北京郵電大學
Beijing University of Posts and Telecommunications

波普特廉价酒店系统概要设计说明书

基于 UML 的面向对象建模方法

班级_小组：2023219108_108e

组长：张晨旭

组员 1：杨珺涵

组员 2：许世典

组员 3：覃疆楠

组员 4：张睿佳

日期：2025/11/25

目录

1. 软件架构

1.1 软件架构示意图

1.1.1 架构概述

该系统采用了前端 Vue 和后端 Django 的分层架构。这种架构将前端用户交互和后端业务逻辑分离，通过 API 进行数据交互，从而提高了系统的灵活性、可扩展性和可维护性。

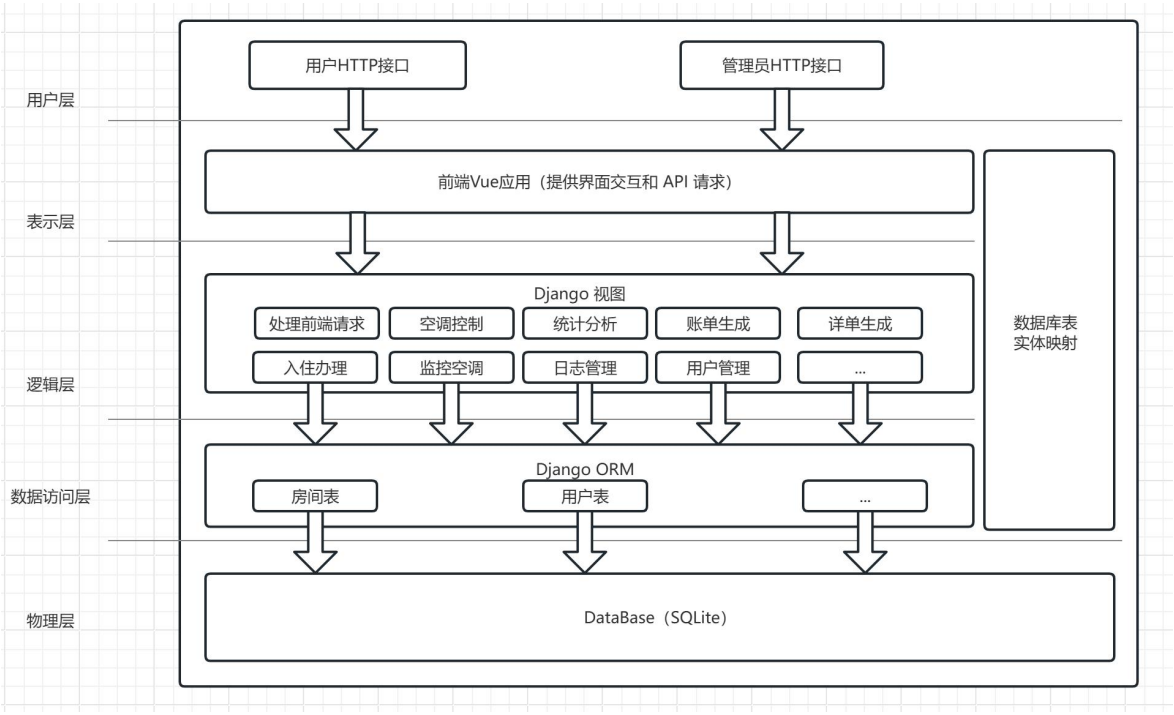

图 1 软件系统架构图

1.1.2 架构模式：前后端分离

该架构采用了前后端分离的设计模式。前端使用 Vue 来负责用户界面和用户交互，而后端使用 Django 来处理业务逻辑和数据存取。前端和后端通过 RESTful API 进行数据交换。

前端（Vue）：负责展示用户界面、接收用户输入、与后端交互。通过 Vue 的组件化开发模式，前端代码易于维护和扩展。Vue 的响应式特性使得用户界面更加流畅和高效。

后端（Django）：负责处理前端发送的请求，执行业务逻辑（如空调控制、账单管理等），并与数据库进行交互。Django 提供了一个强大的 ORM 使得数据库操作更加简洁高效。

1.1.3 工作原理：请求处理流程

以下是系统的工作流程，展示了前端、后端和数据库之间的交互机制：
用户请求：

用户通过 Vue 前端界面发起请求（例如：设置空调温度、查看账单等）。

前端（Vue）：

Vue 通过 HTTP 请求（通常是 AJAX 或 Fetch API）向后端 Django 发送请求。请求类型包括 GET（获取数据）、POST（提交数据）等。

后端（Django）：

后端接收到前端请求后，Django 将请求路由到对应的视图（View），通过 URL 分发器。

视图处理请求并从模型中获取数据。Django 的 ORM 将数据库表映射为 Python 对象，使得数据访问更加简洁。

视图将处理结果返回给前端，通常是 JSON 格式的数据。

数据库（Database）：

Django 使用 ORM 来操作数据库，模型（Model）定义了数据结构和表的关系。每个模型对应数据库中的一个表，ORM 通过 Python 对象来实现对数据的操作。

例如，客户请求获取空调状态时，后端会查询数据库中的空调状态表，并将结果返回给前端。

前端更新：

前端收到后端的响应后，Vue 会根据返回的数据更新用户界面。例如，更新空调的当前温度、风速等信息，或者展示账单详情。

1.1.4 架构的合理性

前后端分离：前端和后端独立开发，互不影响，能够提高开发效率，便于团队协作。前端可以专注于用户体验和界面展示，后端专注于业务逻辑和数据处理。

Vue 的响应式特性：Vue 提供的双向数据绑定和虚拟 DOM 技术，使得前端界面能够在数据变化时自动更新，提高了用户体验。

Django 的模块化和 ORM：Django 提供了内置的 ORM，可以通过 Python 类与数据库表进行交互，避免了手写 SQL 的复杂性。同时，Django 的模块化设计使得功能扩展变得简便。

API 驱动的交互：前端与后端通过 RESTful API 进行交互，数据交换使用标准化的格式（如 JSON），使得系统更加灵活，可以适配不同的客户端（例如移动端和桌面端）。

1.1.5 架构的运行机制

架构的工作机制可以通过以下流程描述：

请求发起：用户通过 Vue 前端发起 HTTP 请求。

请求转发：请求通过 URL 分发器转发到 Django 的视图。

业务处理：Django 视图执行相应的业务逻辑（如空调控制、账单生成等），并通过 ORM 操作数据库。

数据返回：Django 返回处理结果给前端。

界面更新：Vue 根据后端返回的数据更新前端界面，展示最新的空调状态或账单详情等。

1.1.6 总结

这种架构的最大优点是前后端分离和模块化设计，使得开发和维护更加高效。前端和后端各自专注于不同的任务，互不干扰，同时通过标准化的 API 接口进行数据交互。通过 Django 提供的强大功能（如 ORM、视图、表单处理等）和 Vue 的灵活性，使得系统能够高效处理用户请求，提供优秀的用户体验，并且易于扩展和维护。

1.2 分层结构说明

在本软件架构中，系统采用了典型的三层架构（前端、后端、数据库），并通过分层设计进行功能的模块化与解耦。根据这一结构，我们可以明确每个层次中负责不同任务的对象和组件。以下是分层架构的具体说明：

1.2.1 接收前端请求的对象

前端(Vue）：

前端的 Vue.js 组件负责与用户交互，并通过 HTTP 请求将请求发送至后端。前端通过组件化设计，使得每个页面或功能模块对应一个 Vue 组件。

具体责任：接收用户输入，如设置空调温度、查看账单等，并将这些请求转化为 HTTP 请求发往后端。

后端(Django）：

URL 分发器：后端的 URL 分发器负责接收前端发来的请求，解析请求中的 URL，并将其映射到对应的视图函数或类。

具体责任：根据请求的 URL，将请求转发给相应的视图对象进行处理。

1.2.2 创建对象实例的对象

后端(Django）：

视图：Django 的视图层负责处理前端请求，并根据请求创建相应的业务对象。例如，在用户请求获取空调状态时，后端视图会根据请求创建一个空调管理对象来查询数据。

模型（Model）：Django 的模型负责定义数据库中的数据结构。视图会通过模型创建对象实例，从数据库中读取数据或更新数据。

具体责任：根据用户的请求，视图层通过模型层实例化对象来获取或操作数据（如实例化 Room 或 ACStatus 模型来获取空调状态）。

1.2.3 处理请求具体要求的对象

后端 (Django）：

视图：视图是处理具体业务逻辑的核心对象。当前端发出请求时，视图负责执行相关的业务操作，例如根据请求设置空调的温度、生成账单等。

具体责任：处理前端请求的具体要求，调用模型层进行数据操作，处理如温度调节、账单计算等业务逻辑，并生成响应返回给前端。

1.2.4 数据同步和存储的对象

后端（Django）：

模型：Django 的模型层是数据存储的核心，它通过 Django ORM 将 Python 对象映射到数据库表。当请求需要存取数据时，模型对象负责与数据库进行交互（如读取、更新、删除）。

具体责任：通过 ORM 操作数据库，负责数据的同步和持久化存储。例如，空调状态、用户账单等数据通过模型对象进行管理和存储。

数据库：数据库是所有数据的最终存储位置，负责存储所有系统中的持久化数据。

具体责任：负责持久化数据的存储，如用户信息、空调状态、账单信息等。数据库通过 SQL 查询与 Django 的 ORM 层进行交互，提供数据存储和同步功能。

1.2.5 总结

在这个分层架构中，系统的责任被清晰地划分为四个主要部分：

1.前端接收请求：Vue.js 负责接收用户输入，并通过 HTTP 请求与后端通信。

2. 对象实例化：Django 的视图和模型共同负责创建对象实例，根据请求初始化数据对象。

3. 请求处理：视图负责处理具体的业务请求，调用模型进行数据操作。

4. 数据存储与同步：Django 的模型通过 ORM 与数据库进行交互，负责数据的同步和持久化存储。

这样的分层设计保证了系统的高内聚和低耦合，易于维护和扩展，同时支持前后端独立开发和快速迭代。

2. 系统的界面设计

2.1 空调的控制面板设计

图 2 控制面板界面设计图

2.2 前台营业员办理入住和结账界面设计

图 3 办理入住界面设计图

图 4 办理结账界面设计图

2.3 监控空调运行状态界面设计

图 5 监控空调运行界面设计图

3. 系统动态结构设计

3.1 用例:UC_使用空调

3.1.1 已知条件

表 1 使用空调用例说明表

系统事件	返回	操作契约	备注
PowerOn(RoomId,CurrentRoomTemp)	Return(Mode,TargetTemp,CurrentFee,TotalFee)	1、调度对象与房间建立关联； 2、一个服务对象（空调对象实例）被创建（当前服务对象数小于服务对象数上限，验收环境的服务对象上限数=3；否则不分配）； 3、服务对象与房间建立关联； 4、计时器对象被创建（可以作为服务对象的时间戳属性）； 5、详单对象被创建（用于记录服务期间产生的服务信息）； 6、服务队列的信息被修改； 7、详单对象的属性：服务开始时间，模式，目标温度，费率及费用值等被赋值；	1、调度对象需要知道发送请求来自于哪个房间； 2、调度对象需要分配一个服务对象，等于服务对象（空调）需要被实例化； 3、服务对象需要知道被服务的房间信息； 4、服务对象为了能够提供服务还需要准备必要的对象：计时器和详单（空白的记录本）； 5、调度对象对于接收的请求信息需要记录到服务队列或者等待队列中； 6、被服务房间对应的详单对象需要由服务对象来初始化。
ChangeTemp(RoomId,TargetTemp)	Return(isOK)	1、调度对象与房间建立关联； 2、如果该请求在服务队列，则调度对象与服务对象建立关联（调度对象将请求转发给服务对象）； 3、如果该请求在等待队列，则调度对象修改等待队列中的目标温度属性 TargetTemp;	1、注意，该请求在人工环境中不需要发给调度对象（顾客和服务员面对面）；但在系统开发的软件设计时该请求必须由作为控制器对象的调度对象接收再转发给服务对象；
ChangeSpeed(RoomId,FanSpeed)	Return(isOK)	1、调度对象与房间建立关联； 2、如果该请求在服务队列，则调度对象与服务对象建立关联（调度对象将请求转发给服务对象）； 3、详单对象的属性被赋值； 4、如果该请求在等待队列，则调度对象修改等待队列中的风速的属性 FanSpeed;	2、这个对象的关联表示调度对象将接收到的请求转发给服务对象；同时需要更改服务队列或者等待队列中的风速值用于进行调度。 3、表示服务对象会产生一条详单数据
PowerOff(RoomId)	Return(State,CurrentFee,T	1、调度对象与房间删除关联； 2、服务对象与房间删除管理；	1、表示调度对象将服务队列或等待队列中的房间信息清除；

	otalFee)	3、详单对象的属性被赋值;	2、表示服务对象结束服务并释放资源; 3、表示服务对象结束服务时会产生一条详单数据;
RequestState(RoomId)	Return(CurrentFee,TotalFee)	1、服务对象的属性值发生改变;	1、表示此时此刻服务对象查看了当前的费用，并告知顾客; 2、系统开发时这个功能将变为由服务对象定时刷新并主动发给前端;

3.1.2 对象设计：PowerOn(RoomId,CurrentRoomTemp)

1.消息的操作契约：

a.调度对象与房间建立关联;

b.一个服务对象（空调对象实例）被创建（当前服务对象数小于服务对象数上限，验收环境的服务对象上限数=3；否则不分配）；

c.服务对象与房间建立关联;

d.计时器对象被创建（可以作为服务对象的时间戳属性）；

e.详单对象被创建（用于记录服务期间产生的服务信息）；

f.服务队列的信息被修改;

g.详单对象的属性：服务开始时间，模式，目标温度，费率及费用值等被赋值;

2.**第一个接收该消息的对象是系统调度对象 Scheduler。**调度对象负责接收所有房间的请求、判断是否需要调度并将请求转发给服务对象，同时维护服务队列与等待队列。在系统启动时调度对象已被实例化为单例或集中控制器，监听来自房间客户端的请求入口。

3.创建对象实例：

操作契约 b：系统创建服务对象;

操作契约 d：服务对象创建计时器对象，作为服务对象的时间戳属性;

操作契约 e,g：服务对象创建详单对象，作为服务对象的属性;

操作契约 f：调度对象创建服务队列和等待队列，并初始化设置属性。

4.关联关系建立：

房间客户端启动系统，调度对象 Scheduler 与 RoomId 关联。

Scheduler 校验房间并查找当前服务对象数是否到上限：

若是，Scheduler 创建 AirConditionerService 实例。AirConditionerService 与 RoomId。

AirConditionerService 创建 Timer 并关联。

AirConditionerService 创建 DetailRecord 并关联，同时将 DetailRecord 传送给 PersistenceManager。

Scheduler 创建 ServiceQueue 和 WaitQueue 并关联。

5.对象属性初始化或修改：

服务对象创建并初始化 Timer，DetailRecord 属性;

ServiceQueue 和 WaitingQueue 被创建并初始化。

6.数据持久化：

使用 PersistenceManager 将 DetailRecord 写入数据库。

保存相关持久化字段：RoomId，startTime，mode，targetTemp，feeRate

7.该消息对应的 sequence diagram

服务资源无限制情况：

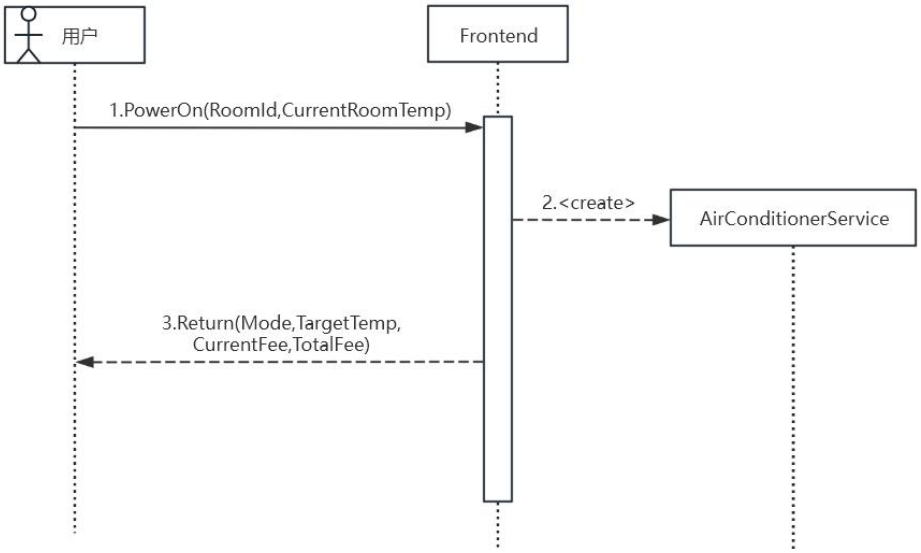

图 6 资源无限制时序图

服务资源有限制情况：

有限制情况下需要执行调度策略，首先给出调度前的准备工作的时序图：

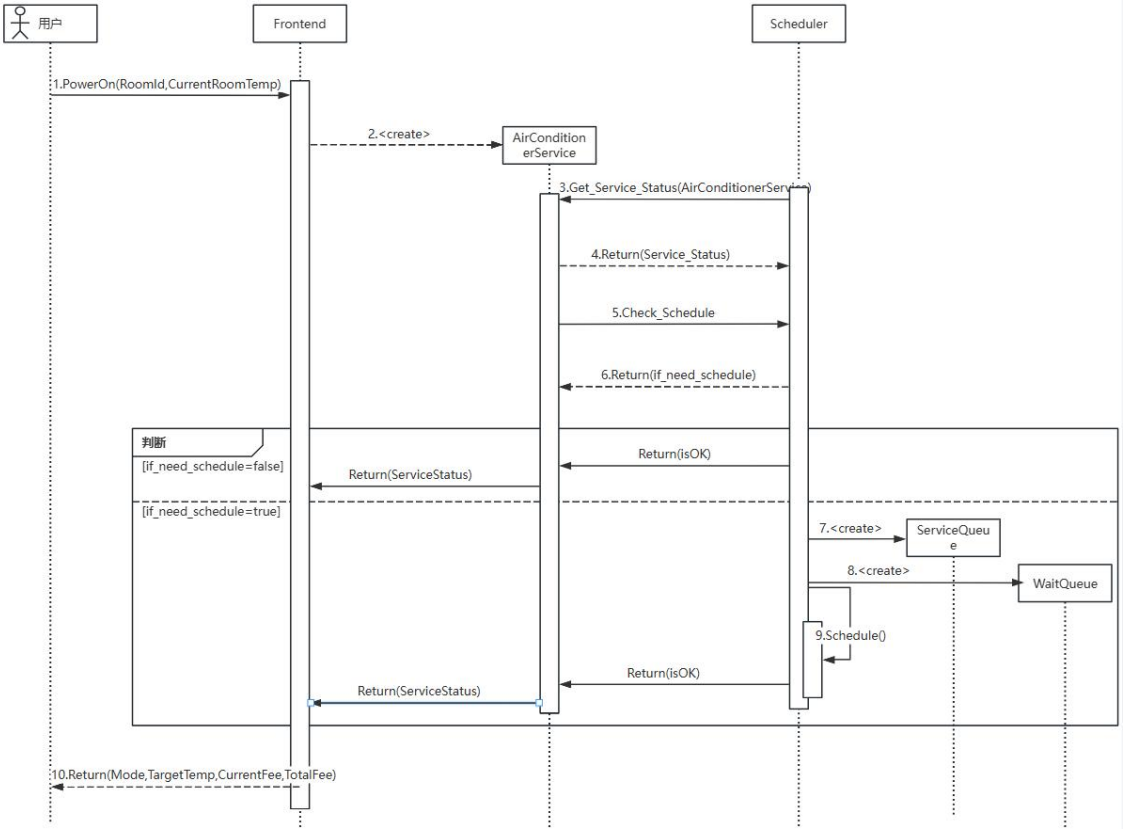

图 7 调度前准备工作时序图

根据题目要求，我们需要实现的基本调度策略为优先级调度+时间片调度。

调度时首先考虑优先级策略，优先级调度的依据是请求的风速；当所有请求的风速相同时，再进行时间片调度。

优先级调度的思路如下：

如果判断=大于，则启动优先级调度策略，再判断有几个服务对象的风速低于请求风速：

如果只有一个，则该房间被放置于等待队列，并被分配一个等待服务时长；服务对象被释放并被分配给新的请求对象；

如果有多个服务对象的风速相等且低于请求对象，则服务时长最大的服务对象被释放并分配给新的请求对象，该房间被放置于等待队列，且分配一个等待服务时长；

如果多个服务对象的风速低于请求风速，且风速不相等，则将风速最低的服务对象释放，该房间被放置于等待队列，且分配一个等待服务时长。

由于调度过程为调度对象的内部活动，故在时序图中不再体现用户对象，体现在创建服务队列和等待队列后的操作。

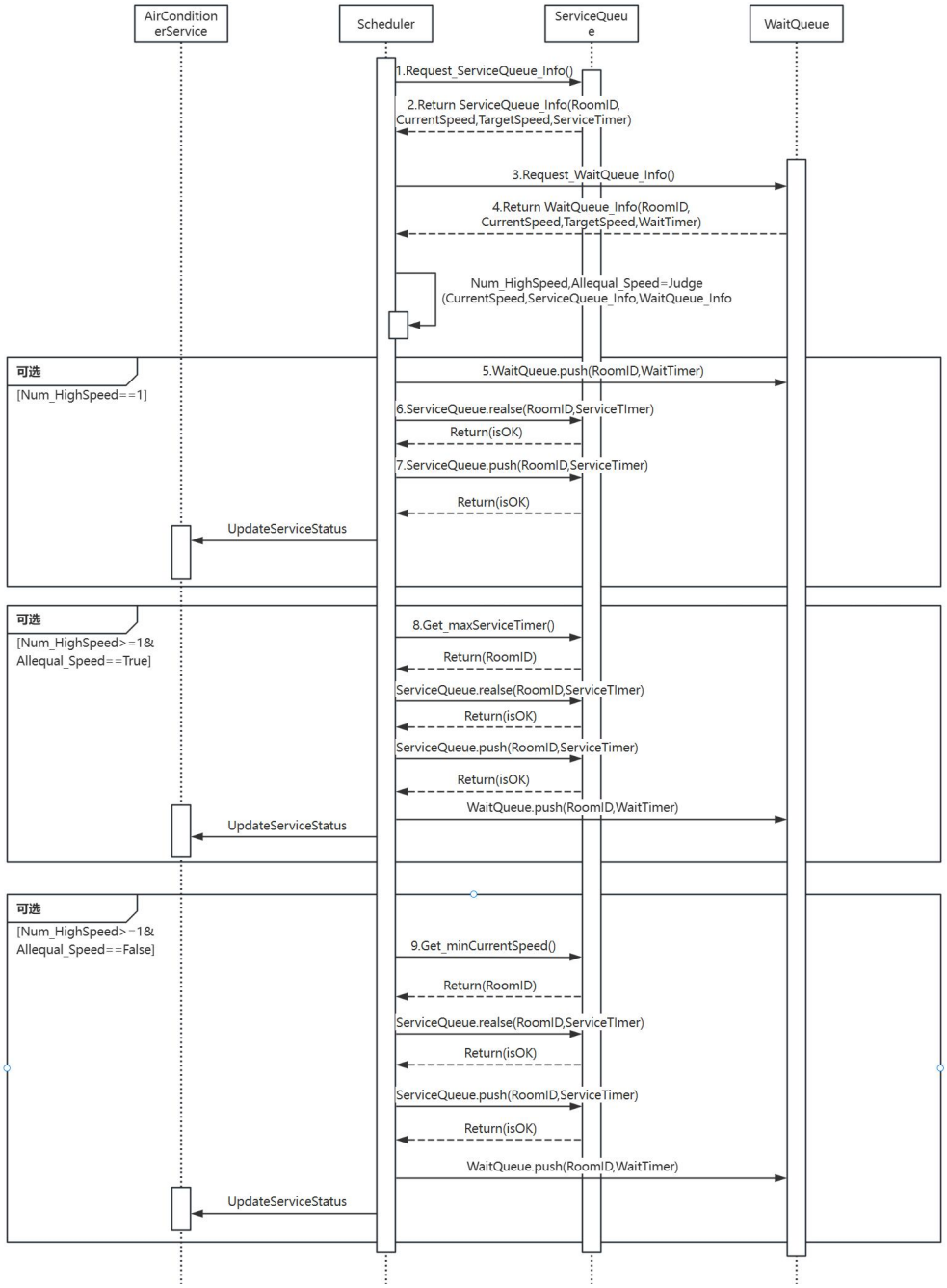

图 8 优先级调度时序图

当优先级相同时，优先级调度会出现无法判断的情况，此时采用时间片调度继续进行调度。时间片调度思路如下：

如果判断=相等，则启动时间片调度策略

将请求对象放置于等待队列，并分配一个等待服务时长；

如果在这两分钟期间，如果没有任何服务状态的变化，当等待服务时长=0时将服务队列中服务时长最大的服务对象释放，该房间被放置于等待队列，且被分配一个等待服务时长；等待服务对象被分配一个服务对象开始服务；

如果在这等待的两分钟期间内，如果有任何一个服务对象的目标温度到达或关机（意味着服务对象释放）则等待队列中的等待服务时长最小的对象获得服务对象；

如果判断=小于，则请求对象必须等到某个服务对象空闲后才能得到服务。

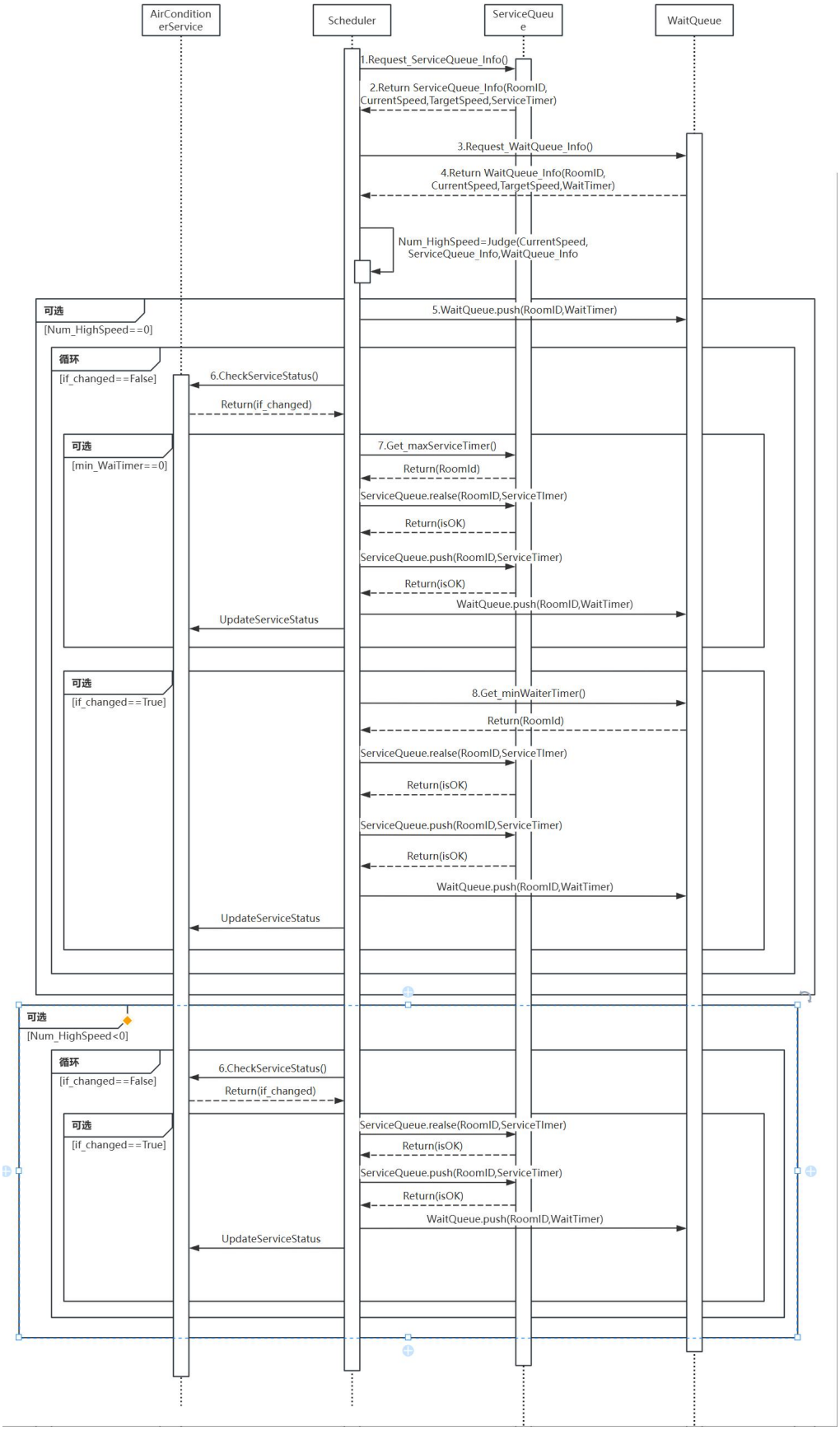

图 9 时间片调度时序图

3.1.3 对象设计：ChangeTemp(RoomId,TargetTemp)

1.消息的操作契约：

a.调度对象与房间建立关联；

b.如果该请求在服务队列，则调度对象与服务对象建立关联（调度对象将请求转发给服务对象）；

c.如果该请求在等待队列，则调度对象修改等待队列中的目标温度属性TargetTemp；

2. 关联关系建立：

调度对象 Scheduler 与 RoomId 关联。

Scheduler 校验房间并查找当前请求是否在服务队列：

若是，Scheduler 与 AirConditionerService 相关联。

3. 对象属性初始化或修改：

服务对象修改 Temp 属性。

详单对象添加记录。

4.该消息对应的 sequence diagram

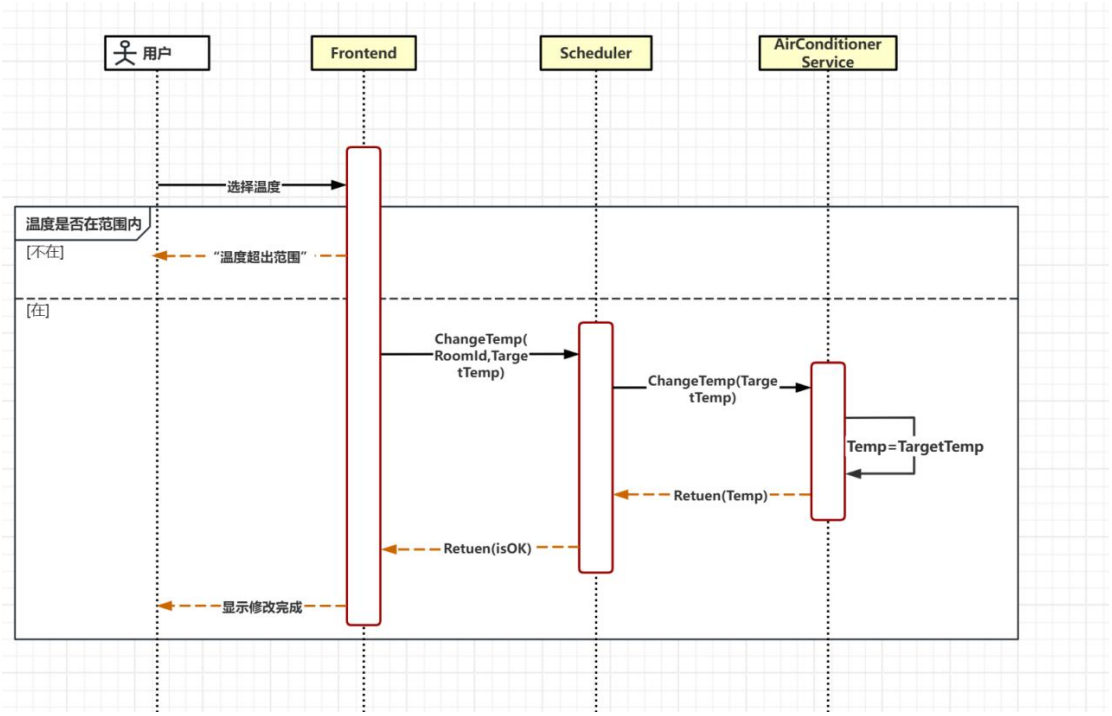

图 10 ChangeTemp 时序图

3.1.4 对象设计：ChangeSpeed(RoomId,FanSpeed)

1.消息的操作契约：

a.调度对象与房间建立关联；

b.如果该请求在服务队列，则调度对象与服务对象建立关联（调度对象将请求转发给服务对象）；

c.详单对象的属性被赋值；

d.如果该请求在等待队列，则调度对象修改等待队列中的风速的属性FanSpeed;

2. 关联关系建立：

调度对象 Scheduler 与 RoomId 关联。

Scheduler 校验房间并查找当前请求是否在服务队列：若是，Scheduler 与 AirConditionerService 相关联。

3. 对象属性初始化或修改：

服务对象修改 FanSpeed 属性。

数据库中修改详单信息。

4.该消息对应的 sequence diagram

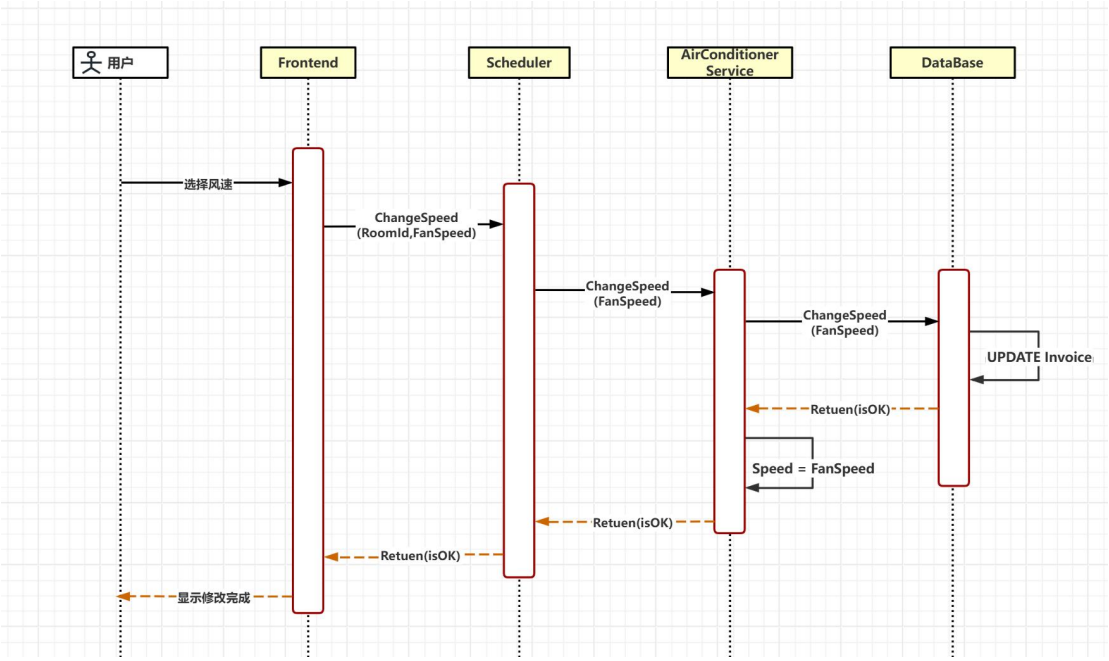

图 11 ChangeSpeed 时序图

3.1.5 对象设计：PowerOff(RoomId)

1.消息的操作契约：

a.调度对象与房间删除关联；

b.服务对象与房间删除管理；

c.详单对象的属性被赋值

2.第一个接收该消息的对象是系统调度对象 Scheduler。Scheduler 负责捕获来自 RoomClient 的退出服务请求；Scheduler 负责判断房间是否处于服务队列或等待队列；Scheduler 决定是否需要释放服务对象、更新系统资源状态；全局控制器，在系统启动时即被实例化并持续监听来自房间的请求。

3.创建或销毁对象实例：

Scheduler 根据 RoomId 找到服务对象 ID，删除服务队列和等待队列中的记录。

AirConditionerService 停止 Timer，完成并冻结 DetailRecord。

PersistenceManager 将 DetailRecord 持久化保存至数据库

4.关联关系解除：

Scheduler 在 ServiceQueue 和 WaitQueue 中删除 RoomId 的记录，解除关联；

AirConditionerService 解除 RoomId 关联，记录终止时间戳，然后销毁 Timer，保存 DetailRecord。

5.需要修改的属性：

在 Scheduler 中修改 ServiceQueue，WaitQueue 删去 RoomId；

AirConditionerService 中修改 ServiceStatus 为 STOP，计算 totalFee，保存至 DetailRecord。

6.数据持久化：

使用 PersistenceManager 将 DetailRecord 写入数据库。

保存相关持久化字段：RoomId，endTime，mode，totalFee

7.该消息对应的 sequence diagram

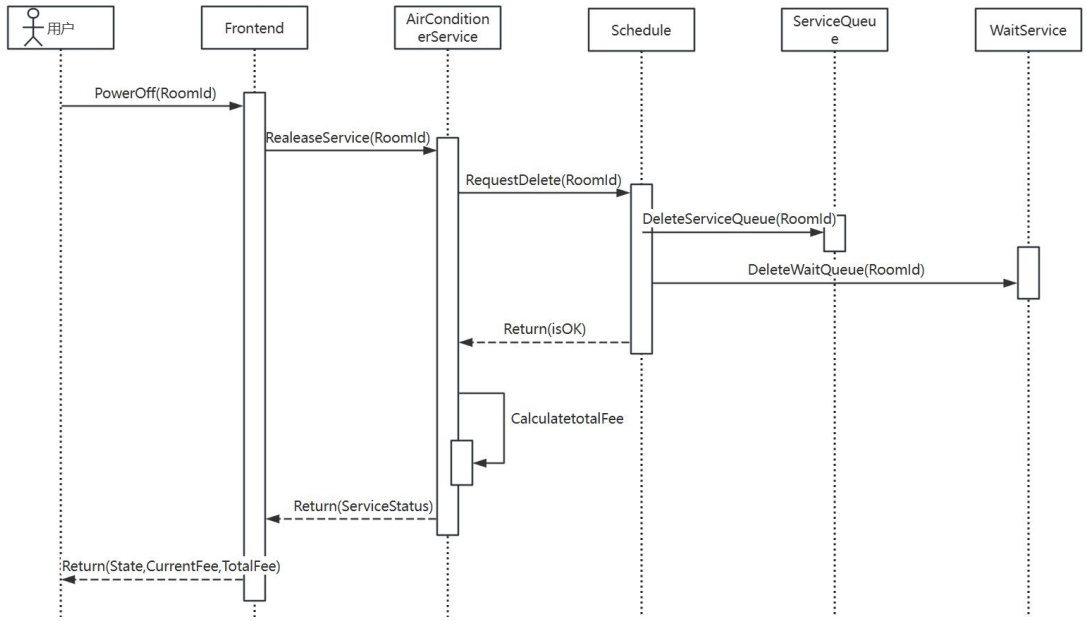

图 12 PowerOff 时序图

3.2 用例：UC_办理入住

3.2.1 已知条件

表 2 办理入住用例说明表

系统事件	返回	操作契约	备注
Registe_CustomerInfo(Cust_Id, Cust_name, number, date)	Return(isOK)	1、顾客对象被创建； 2、顾客的属性被赋值；	首先创建一个顾客对象实例，然后给顾客对象的属性赋值；
Check_RoomState(date)	Return(list_RoomState)	1、前台对象与客房对象建立关联； 2、客房的状态属性被赋值；	前台对象通过客房 Id 查询客房的状态
Create_Accomodation_Order(Customer_id,Room_id)	Return(isOK)	1、住宿订单对象被创建； 2、顾客与客房建立关联； 3、客房的信息及状态属性	创建一个住宿订单，同时修改客房对象有关顾客的信息，表示当前

		被修改; 4、住宿订单的属性被赋值;	客房的状态以及被哪个顾客占用

3.2.2 对象设计：Create_Accomodation_Order(Customer_id,Room_id)

1. 消息中的操作契约

a.住宿订单对象被创建;

b. 顾客与客房建立关联;

c. 客房的信息及状态属性被修改;

d. 住宿订单的属性被赋值;

2. 第一个接收消息的软件对象

该消息由前台营业员发出，AC_System 第一个接收。AC_System 作为后端控制器，负责协调入住业务流程，处理订单创建请求。

3. 创建对象实例

住宿订单对象由 AC_System 创建，用于记录顾客的入住信息、房间关联及押金状态。该对象是整个住宿业务的核心实体，贯穿入住到结账的全生命周期。

4. 关联关系建立

AC_System 通过调用顾客仓库和客房仓库接口，查询并获取已注册的顾客对象和已核验的客房对象。建立三者之间的关联关系：住宿订单对象关联顾客对象（记录入住人信息）和客房对象（锁定目标房间），形成完整的订单业务实体。

5. 对象属性的修改/初始化

a. 住宿订单对象的基础属性（订单号、顾客 ID、房间 ID、创建时间、订单状态）被初始化;

b. 客房对象的状态属性从"空闲待清理"修改为"已入住"，并设置当前占用顾客 ID;

c. 顾客对象的关联房间属性被赋值，标识当前入住的房间;

d. 住宿订单的押金状态初始化为"待缴纳",住宿天数默认为 0(退房时结算)。

6. 数据持久化设计

住宿订单对象创建完成后，通过 Django ORM 调用 order_repository.save()方法，将订单数据写入 SQLite 数据库的 accommodation_order 表。客房状态更新通过 room_repository.update_status()方法同步至 rooms 表，确保数据一致性。

7. 该消息对应的 sequence diagram

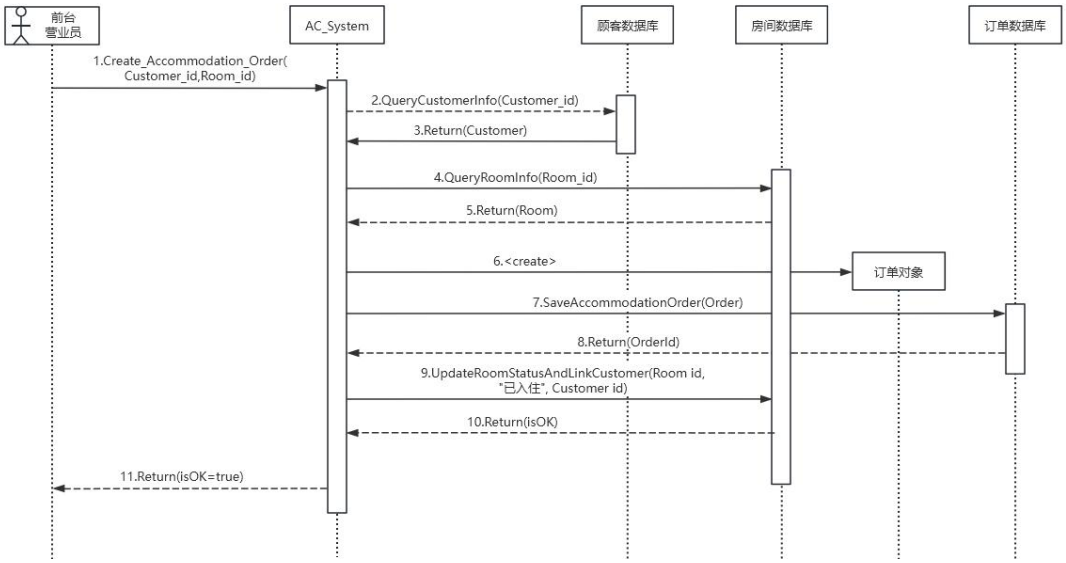

图 13 Create_Accomodation_Order 时序图

前台营业员发起创建住宿订单请求，AC_System 接收并验证参数完整性；系统分别查询顾客和客房实体，确保业务合法性（顾客存在、房间可用）；创建住宿订单对象并初始化属性，通过订单实体建立顾客与客房的业务关联；订单数据持久化后，同步更新客房状态为"已入住"并锁定顾客信息；最后向前台返回操作成功响应，完成入住登记流程。

3.3 用例：UC_办理结账

3.3.1 已知条件

表 3 办理结账用例说明表

系统事件	返回	操作契约
Process_CheckOut(Room_id)	Return(isOK)	1、前台与客房对象建立关联; 2、客房的状态信息等属性被修改;
query_FeeRecords(RoomId, date_out)	Return(days_of_Accommodation, Detail_Records_of_AC)	1、客房对象的住宿天数属性被赋值; 2、与空调详单对象建立关联;
calculate_Accommodation_Fee（days_of_Accommodation，Fee_of_day）	Return(Total_Fee_of_Accommodation)	1、住宿费账单对象被创建; 2、住宿费账单对象的费用属性被赋值
calculate_AC_Fee(list_of_Detail_Records)	Return(Total_Fee_of_AC)	1、空调使用费账单被创建; 2、空调使用费账单对象的费用属性被赋值;
Create_Accommodation_Bill(Room_Id,date)	Return(isOK)	1、住宿费账单对象的其他属性值被修改;
Create_AC_Bill(Room_Id,date)	Return(isOK)	1、空调使用费账单对象被创建; 2、空调使用费账单对象与详单对

		象建立关联; 3、账单对象的属性值被修改;
Create_DetailRecord_AC(RoomId,date_in,date_out)	Return((list_DR_AC(RoomId,RequestTime, RequestDuration,FanSpeed,FeeRate,Fee))	1、账单对象与详单对象关联; 2、详单对象的所有属性被赋值;
Set_RoomState(RoomId)	Return(RoomState=idle)	1、客房对象的状态属性被修改;

3.3.2 对象设计：Create_Accommodation_Bill(Room_Id,date)

1. 消息的操作契约

a.住宿费账单对象的其他属性值被修改，例如：账单时间、房间号、支付状态等。

2. 第一个接收消息的软件对象

该消息由前台营业员在前台界面触发，AC_system 第一个接收。

3. 创建对象实例

住宿费账单对象由 AC_system 创建，用于封装该房间的住宿费用信息，作为最终账务记录的一部分被持久化保存。

4. 关联关系建立

建立账单与客房对象之间的关联（账单属于哪一间房）。

5. 对象属性的修改/初始化

a. 房间号由输入 Room_id 赋值

b. 入住时间从 Accomodation_Order 中获取，退房时间即为当天账单生成的日期

c. 住宿费总价调用 Calculate_Accomodation_Fee()获取

6. 数据持久化设计

住宿费账单对象创建后，通过接口写入酒店账务数据库储存，实现数据持久化

7. 该消息对应的 sequence diagram

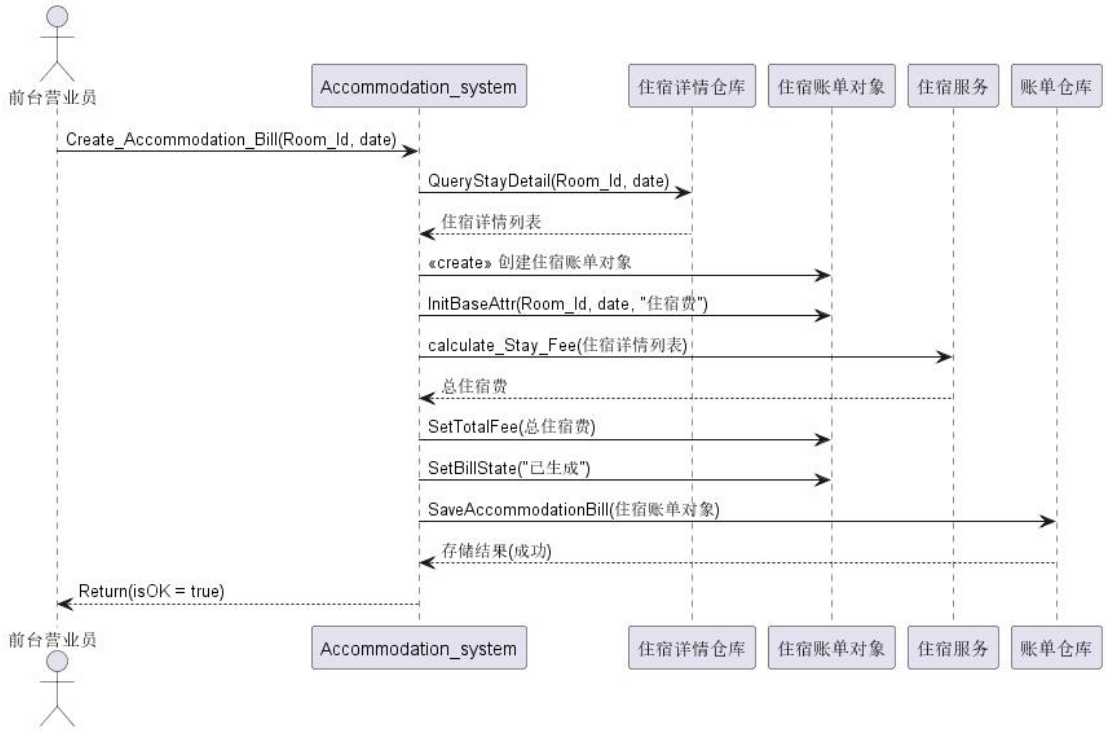

图 14 Create_Accomodation_Bill 时序图

3.3.3 对象设计：Create_AC_Bill(Room_Id,date)

1. 消息中的操作契约

a. 空调使用费账单对象被创建；

b. 空调使用费账单对象与详单对象建立关联；

c. 账单对象的属性值被修改；

2. 第一个接收消息的软件对象

根据系统架构，前台营业员发出的请求需由后端 AC_system 作为第一层接收对象，负责协调后续业务逻辑处理。该消息由前台营业员发出，AC_system 第一个接收。

3. 创建对象实例

空调使用费账单对象由 AC_system 创建，用于存储空调使用相关的费用明细及汇总信息。

4. 关联关系建立

AC_system 调用接口，查询该 Room_Id 在指定 date 范围内的空调详单数据，建立空调使用费账单对象与详单对象的关联关系，确保费用计算有数据支撑。

5. 对象属性的修改/初始化

a. 空调使用费账单对象的基础属性（房间号、账单日期、账单类型）被初始化；

b. 根据关联的详单数据（请求时长、风速、费率等）计算总空调使用费，赋值给账单对象的费用属性；

c. 账单对象的状态属性被修改。

6. 数据持久化设计

空调使用费账单对象创建完成后，通过接口将账单数据写入 hotel.db 数据库的 AC_Bill 表，实现数据持久化。

7. 该消息对应的 sequence diagram

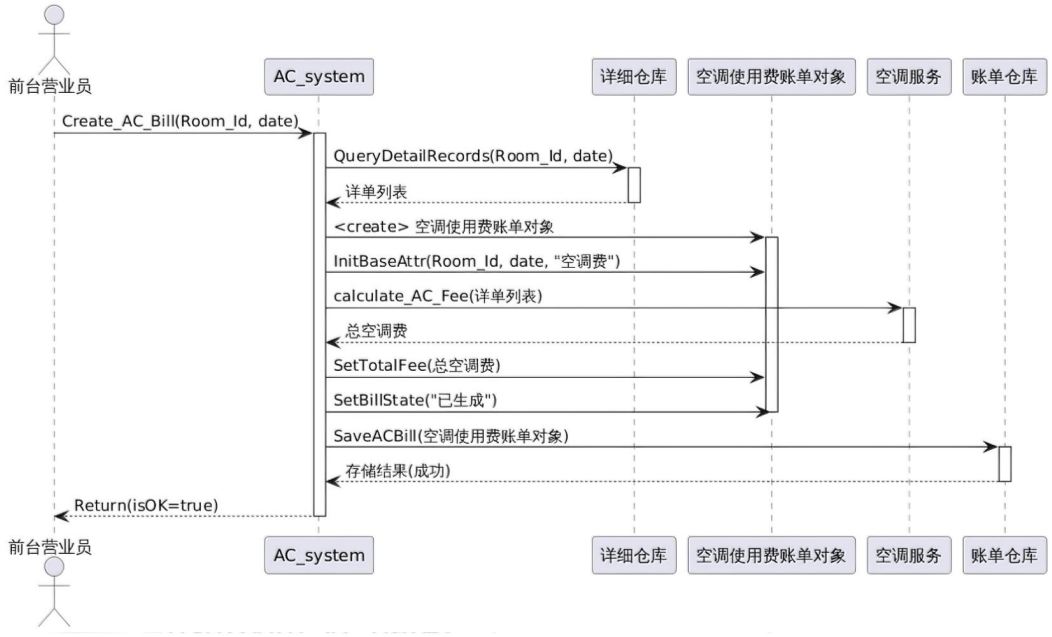

图 15 Create_AC_Bill 时序图

说明：

前台营业员发起创建空调费账单请求，携带房间号和日期参数；

AC_system 作为请求接收对象，先查询该房间指定日期内的所有空调详单记录；

AC_system 创建空调使用费账单对象并初始化基础属性；

调用 ACService 的费用计算方法，基于详单数据计算总空调费；

将计算结果赋值给账单对象，修改账单状态，将账单数据持久化到数据库；

最后 AC_system 向前台营业员返回创建成功的响应，完成整个流程。

3.3.4 对象设计：Create_DetailRecord_AC(RoomId,date_in,date_out)

1. 消息中的操作契约
   a. 账单对象与详单对象关联；
   b. 详单对象的所有属性被赋值；
2. 第一个接收消息的软件对象

   问题：前端发起 Create_DetaiRecords_AC 请求后，系统中哪个软件对象应作为首个接收者，确保请求参数的精准接收与业务逻辑的有序调度？

   解决方案：设计前台服务控制器（FrontDeskController）作为第一个接收消息的软件对象。该对象是前端与后台业务逻辑的桥梁，直接承接前台营业员发起的空调详单生成请求，持有核心参数，能够快速调度后续操作，符合作业 1“前台处理结账相关详单生成”的业务场景定位。
3. 创建对象实例

   由 AC_System 创建 AC_DetailRecord（空调使用详单对象），用于存储单条空调使用的明细数据。

4. 关联关系建立

AC_System 调用接口，通过 Room_Id 关联对应的房间对象，确保详单与房间的归属关系。

5. 对象属性的修改/初始化

a. 初始化基础属性：Room_Id（入参 Room_Id）、RequestTime（入参 date_in）、RequestDuration（通过 date_out - date_in 计算）；

b. 调用 ACService 的 get_FanSpeed 方法，获取该房间空调的当前风速，赋值给 FanSpeed 属性；

c. 调用 ACService 的 calculate_FeeRate 方法，基于 FanSpeed 和预设规则计算 FeeRate（费率）；

d. 计算 Fee 属性：RequestDuration × FeeRate；

e. 设置详单状态为 “已完成”。

6. 数据持久化设计

AC_System 调用接口，将 AC_DetailRecord 对象数据写入 AC_Detail 表，完成持久化。

7. 该消息对应的 sequence diagram

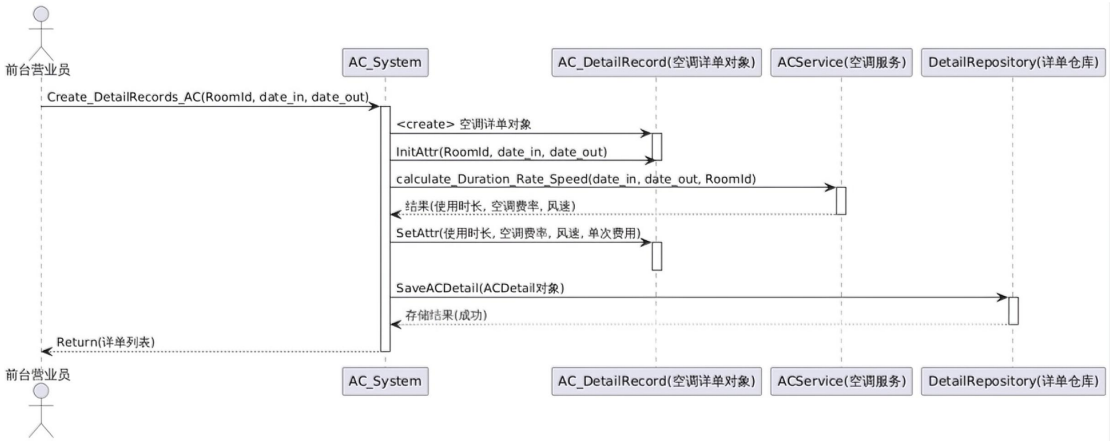

图 16 Create_DetailRecord_AC 时序图

说明

前台发起创建空调详单请求，携带房间号、使用起止时间；

AC_System 先关联对应房间对象，确保数据归属；

创建详单对象并初始化基础属性，通过 ACService 计算时长、获取风速、计算费率；

完成详单属性赋值与状态设置后，持久化数据；

向前台返回包含完整明细的详单列表，完成流程。

4. 系统静态结构设计

4.1 用例:UC_使用空调

1.

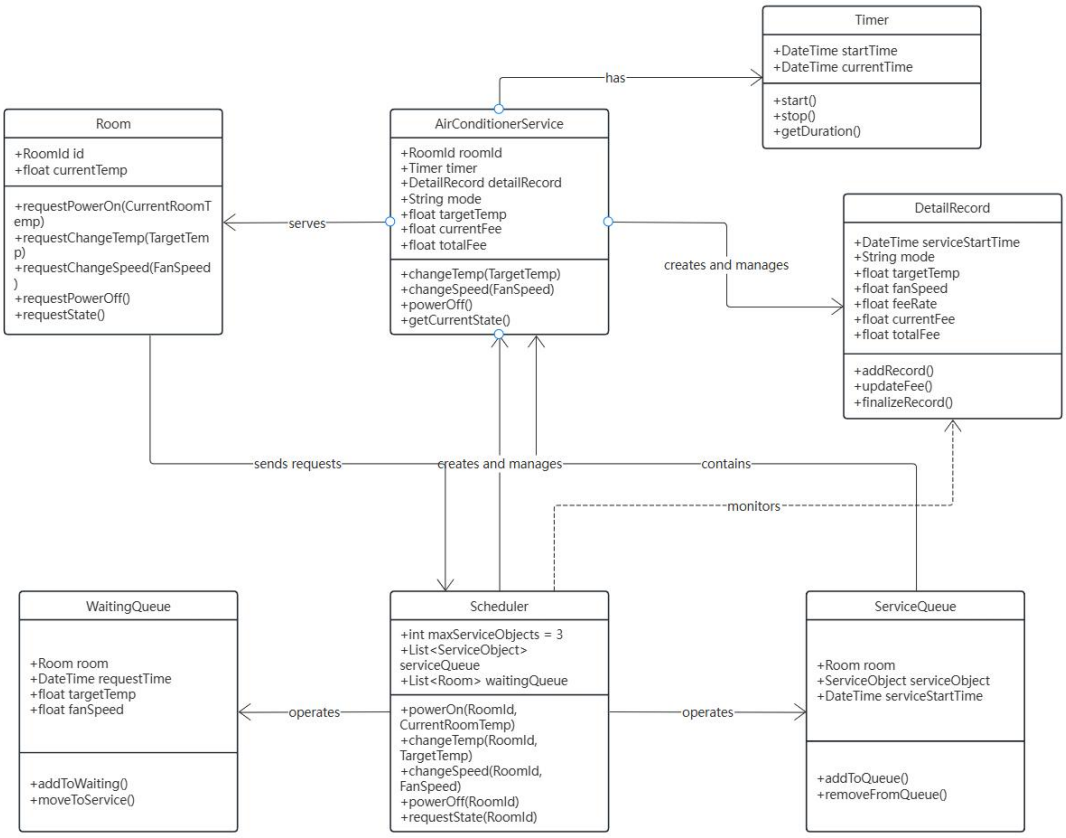

图 17 使用空调类图

2.属性方法说明

类_1:AirConditionerService

类型	名称	描述说明
属性	RoomId	用户入住房间的 Id
属性	Timer	记录使用时间的时间戳
属性	DetailRecord	详细账单信息
属性	mode	空调模式
属性	TargetTemp	空调设定目标温度
属性	currentFee	当前空调费率
属性	totalFee	使用空调总费用
方法	ChangeTemp	接收调温请求，转发给服务层，返回操作结果
方法	ChangeSpeed	接收调风请求，触发调度策略，返回新风速状态

类_2：Timer

类型	名称	描述说明
属性	startTime	计时开始时刻

属性	currentTime	当前时刻
方法	Start	将 startTime 置为当前系统时间，并重置 currentTime
方法	Stop	用当前系统时间刷新 currentTime，返回累计时长
方法	getDuration	计算并返回 startTime→currentTime 的持续时间

类_3:DetailRecord

类型	名称	描述说明
属性	currentFee	本段服务已产生费用
属性	totalFee	本订单累计总费用
属性	mode	工作模式：制冷/制热
属性	targetTemp	本段目标温度
属性	fanSpeed	本段风速（低/中/高）
属性	feeRate	本段费率
方法	addRecord	新建一条详单记录，初始化各字段
方法	updateFee	根据时长×feeRate 刷新 currentFee 与 totalFee
方法	finalizeRecord	服务结束或换风速时落库，返回本段费用

类_4:WaitQueue

类型	名称	描述说明
属性	List<Room>	存放等待服务的 Room 对象列表
方法	addToWaiting	将房间加入队尾，记录 requestTime
方法	moveToService	弹出队首 Room，返回给 Scheduler 进入 ServiceQueue

类_5：ServiceQueue

类型	名称	描述说明
属性	maxServiceObjects=3	最大并发服务对象数
属性	List<ServiceObject>	当前正在服务的对象列表
方法	addToQueue	若未达上限，将加入列表并记录 serviceStartTime
方法	removeFromQueue	从列表移除，释放一个服务槽位

类_6:Room

类型	名称	描述说明
方法	requestPowerOn	接收开机请求，创建 Timer&DetailRecord，调用 Scheduler.powerOn
方法	requestChangeTemp	接收调温，转发 Scheduler.changeTemp()，返回操作结果
方法	requestChangeSpeed	接收调风，转发 Scheduler.changeSpeed()，返回新风速状态
方法	requestPowerOff	接收关机，调用 Scheduler.powerOff()并返回总费用
方法	requestState	接收状态查询，调用 Scheduler.requestState()返回实时数据

类_7：Schedule

类型	名称	描述说明
属性	maxServiceObjects=3	最大并发服务对象数

属性	List<ServiceObject>	当前正在服务的对象列表
属性	List<Room>	存放等待服务的 Room 对象列表
方法	PowerOn	开机请求入口：若服务队列未满则直接分配；否则加入 WaitQueue
方法	ChangeTemp	调温请求：找到对应 ServiceObject，更新目标温度并重新计算优先级
方法	ChangeSpeed	调风请求：触发优先级/时间片调度，必要时抢占或轮转
方法	PowerOff	关机请求：从 ServiceQueue 移除，释放槽位；WaitQueue 队首补位
方法	requestState	查询入口：返回房间当前温度、风速、费用等实时状态

4.2 用例：UC_办理入住

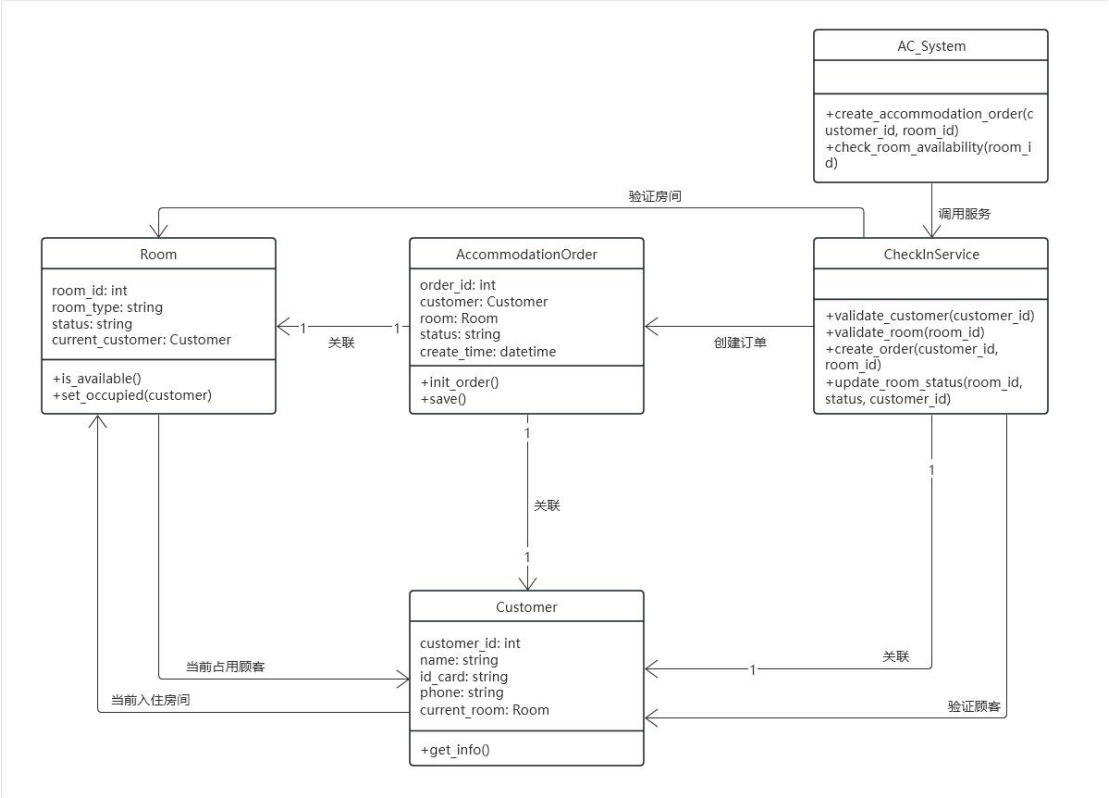

图 18 办理入住类图

类_1:AC_System

类型	名称	描述说明
方法	create_accommodation_order	接收前端请求，完成订单创建
方法	check_room_availability	查询指定房间是否可入住

类_2:CheckInService

类型	名称	描述说明
方法	validate_customer	验证顾客信息是否存在且未入住，返回顾客实

		体或错误
方法	validate_room	验证房间是否存在且状态为"空闲"，返回房间实体或错误
方法	create_order	实例化 AccommodationOrder 对象，初始化属性并建立与顾客、房间的关联
方法	update_room_status	更新 Room 对象的状态为"已入住"，设置当前占用顾客 ID
方法	save_order	保存订单到数据库，返回保存结果

类_3:Room

类型	名称	描述说明
属性	room_id	房间唯一编号
属性	room_type	房间类型
属性	status	房间状态
属性	current_customer	当前占用房间的顾客
方法	is_available	判断房间状态是否为"空闲"，返回布尔值
方法	set_occupied	设置房间状态为"已入住"，并关联当前顾客

类_4:AccommodationOrder

类型	名称	描述说明
属性	order_id	自动生成的唯一订单编号
属性	customer	入住的顾客对象
属性	room	被占用的房间对象
属性	status	订单状态
属性	create_time	订单创建时间
方法	init_order	初始化订单基础属性
方法	save	持久化订单数据

类_5:Customer

类型	名称	描述说明
属性	customer_id	唯一顾客编号
属性	name	顾客姓名
属性	id_card	身份证号
属性	phone	联系方式
属性	current_room	当前入住的房间
方法	get_info	返回顾客基本信息

4.3 用例:UC_办理结账

1.

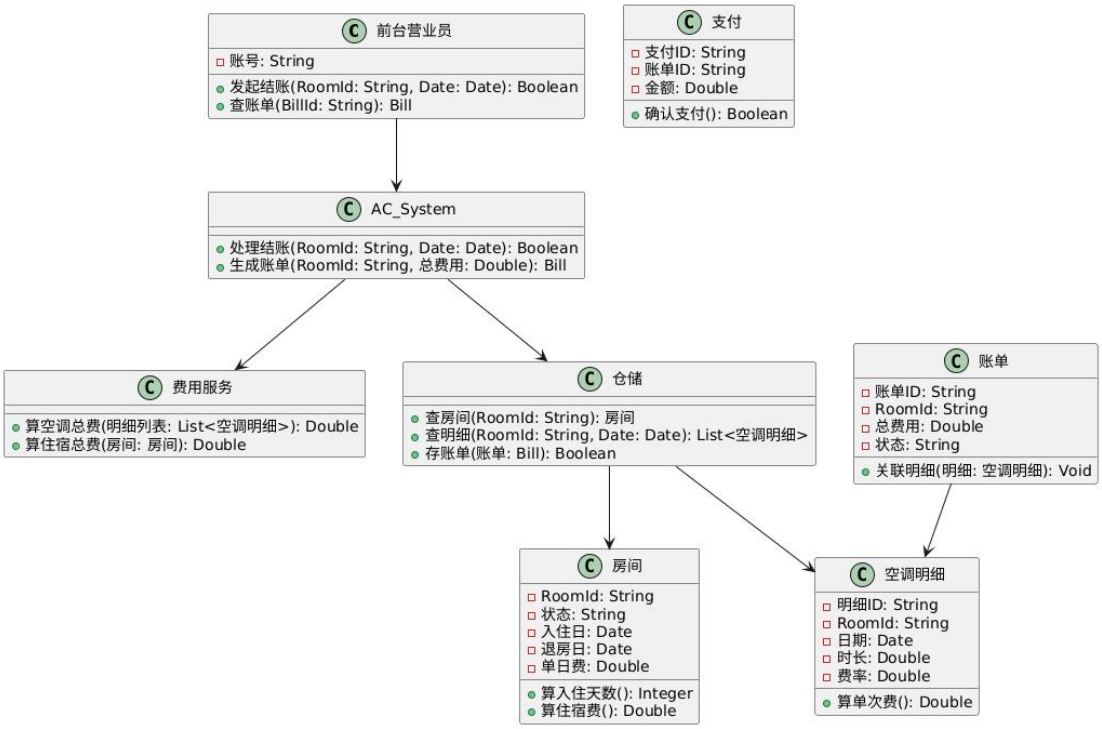

图 19 办理结账类图

2.属性说明

类_1：前台营业员

类型	名称	描述说明
属性_1	账号	前台营业员的登录标识（如 “FD_001”）
属性_2	权限	营业员的操作权限范围(如“结账管理”)
方法_1	发起结账	提交指定房间的结账请求
方法_2	查账单	根据账单 ID 查询账单详情

类_2：房间

类型	名称	描述说明
属性_1	RoomId	房间唯一标识（如 “302”）
属性_2	状态	房间当前状态（如 “已入住 / 空闲”）
方法_1	算入住天数	计算房间的入住时长（单位：天）
方法_2	算住宿费	计算总住宿费用（入住天数 × 单日费）

类_3：空调明细

类型	名称	描述说明
属性_1	明细 ID	空调使用明细的唯一标识
属性_2	时长	空调单次使用时长（单位：小时）
方法_1	算单次费	按 “时长 × 费率” 计算单次费用
方法_2	关联房间	绑定明细对应的房间

类_4：账单

类型	名称	描述说明

属性_1	账单 ID	账单的唯一标识
属性_2	总费用	账单的金额（如 “850.0”）
方法_1	关联明细	绑定空调明细到当前账单
方法_2	查状态	查询账单当前状态（如 “已生成”）

类_5：AC_System

类型	名称	描述说明
属性_1	处理状态	记录结账流程的进度（如 “处理中”）
属性_2	服务列表	依赖的业务服务集合
方法_1	处理结账	协调 “查房间→算费用→生成账单” 全流程
方法_2	生成账单	创建并初始化账单对象

类_6：仓储

类型	名称	描述说明
属性_1	数据库连接	与数据库的连接实例
属性_2	表名	操作的数据表名称（如 “room”）
方法_1	查房间	根据 RoomId 查询房间信息
方法_2	存账单	将账单数据保存到数据库

5. 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 2 作业工作内容及工作量统计

		张晨旭	杨珺涵	许世典	覃疆楠	张睿佳
软件架构	软件架构示意图				√	
	分层结构说明				√	
界面设计	控制面板					√
	办理入住					√
	办理结账					√
	监控空调					√
UC_使用空调	PowerOn 无限制			√		
	PowerOn 有限制	√				
	ChangeTemp				√	
	ChangeSpeed				√	
	PowerOff	√				
UC_办理入住	Create_Accomodation_Order			√		
UC_办理结账	Create_Accomodation_Bill					√
	Create_AC_Bill		√			
	Create_DetailRecord_AC		√			
静态结构设计	UC_使用空调	√				
	UC_办理入住			√		
	UC_办理结账		√			