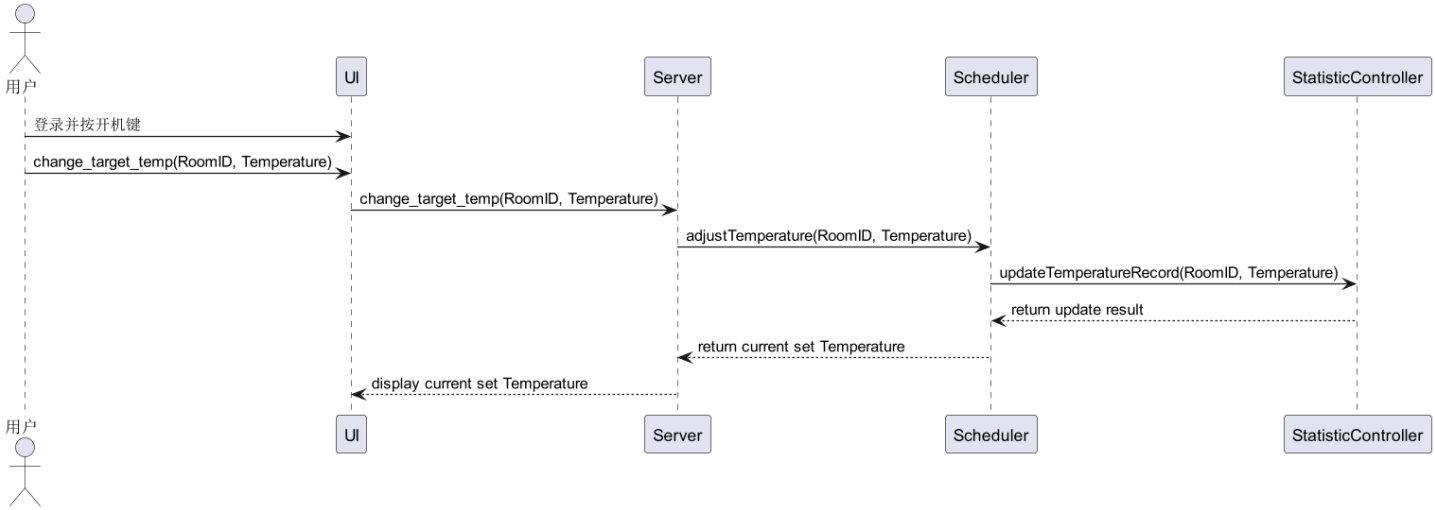
4. **需要修改或初始化的对象属性**

- StatisticController 负责记录温度设置的变更，包括更新时间和新的温度值。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化温度变更记录，确保在系统重启后能够恢复最后设置的温度记录。

6. **Sequence Diagram**：



序列图说明

- 用户登录系统并启动空调，满足前置条件。
- 用户通过用户界面发送修改目标温度的请求。
- Server 接收请求并指示 Scheduler 调整温度。
- Scheduler 请求 StatisticController 更新温度记录。
- StatisticController 更新后将结果返回给 Scheduler。
- Scheduler 将当前设置的温度反馈给 Server。
- Server 将新设定的温度显示在UI上，让用户确认温度已按需调整。

4.1.4 对象设计：InqueryCost(RoomID)

操作契约

- **前置条件**：用户已登录系统。
- **后置条件**：
  a. 用户与数据库建立“关联”。
  b. 用户获取房间的消费信息。
  c. 返回当前状态信息。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户查询费用的操作中，哪个对象应首先接收这个请求？ **解决方案**：Server 作为首先接收 InqueryCost(RoomID) 消息的软件对象，主要负责处理来自用户的查询请求并协调后续的数据获取流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的数据查询操作。

3. **对象间的关联关系**

- Server 与 Scheduler 建立关联，Scheduler 负责实际的数据查询。
- Scheduler 进一步与 StatisticController 关联，后者负责执行数据检索和信息反馈。

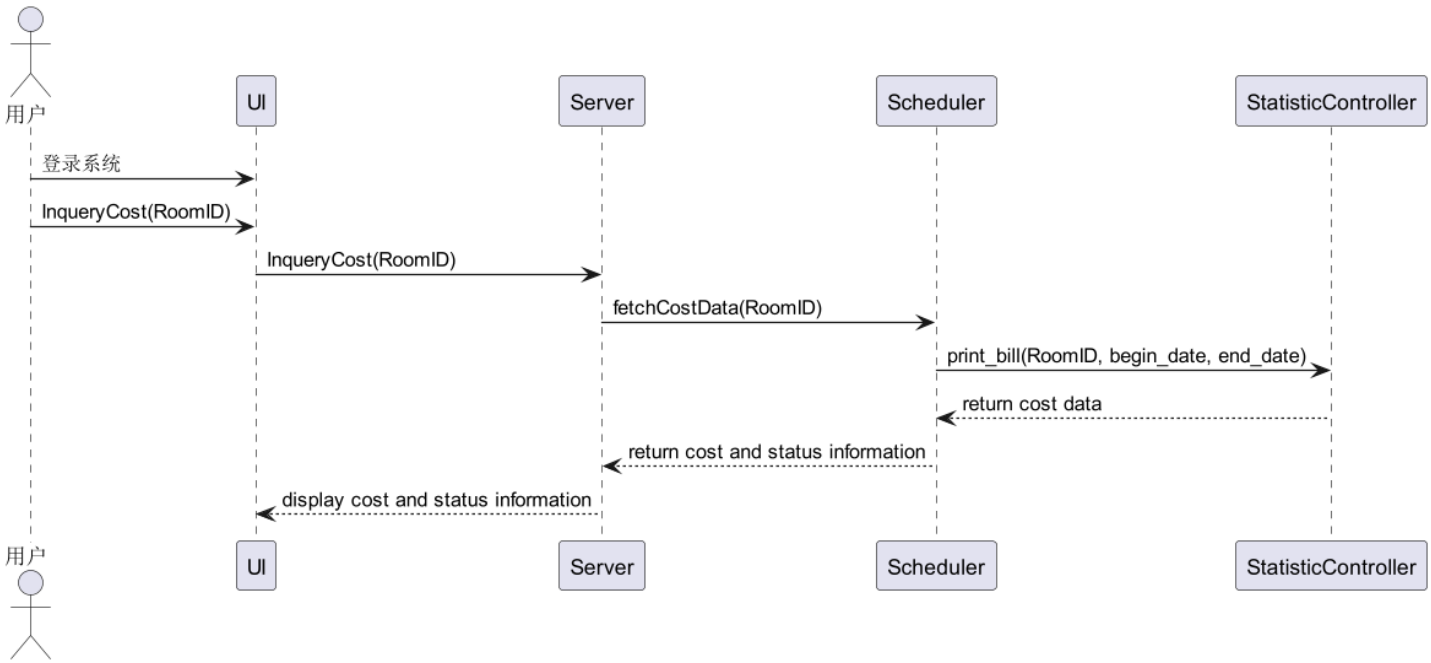
4. **需要修改或初始化的对象属性**

- StatisticController 负责执行 print_bill(room_id, begin_date, end_date) 函数，获取和计算房间的消费信息。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化查询记录和相关状态信息，以便为用户提供历史查询数据。

6. **Sequence Diagram：**



序列图说明

- 用户登录系统，满足前置条件。
- 用户通过用户界面发起查询费用的请求。
- Server 接收请求并指示 Scheduler 进行数据查询。
- Scheduler 请求 StatisticController 执行 print_bill 函数，获取指定房间的消费信息。
- StatisticController 处理请求并将费用数据返回给 Scheduler。
- Scheduler 将费用及当前状态信息反馈给 Server。
- Server 将费用和状态信息显示在UI上，让用户可以查看。

4.1.5 对象设计：set_fan_speed(self, room_id, fan_speed, fee_rate)

操作契约

- **前置条件**：用户已经登录系统，并且已经按下空调的开机键。
- **后置条件**：
  a. 空调的风速设置为用户指定的风速。
  b. 系统返回当前设置的风速给用户。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户设置风速的操作中，哪个对象应首先接收这个请求？ **解决方案**：Server 作为首先接收 set_fan_speed(self, room_id, fan_speed, fee_rate) 消息的软件对象，主要负责处理来自用户的设置请求并协调后续的处理流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的风速调整操作。

3. **对象间的关联关系**

- Server 与 Scheduler 建立关联，Scheduler 负责具体的风速调整。
- Scheduler 进一步与 StatisticController 关联，后者负责记录风速设置的历史和相关费率数据。

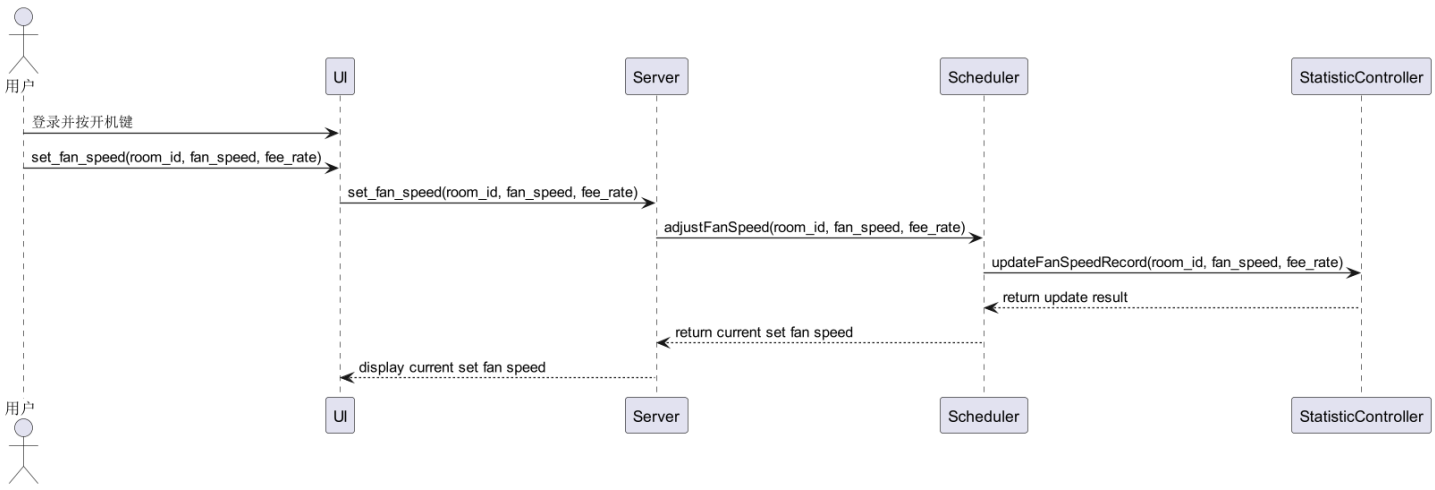
4. **需要修改或初始化的对象属性**

- StatisticController 负责记录风速设置的变更，包括更新时间、新的风速值以及相关的费率。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化风速变更记录和费率信息，确保在系统重启后能够恢复最后设置的风速和费率。

6. **Sequence Diagram**：



序列图说明

- 用户登录系统并启动空调，满足前置条件。
- 用户通过用户界面发送设置风速的请求。
- Server 接收请求并指示 Scheduler 调整风速。
- Scheduler 请求 StatisticController 更新风速记录和费率。
- StatisticController 处理请求并将数据返回给 Scheduler。
- Scheduler 将当前设置的风速和费率信息反馈给 Server。
- Server 将新设定的风速显示在UI上，让用户确认风速已按需调整。

4.1.6 对象设计：TurnOff(RoomID)

操作契约

- **前置条件**：空调已经开启。
- **后置条件**：
  a. 空调停止运行。
  b. 计费停止。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户想要关闭空调的操作中，哪个对象应首先接收这个请求？

解决方案：Server 作为首先接收 request_on 方法发送的 TurnOff(RoomID) 消息的软件对象，主要负责处理来自用户的关闭请求并协调后续的处理流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的空调关闭操作和计费停止处理。

3. **对象间的关联关系**

- Server 与 Scheduler 建立关联，Scheduler 负责具体的空调关闭和停止计费的操作。
- Scheduler 进一步与 StatisticController 关联，后者负责记录关闭空调的服务记录和更新计费状态。

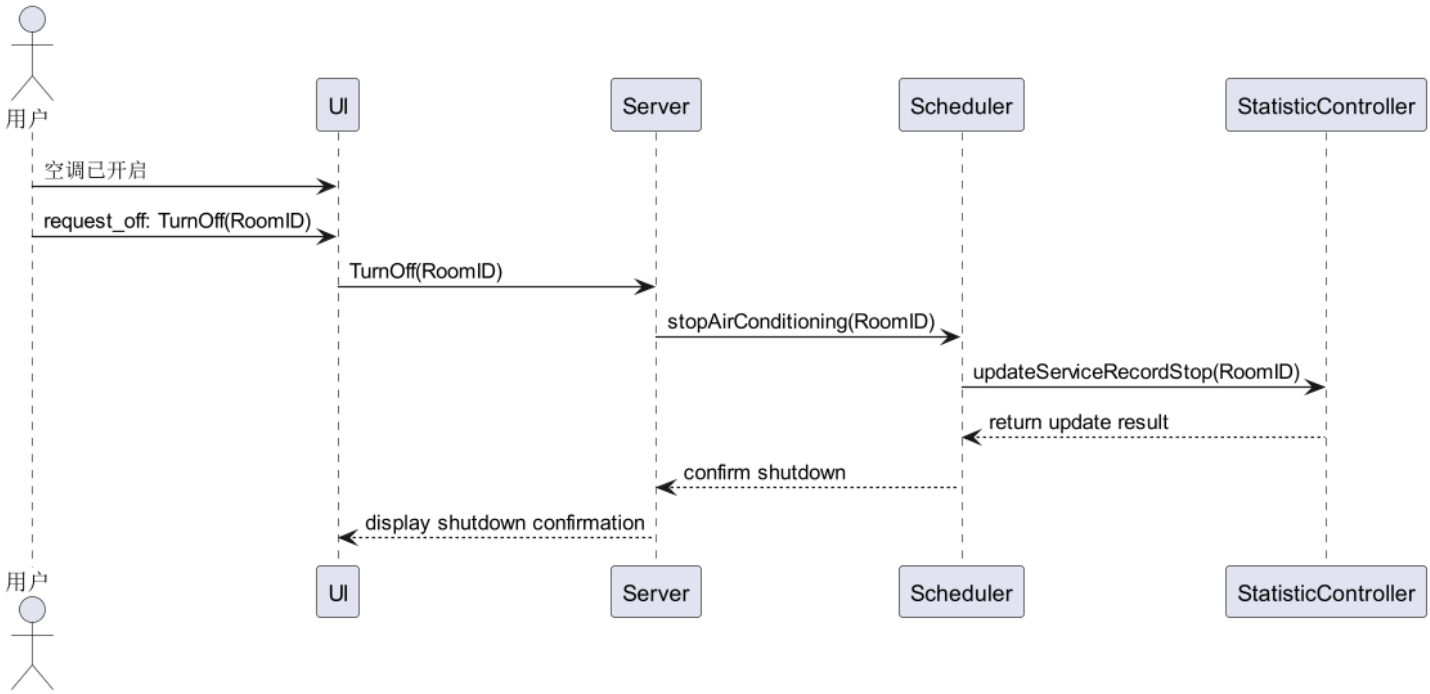
4. **需要修改或初始化的对象属性**

- StatisticController 负责记录空调关闭的事件和停止计费的信息。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化关闭空调和计费停止的记录，确保在系统重启后能够保持正确的计费状态。

6. **Sequence Diagram：**



序列图说明

- 用户确认空调已开启，满足前置条件。
- 用户通过用户界面发送关闭空调的请求（通过 request_on 方法）。
- Server 接收关闭请求并指示 Scheduler 停止空调运行和计费。
- Scheduler 请求 StatisticController 更新服务记录和停止计费。
- StatisticController 处理请求并将更新结果返回给 Scheduler。
- Scheduler 将关闭确认信息反馈给 Server。
- Server 将关闭确认信息显示在UI上，让用户确认空调已成功关闭。

4.2 用例:前台营业员处理表单

4.2.1 已知条件

消息可编程名称	参数列表	消息含义	返回值
GetBill	RoomID,CustomerID	获取账单	BillID，CheckinTime，

			CheckoutTime，TotalCost
PrintBill	BillID	打印账单	OK
GetDetail	RoomID,CustomerID	获取详单	DetailID，List(StartTime，EndTime，Temperature，FanSpeed，CostPerHour，Cost)
PrintDetail draw_report	DetailID room_id type_report year month week	打印详单 打印报告	OK report

4.2.2 对象设计：GetBill (RoomID, CustomerID)

操作契约

- **前置条件**：前台正在处理账单。
- **后置条件**：
  a. 账单与房间建立“关联”。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户请求获取账单的操作中，哪个对象应首先接收这个请求？ **解决方案**：Server 作为首先接收 GetBill(RoomID, CustomerID) 消息的软件对象，主要负责处理来自用户的获取账单请求并协调后续的处理流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的账单生成和关联处理。

3. **对象间的关联关系**

- Server 与 Scheduler 建立关联，Scheduler 负责实际的账单生成和管理房间关联的操作。

- Scheduler 进一步与 StatisticController 关联，后者负责存储和管理账单数据。

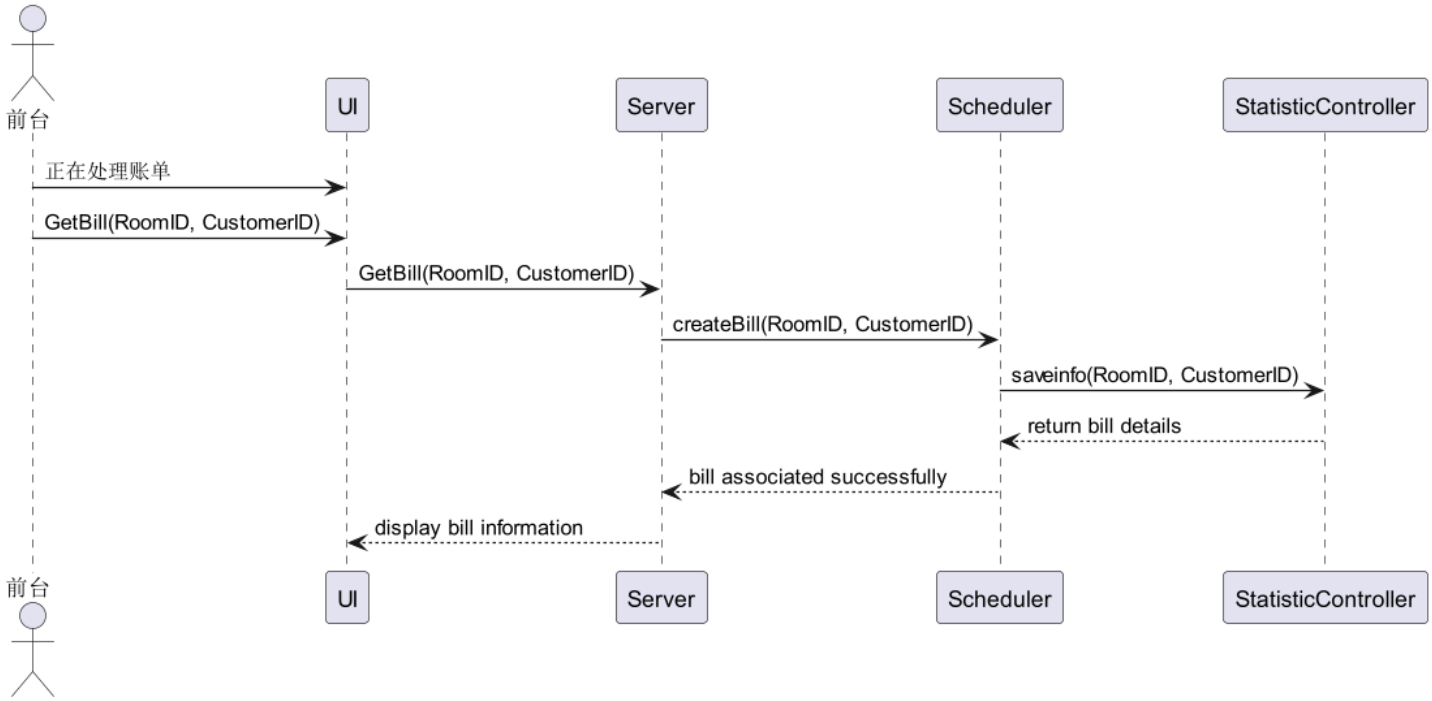
4. **需要修改或初始化的对象属性**

- StatisticController 负责初始化和更新账单信息，确保每个账单都准确反映了与特定房间和顾客ID的关联。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化账单信息，包括账单的生成、房间关联和顾客信息，以便未来查询和审计。

6. **Sequence Diagram：**



序列图说明

- 前台正在处理账单，满足前置条件。
- 前台通过用户界面发送获取账单的请求。
- Server 接收请求并指示 Scheduler 生成账单和管理房间关联。
- Scheduler 请求 StatisticController 关联账单与指定房间和顾客ID。
- StatisticController 处理请求并将账单详情返回给 Scheduler。
- Scheduler 将账单关联成功的确认信息反馈给 Server。
- Server 将账单信息显示在UI上，让前台可以查看并处理账单。

4.2.3 对象设计：PrintBill (RoomID)

操作契约

- **前置条件**：前台完成处理账单。
- **后置条件**：
  a. 账单打印完成，向系统返回状态。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户请求打印账单的操作中，哪个对象应首先接收这个请求？ **解决方案**：Server 作为首先接收 PrintBill(RoomID) 消息的软件对象，主要负责处理来自前台的打印账单请求并协调后续的打印流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的账单打印操作。

3. **对象间的关联关系**

- Server 与 Scheduler 建立关联，Scheduler 负责实际的账单打印任务。
- Scheduler 进一步与 StatisticController 关联，后者负责记录账单打印的操作和状态。

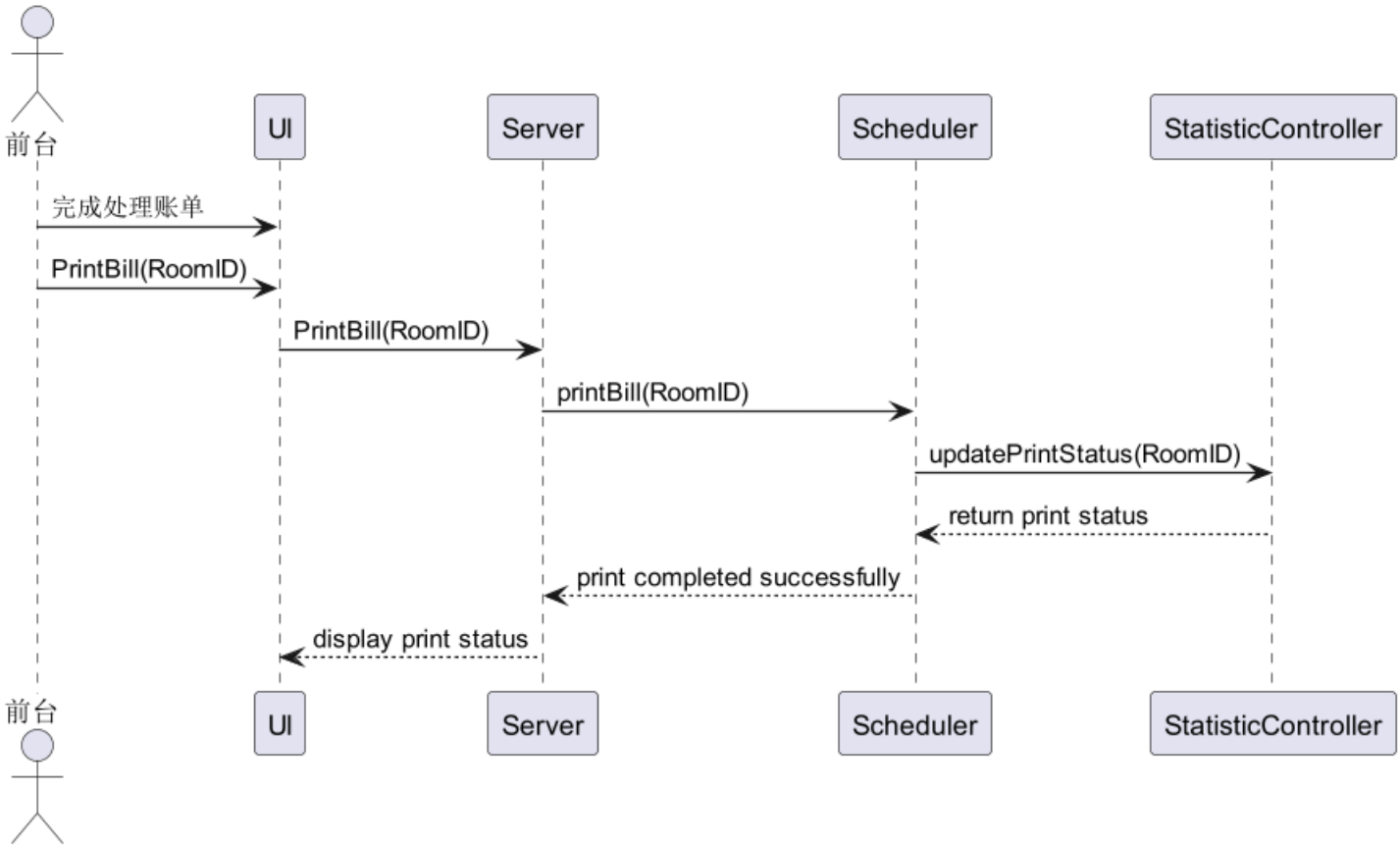
4. **需要修改或初始化的对象属性**

- StatisticController 负责记录账单的打印状态，包括打印完成的时间和结果状态。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化账单打印状态，以确保记录打印操作的完成情况和时间，便于未来的追踪和验证。

6. **Sequence Diagram**：



序列图说明

- 前台完成账单处理，满足前置条件。
- 前台通过用户界面发送打印账单的请求。
- Server 接收请求并指示 Scheduler 进行账单打印。
- Scheduler 请求 StatisticController 更新账单的打印状态。
- StatisticController 处理请求并将打印状态返回给 Scheduler。
- Scheduler 将打印完成的确认信息反馈给 Server。
- Server 将打印状态显示在UI上，让前台确认账单已成功打印。

4.2.4 对象设计：GetDetail (RoomID)

操作契约

- **前置条件**：前台正在处理详单。
- **后置条件**：
  a. 一个新的详单创建。
  b. 详单与房间建立“关联”。
  c. 详单的属性初始化：开始时间、结束时间、空调温度、风速、每小时费用、总费用。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户请求获取详单的操作中，哪个对象应首先接收这个请求？ **解决方案**：Server 作为首先接收 GetDetail(RoomID) 消息的软件对象，主要负责处理来自前台的获取详单请求并协调后续的详单生成流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的详单创建和属性初始化操作。

3. **对象间的关联关系**

- Server 与 Scheduler 建立关联，Scheduler 负责实际的详单创建和属性设置。
- Scheduler 进一步与 StatisticController 关联，后者负责详单数据的存储和管理。

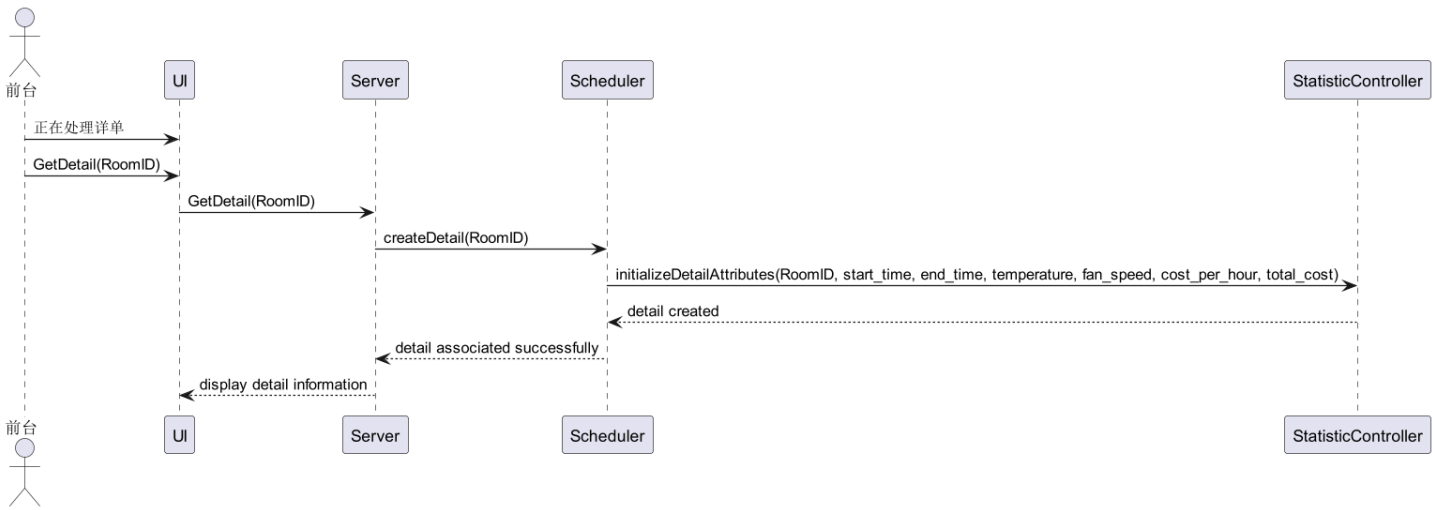
4. **需要修改或初始化的对象属性**

- StatisticController 负责初始化详单的属性，包括开始时间、结束时间、空调温度、风速、每小时费用和总费用。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化详单信息，包括所有初始化的属性，以便未来查询和审核。

6. **Sequence Diagram**：



序列图说明

- 前台正在处理详单，满足前置条件。
- 前台通过用户界面发送获取详单的请求。
- Server 接收请求并指示 Scheduler 进行详单的创建和属性初始化。
- Scheduler 请求 StatisticController 初始化详单属性，并存储详单数据。
- StatisticController 完成详单属性的初始化并返回成功创建的确认信息给 Scheduler。
- Scheduler 将详单关联成功的确认信息反馈给 Server。
- Server 将详单信息显示在UI上，让前台可以查看并进一步处理。

-

4.2.5 对象设计：PrintDetail (DetailID)

操作契约

- **前置条件**：前台完成处理详单。
- **后置条件**：
  a. 详单打印完成，向系统返回状态。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户请求打印详单的操作中，哪个对象应首先接收这个请求？ **解决方案**：Server 作为首先接收 PrintDetail(DetailID) 消息的软件对象，主要负责处理来自前台的打印详单请求并协调后续的打印流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的详单打印操作。

3. **对象间的关联关系**

- Server 与 Scheduler 建立关联，Scheduler 负责实际的详单打印任务。
- Scheduler 进一步与 StatisticController 关联，后者负责记录详单打印的操作和状态。

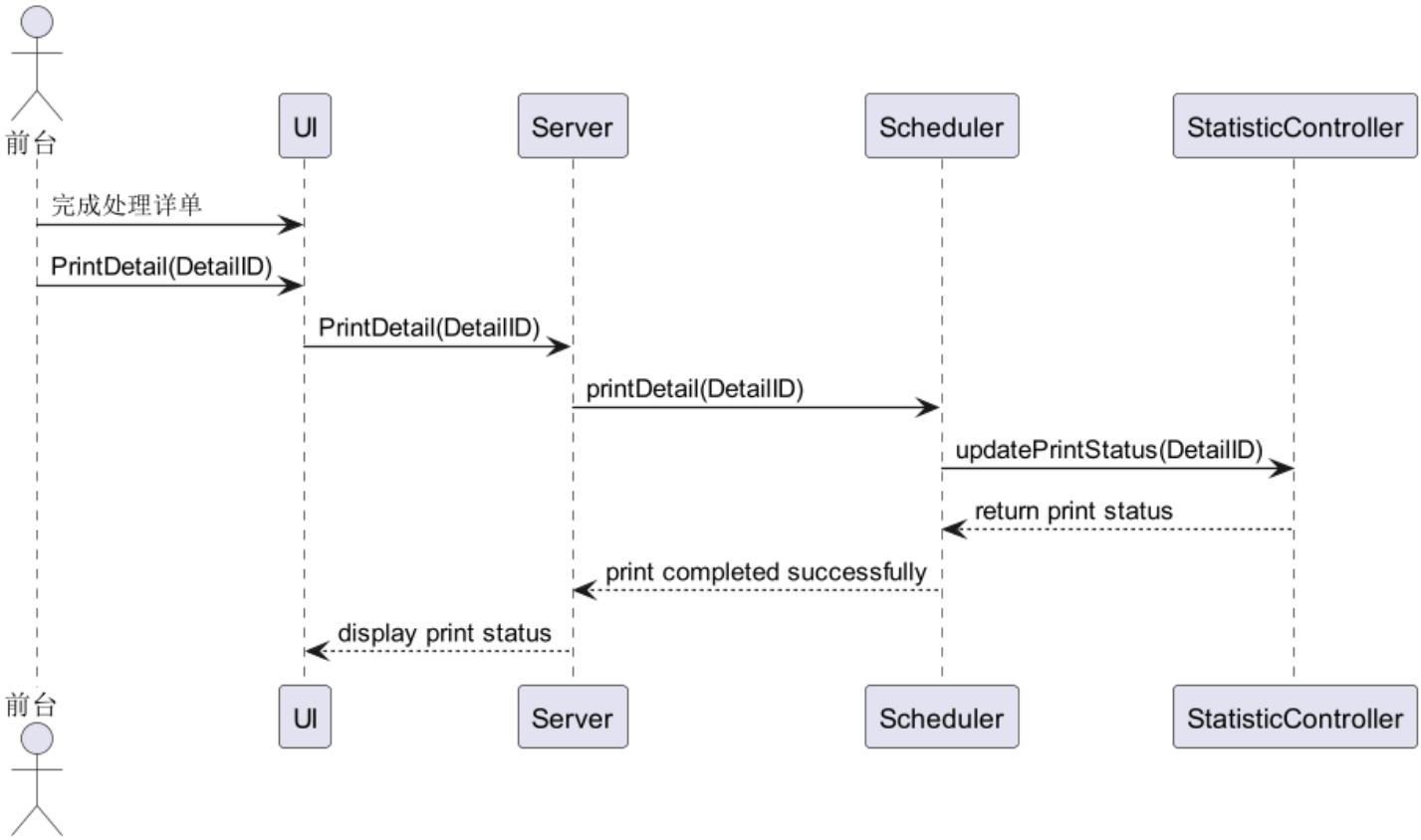
4. **需要修改或初始化的对象属性**

- StatisticController 负责记录详单的打印状态，包括打印完成的时间和结果状态。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化详单打印状态，以确保记录打印操作的完成情况和时间，便于未来的追踪和验证。

6. **Sequence Diagram**：



序列图说明

- 前台完成处理详单，满足前置条件。
- 前台通过用户界面发送打印详单的请求。
- Server 接收请求并指示 Scheduler 进行详单打印。
- Scheduler 请求 StatisticController 更新详单的打印状态。
- StatisticController 处理请求并将打印状态返回给 Scheduler。
- Scheduler 将打印完成的确认信息反馈给 Server。
- Server 将打印状态显示在UI上，让前台确认详单已成功打印。

4.2.6 对象设计：draw_report(room_id, type_report, year, month, week)

操作契约

- **前置条件**：前台正在制作报告。
- **后置条件**：
  a. 返回报告内容。

设计过程

1. **第一个接收该消息的软件对象**

问题：在用户请求绘制报告的操作中，哪个对象应首先接收这个请求？ **解决方案**：Server 作为首先接收 draw_report(room_id, type_report, year, month, week) 消息的软件对象，主要负责处理来自前台的报告制作请求并协调后续的报告生成流程。

2. **负责创建对象实例的对象**

Scheduler 实例由 Server 创建或获取，以便实施具体的报告绘制操作。

3. **对象间的关联关系**

- Server 与 Scheduler 建立关联，Scheduler 负责实际的报告绘制任务。
- Scheduler 进一步与 StatisticController 关联，后者负责管理和提供报告所需的数据。

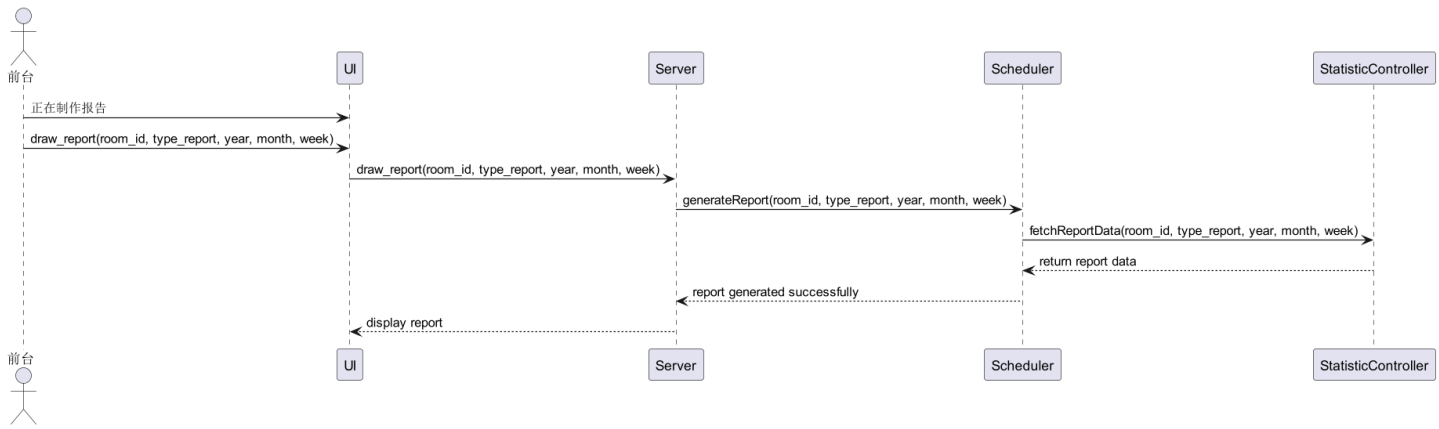
4. **需要修改或初始化的对象属性**

- StatisticController 负责提供报告所需的数据，包括时间范围内的各种统计数据和分析。

5. **数据持久化对象的设计**

StatisticController 可以在数据库中持久化报告数据，以确保在需要时可以重新生成或验证报告。

6. **Sequence Diagram**：



序列图说明

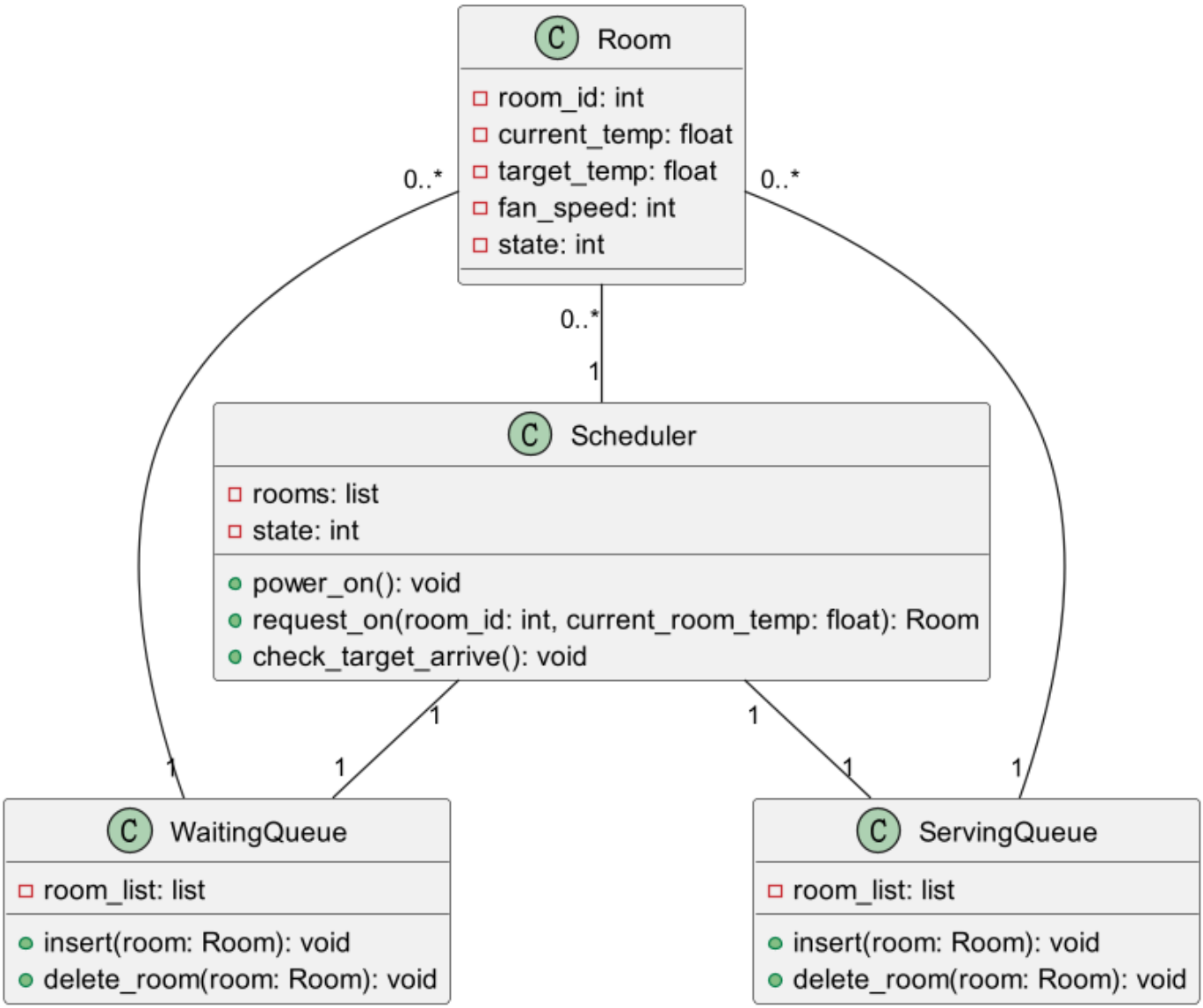
- 前台正在制作报告，满足前置条件。
- 前台通过用户界面发送绘制报告的请求。
- Server 接收请求并指示 Scheduler 进行报告的绘制。
- Scheduler 请求 StatisticController 提供所需的报告数据。
- StatisticController 处理请求并将数据返回给 Scheduler。
- Scheduler 绘制完成后的报告反馈给 Server。
- Server 将绘制好的报告显示在UI上，让前台可以查看并进一步处理。

---

5. 系统静态结构设计

5.1 用例:用户使用空调

软件分层类图:



类的说明:

- **Room介绍:**

属性	类型	描述
room_id	int	房间的唯一标识符
current_temp	float	房间当前的温度
target_temp	float	房间的目标温度
fan_speed	int	房间的风速级别
state	int	房间的当前状态

- **WaitingQueue**

方法	返回类型	描述
insert	void	将房间对象添加到等待队列中
delete_room	void	从等待队列中删除指定的房间对象

- **ServingQueue**

方法	返回类型	描述
insert	void	将房间对象添加到服务队列中
delete_room	void	从服务队列中删除指定的房间对象

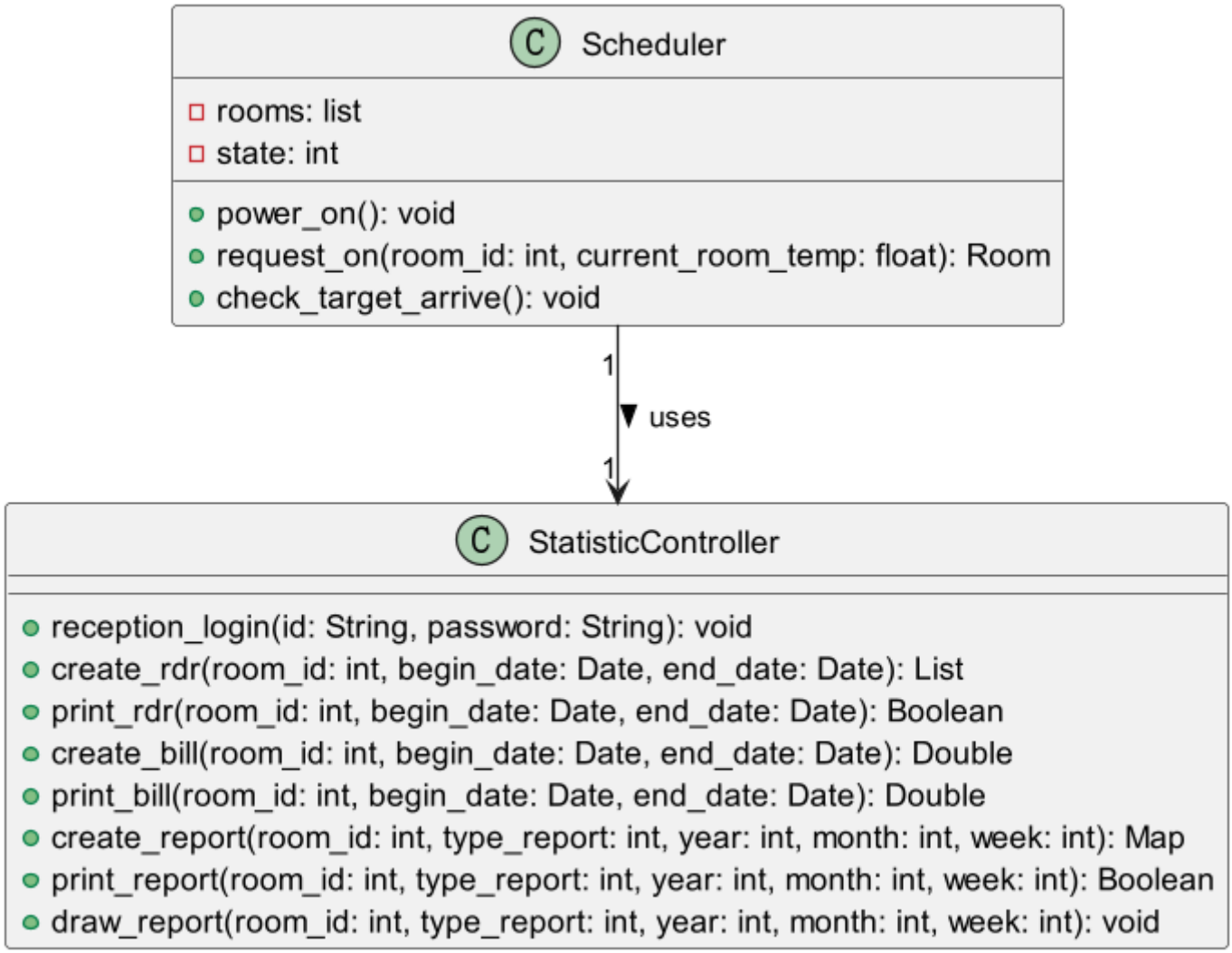
- **Scheduler**

属性	类型	描述
rooms	list	管理所有房间对象的列表
state	int	调度器的当前状态

方法	返回类型	描述
power_on	void	启动调度器并初始化房间对象
request_on	Room	处理房间的开机请求并进行调度
check_target_arrive	void	检查是否有房间达到目标温度并进行处理

5.2 用例:前台处理表单

软件分层类图:



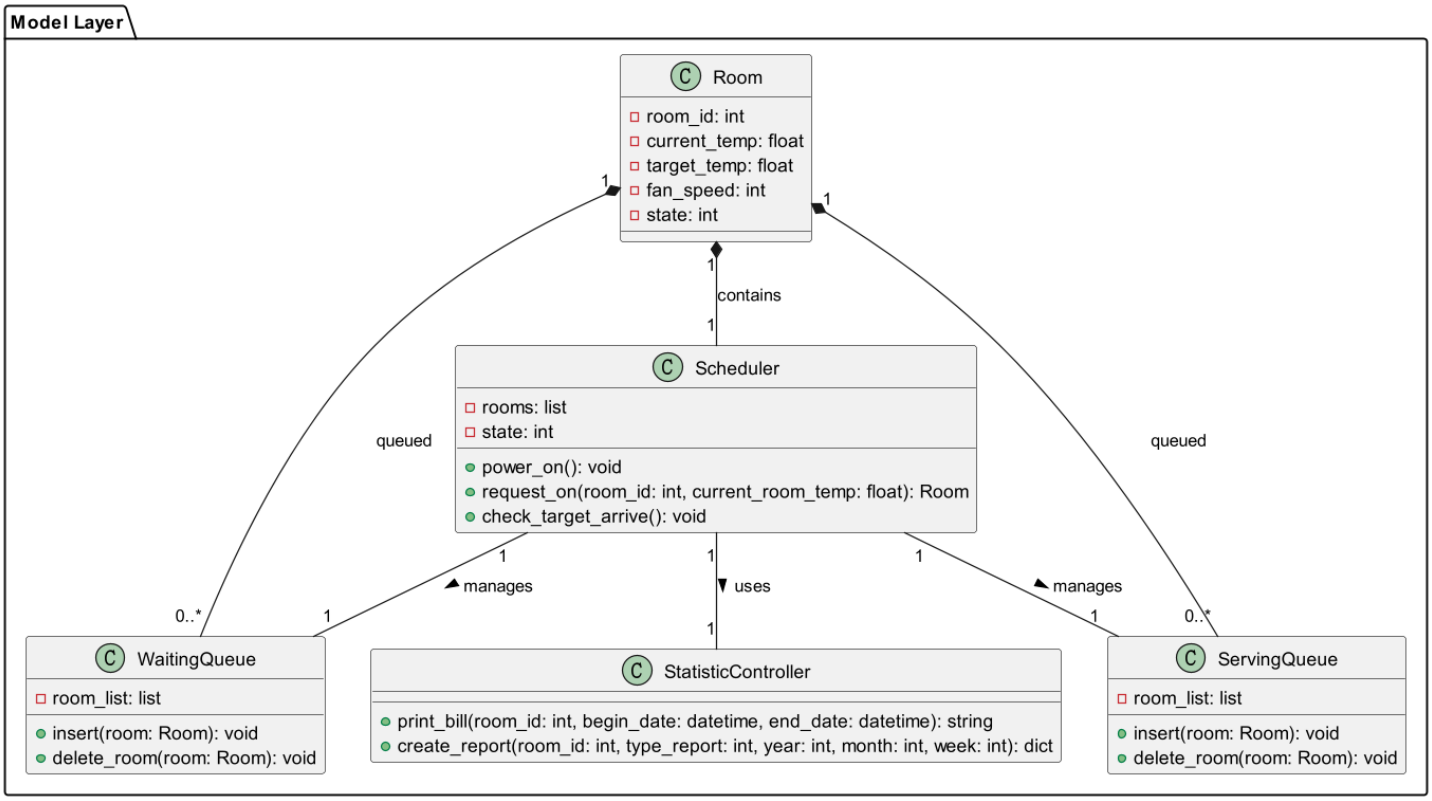
类的说明：

StatisticController 介绍

方法	返回类型	描述
reception_login	void	登录验证功能，通常与用户界面交互
create_rdr	List	根据房间号和日期范围创建详单
print_rdr	Boolean	打印详单，通常包括将详单数据格式化并输出到特定的媒介
create_bill	Double	创建账单，通常涉及计算特定时间段内的费用
print_bill	Double	打印账单，输出费用细节到文件或显示界面
create_report	Map	创建报告，根据房间号和时间生成统计报告
print_report	Boolean	打印报告，将报告输出到文件或显示界面
draw_report	void	绘制报告，可能涉及生成图形表示的报告，如条形图、饼图等

5.3 系统级的静态结构

系统级别类图：



6. 工作量统计

正文：以表格的形式如实给出各个组员的工作内容及工作量描述；

表 4 作业工作内容及工作量统计

		张梓良	杨晨	苗雨	吉奥博	朱馨妍	魏陈正树
界面设计	登录界面					√	
	控制面板界面					√	
	前台营业员界面			√			
	帐单打印界面			√			
	详单打印界面			√			
	空调管理员界面				√		
	房间参数设置界面				√		

	房间状态监控界面				√		
界面控制与前后台交互	reception			√			
	reception_bill			√			
	reception_details			√			
	mon_submit(request)				√		
	monitor(request)				√		
	get_monitor_data(request)				√		
动态结构 用户使用空调	TurnOn					√	√
	change_target_temp	√					√
	Set_fan_Speed	√					√
	InqueryCost		√				√
	TurnOff		√				√
动态结构 前台营业员处理表单	GetBill		√				√
	PrintBill		√				√
	GetDetail		√				√
	PrintDetail		√				√
	draw_report		√				√
静态结构设计	Room	√					√
	Scheduler	√					√
	WaitingQueue	√					√

	ServingQueue	√					√
	StatisticController		√				√
登录界面和控制面板交互	log_in					√	
	get_room_id					√	
	client_on					√	
	client_off					√	