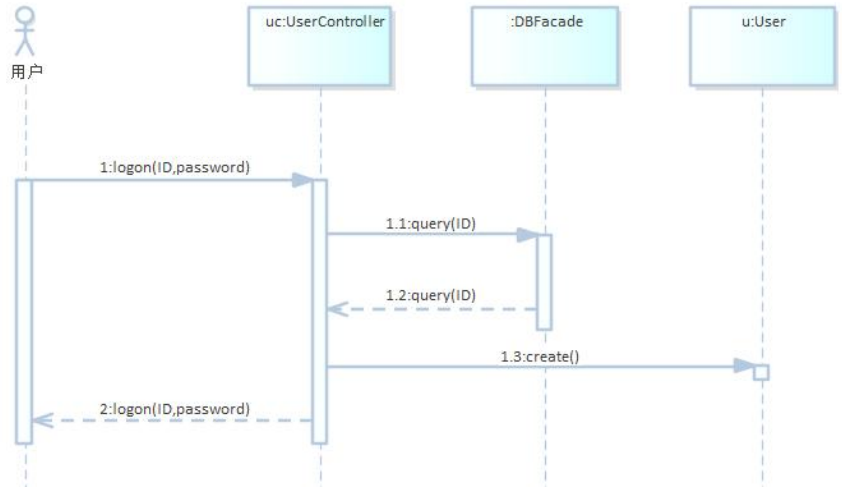
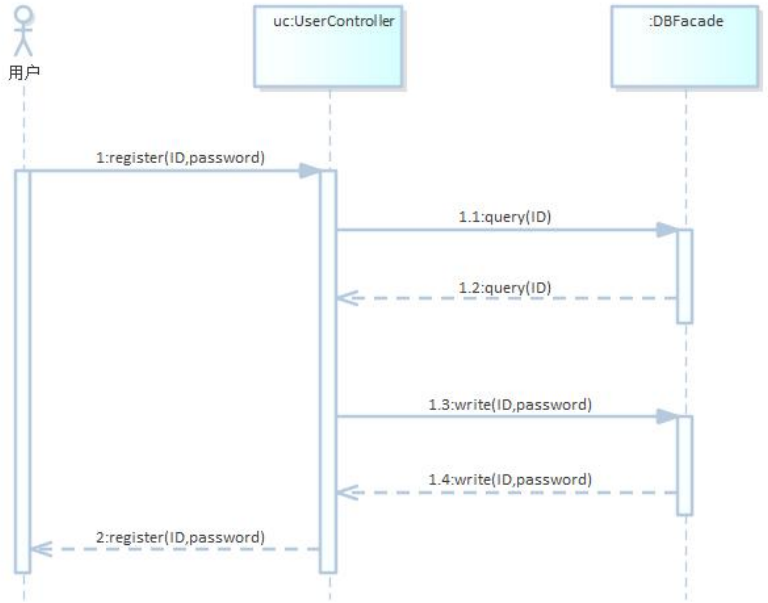
**用户相关：**

1. 用户登录



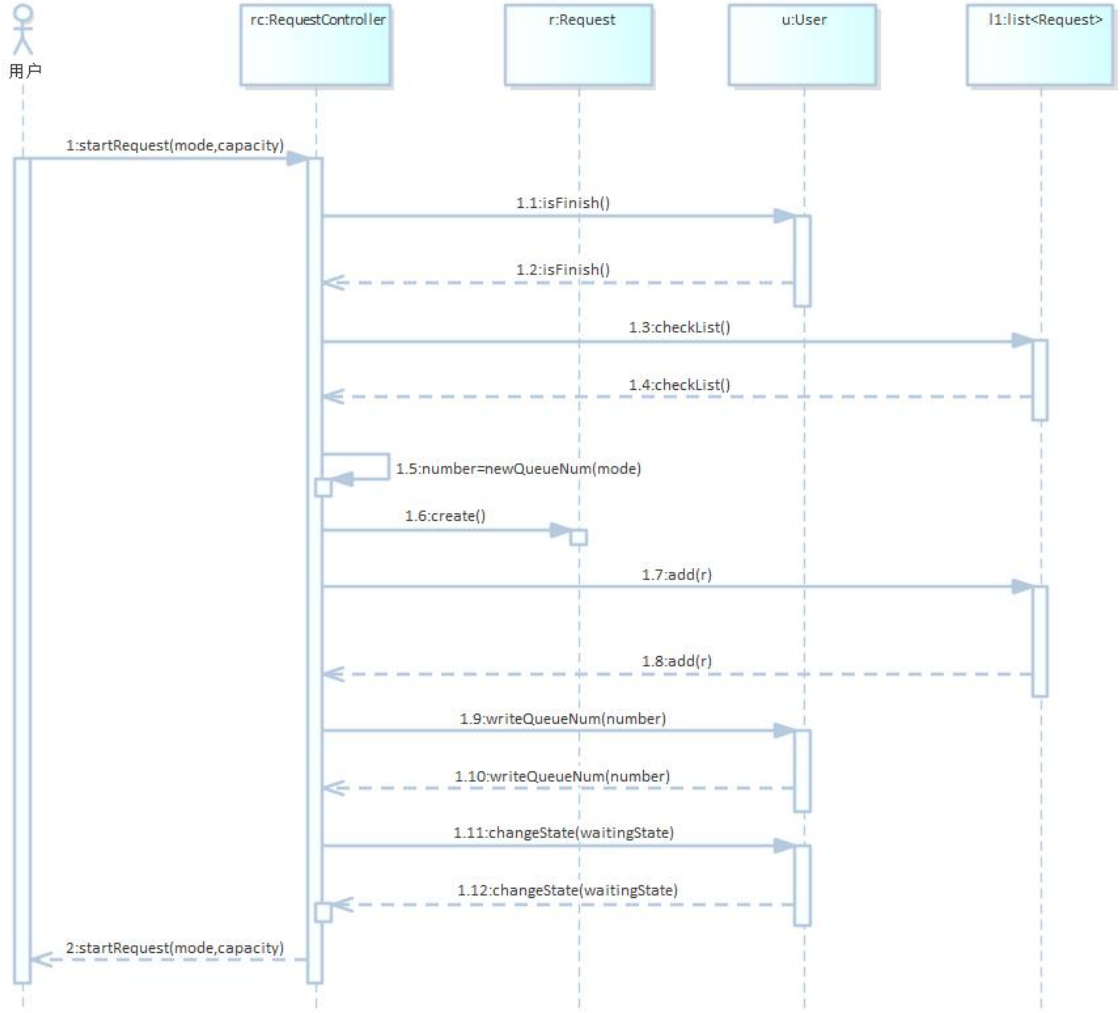

用户将 logon(ID,password)消息发给控制器 uc，其中 ID 和 password 分别表示用户名和密码，之后控制器通过消息 1.1 调用 DBFacade 对象，根据 ID 从数据库中检索出用户所对应的项，并比对密码是否正确。如果密码正确则创建一个用户对象 u。（注：由于画图软件的限制，创建消息没有指向被创建对象的顶部）

2. 用户注册



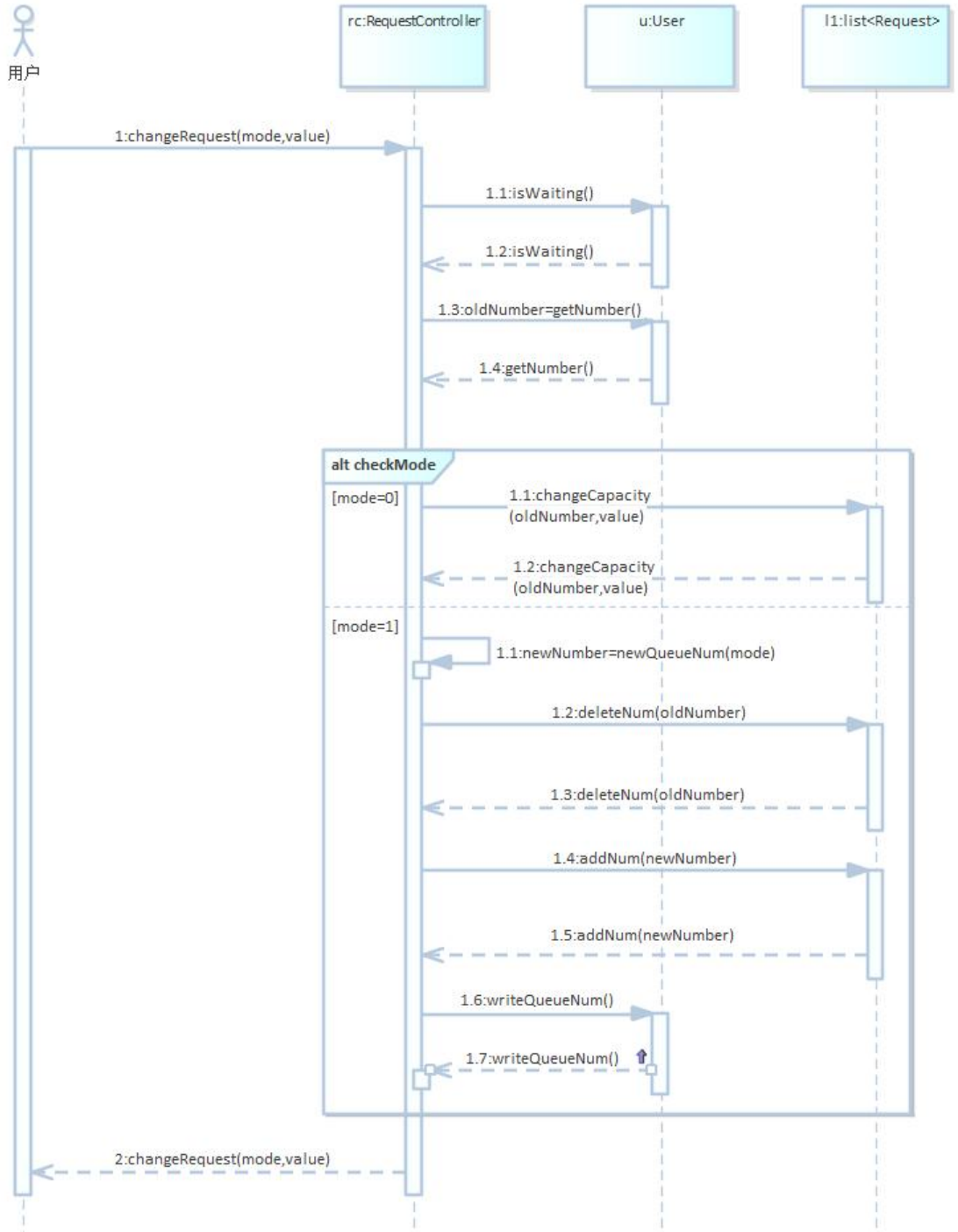

用户将 register(ID,password)消息发给控制器 uc，之后通过消息 1.1 调用 DBFacade 对象，根据 ID 检索用户对应的项，如果没查到就说明没注册过，可以通过消息 1.3 在数据库中写入用户信息。

3. 提交充电请求




用户将 startRequest(mode,capacity)发送给控制器 rc，其中 mode 和 capacity 分别表示充电模式和充电量。控制器首先通过 isFinish()检查用户是否有未完成的请求，如果没有，则 isFinish()返回 true，继续后面的处理。之后通过 checkList()检查等候区是否还有空闲车位，如果还有就分配一个新的排队号 number，创建一个请求对象 r，并通过消息 1.7 将它加入到等候区队列 l1 中。最后通过 writeQueueNum(number)把排队号记录在用户类中（后面要用排队号来查询用户的请求），并通过 changeState(waitingState)将用户的状态改为等待区状态（用户一共有 3 种状态，分别是等待区状态、充电区状态和完成状态，处于哪种状态由用户类中的一个整数属性 state 来表示）。（注：由于画图软件的限制，对自身的消息传递没有返回，如消息 1.5）

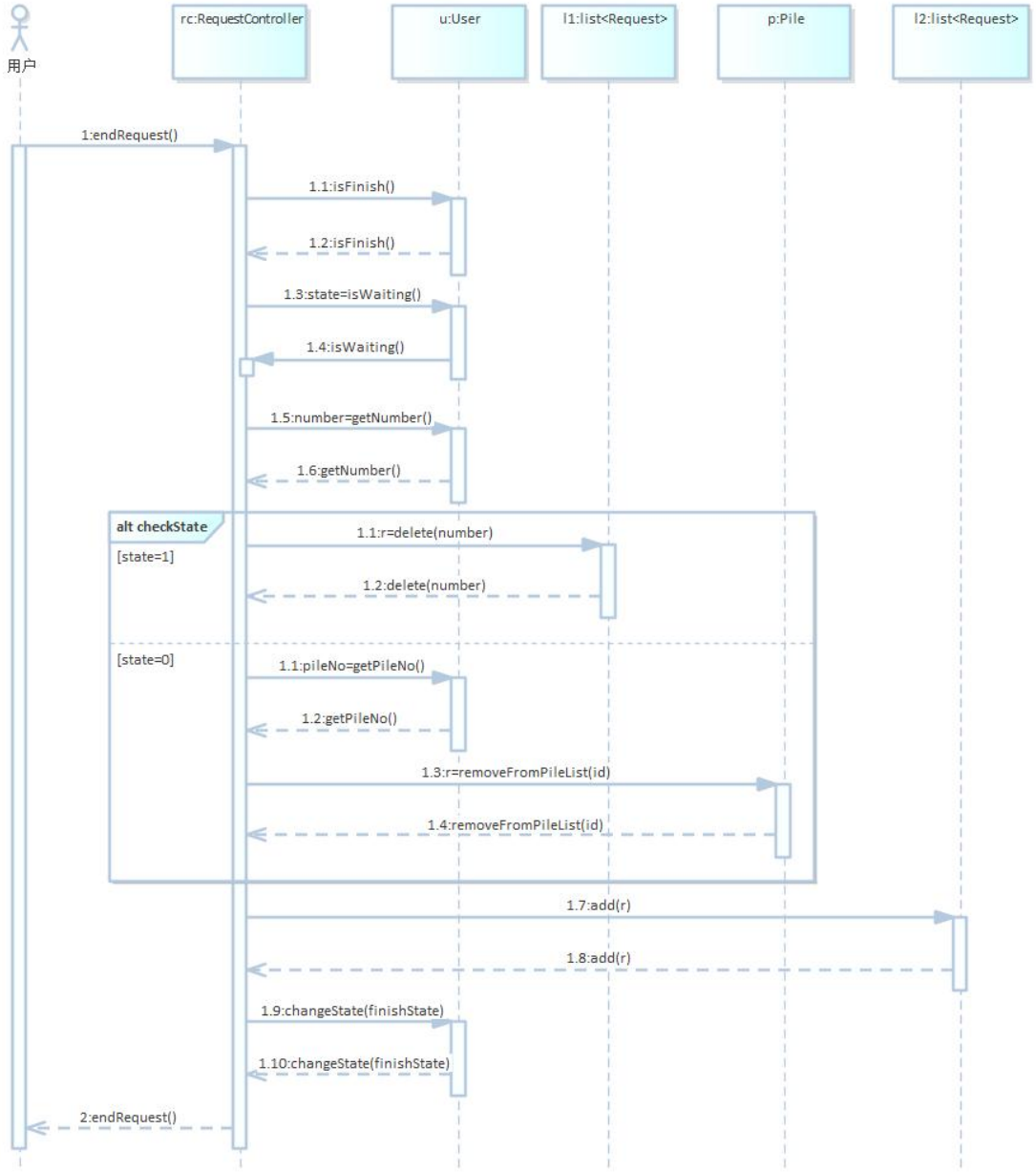
4. 修改充电请求




用户向控制器发送修改请求 changeRequest(mode, value)，其中 mode 表示用户想修改充电模式还是充电量（0 表示充电量，1 表示充电模式），value 表示修改后的值。控制器收到请求后，通过 isWaiting()检查用户是否处于等待区，只有处于等待区时，请求才会被受理。

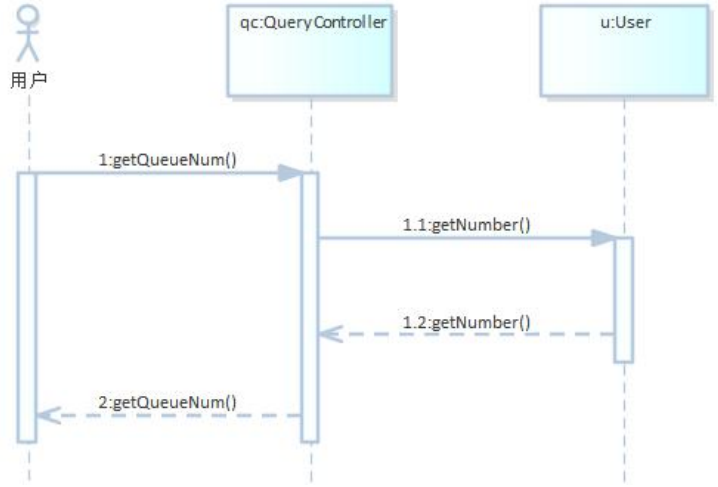
请求被受理后，控制器获取用户的排队号。如果用户要改充电量，控制器就通过 changeCapacity(oldnumber,value)在等待区队列中找到用户的请求并修改充电量。如果用户要改充电模式，控制器就先分配一个新的排队号，之后在队列中取出用户之前的请求，修改排队号后加到队列的最后。需要注意的是，用户类中记录的排队号也需要更新。

5. 取消/结束充电



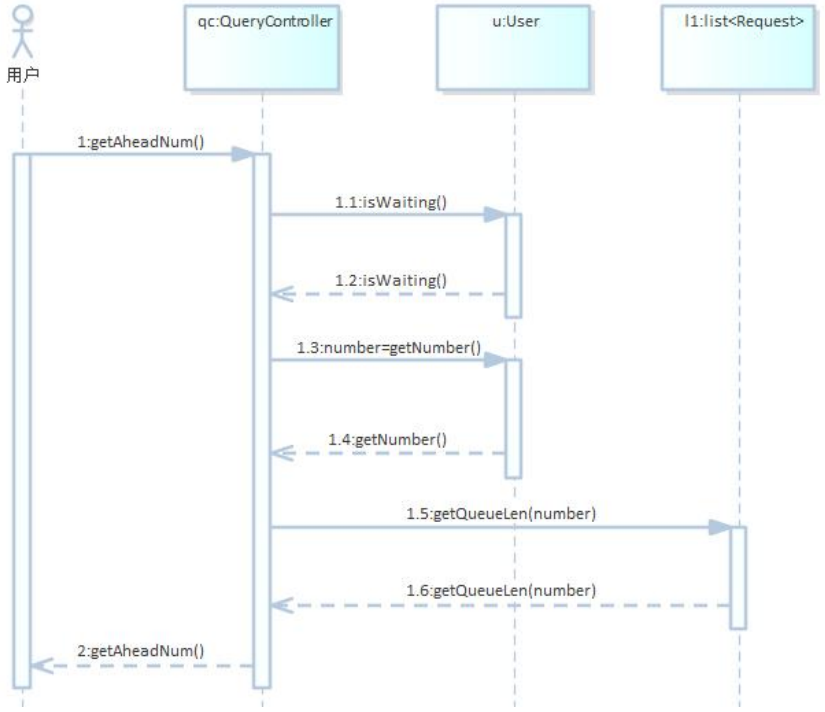

用户将 endRequest()消息发给控制器 rc。rc 首先检查用户是否已经处于结束状态，如果不是则检查是否在等待区并取得用户的排队号。如果用户在等待区（state=1）则从队列里删除用户的请求。如果用户不在等待区（state=0）就说明用户在充电区，此时获取用户所在的充电桩号，然后从充电桩的排队队列中删除用户的请求（注意删除请求的时候要在请求中写入停止时间并计算充电时长及有关费用，同时累加到充电桩内部的属性上，用于之后的详单和报表生成）。最后通过消息 1.7 将用户请求加入完成队列 l2，并改变用户状态为完成态。（注：充满电后充电桩也可以自动结束充电，其操作与 state=0 时的情况类似）

6. 查看本车排队号码



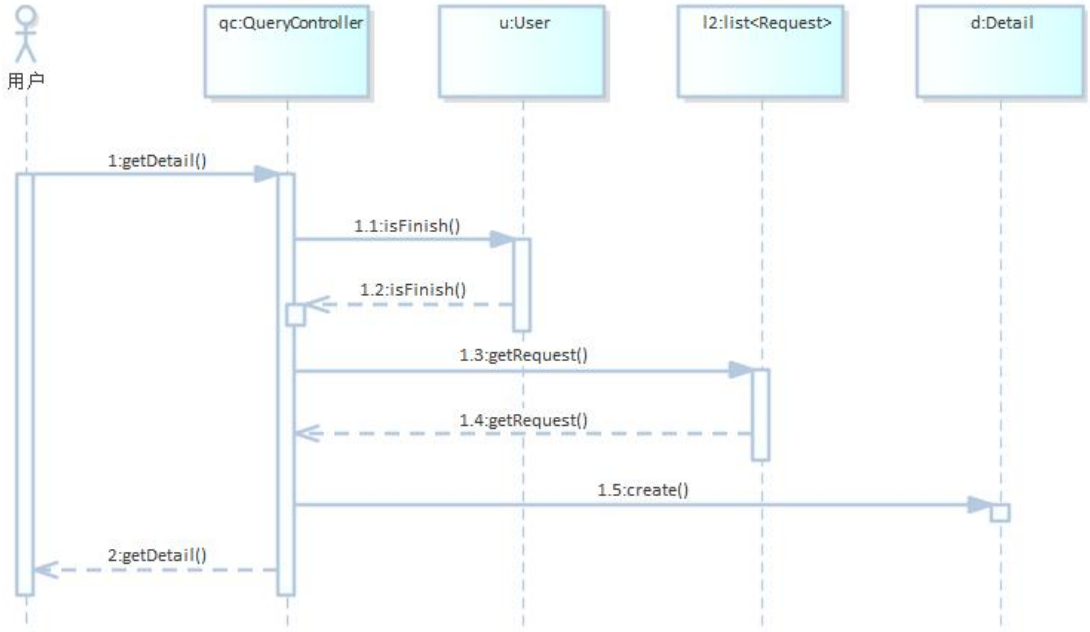

用户向控制器提交 getQueueNum()请求，之后控制器调用用户类的 getNumber()方法即可获取排队号。

7. 查看本充电模式下前车等待数量




用户使用 getAheadNum()向控制器传递消息，查询本充电模式下前车等待数量。控制器通过 isWaiting()检查用户是否在等待区，如果是就进行后面的操作。控制器通过 getNumber()获取用户的排队号，之后用此排队号在 l1 中查询前车等待数量。

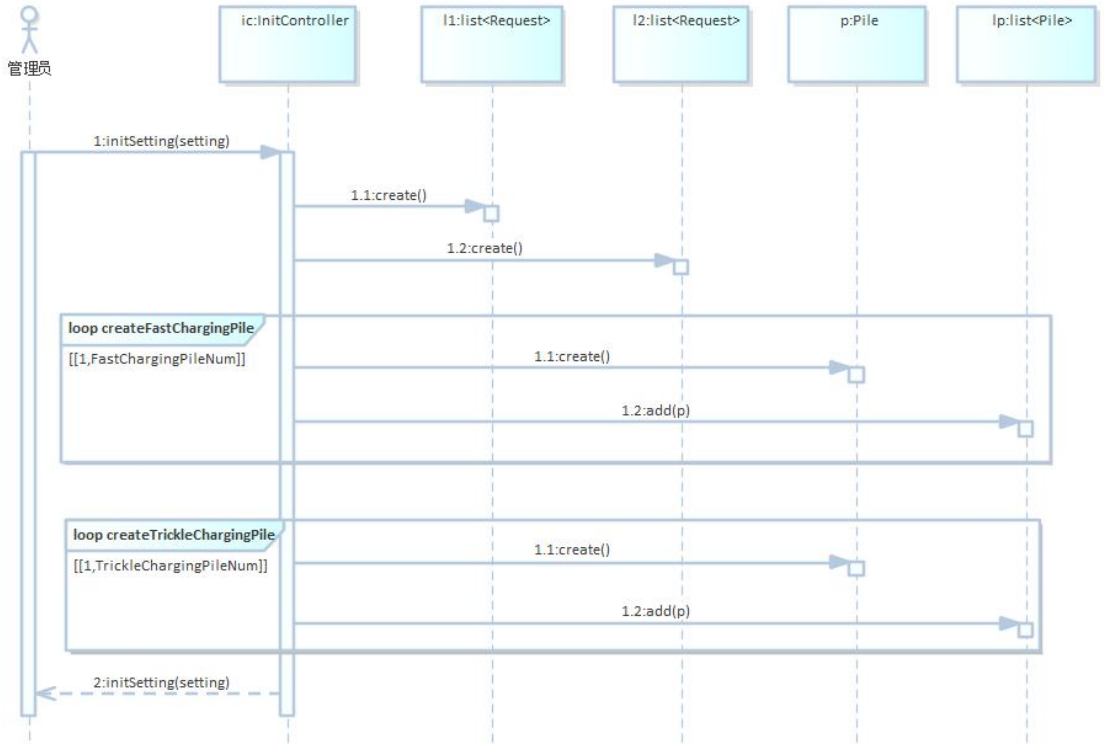
8. 查看详单




用户将 getDetail()消息发给控制器 qc。qc 首先查看用户充电请求是否已完成，如果已完成就从完成队列里找到此用户的请求，然后用该请求生成详单对象。

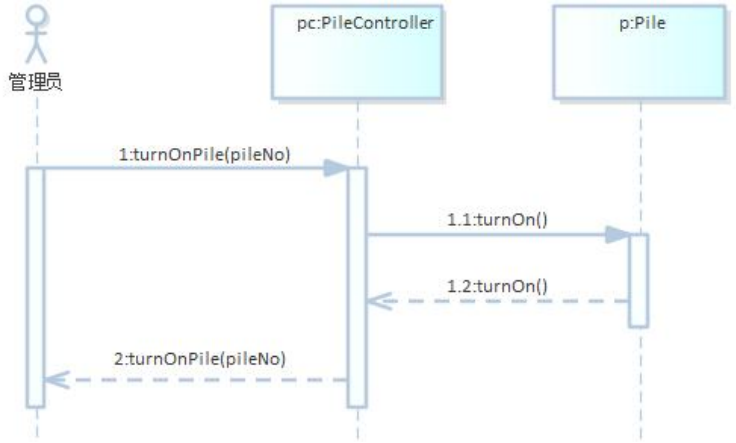
**管理员相关：**

1. 系统初始化



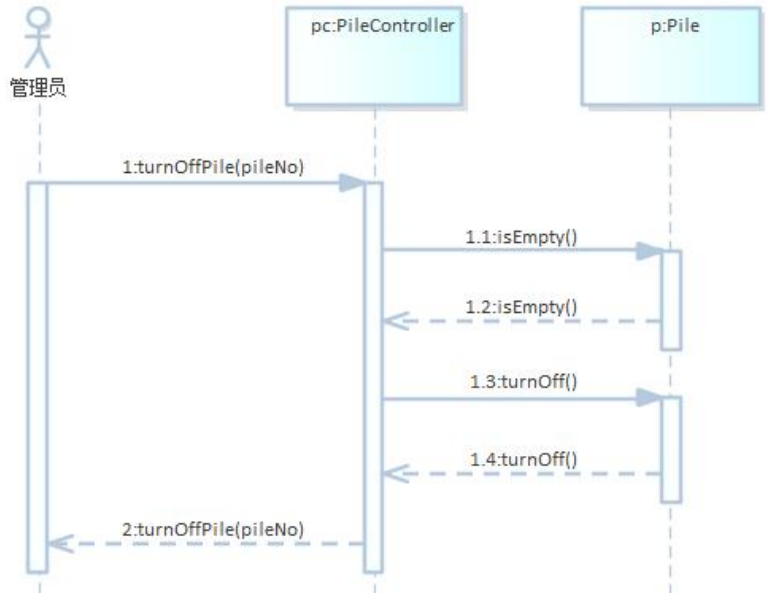

管理员设定好启动参数后发送消息 initSetting(setting)传入设置信息（setting 中的内容包括：快充电桩数（FastCharingPileNum）、慢充电桩数（TrickleChargingPileNum）、等候区车位容量（WaitingAreaSize）和充电桩排队队列长度（ChargingQueueLen））。控制器 ic 首先创建等候区队列 l1 和完成队列 l2，之后通过循环创建快充电桩和慢充电桩，并把它们加入充电桩队列 lp。

2. 开启充电桩：



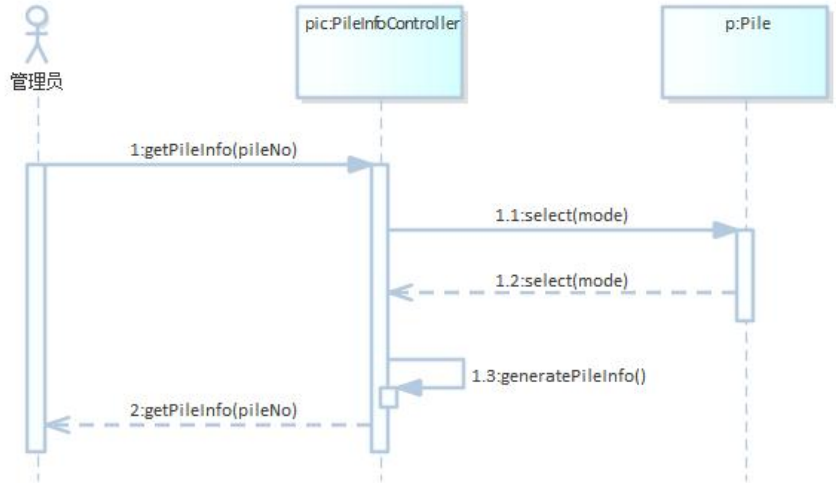

管理员（在 GUI 界面）输入动作“打开充电桩”和目标充电桩的编号，然后发送 turnOnPile(pileNo)消息给控制器 pc，pc 调用充电桩类的 turnOn()方法来打开充电桩。

3. 关闭充电桩



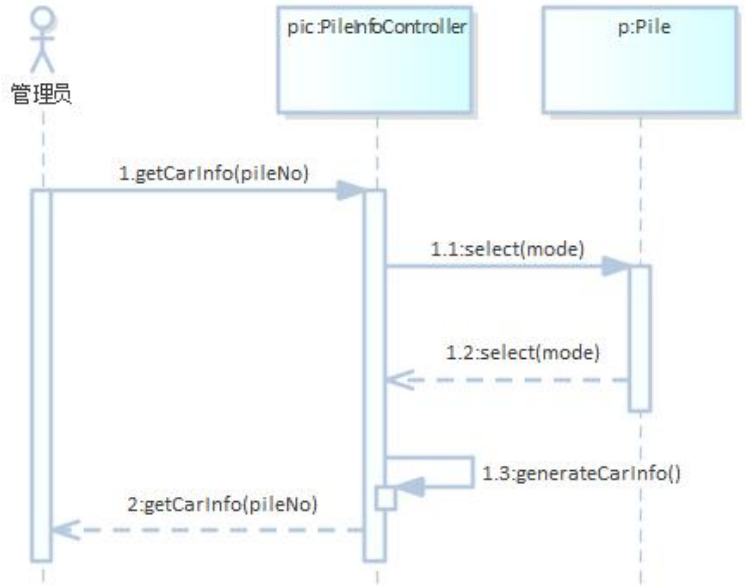

管理员（在 GUI 界面）输入动作“关闭充电桩”和目标充电桩的编号，然后发送 turnOffPile(pileNo)消息给控制器 pc，pc 首先检查充电桩的队列是否为空，如果为空则调用充电桩类的 turnOff()方法关闭充电桩。

4. 查看充电桩状态



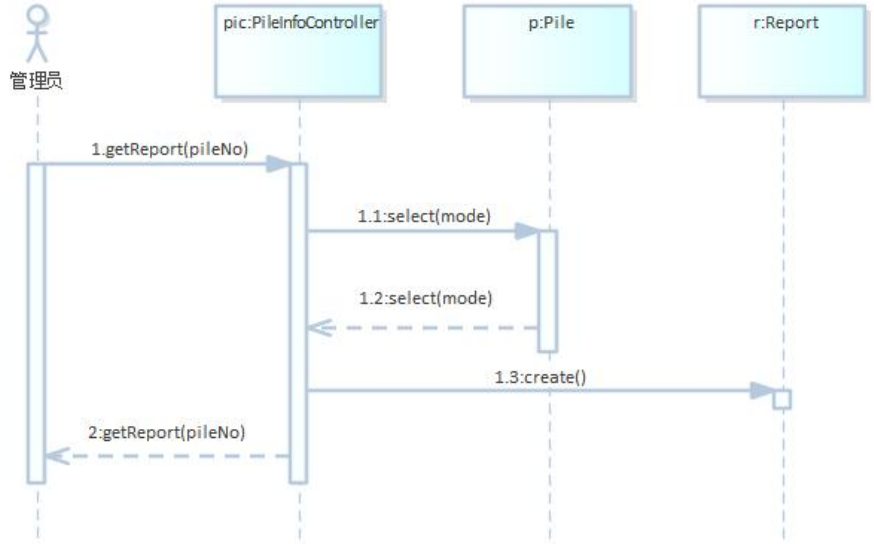

管理员（在 GUI 界面）输入动作“查询充电桩状态”和目标充电桩的编号，然后调用控制器 pic 的方法 getPileInfo(pileNo)。getPileInfo(pileNo)使用 select 方法访问充电桩，返回查询结果（mode 用来选择要查询的数据）。之后用 generatePileInfo()对查询结果进行处理，产生要返回的数据。

5. 查看各充电桩等候服务的车辆信息




管理员要查询某（或所有）充电桩的等候服务的车辆的信息，将这个请求传给控制器 pic，参数为待查询的充电桩唯一编号。与查看充电桩状态的操作类似，先调用 select 来获取数据，再用 generateCarInfo()对其进行处理。

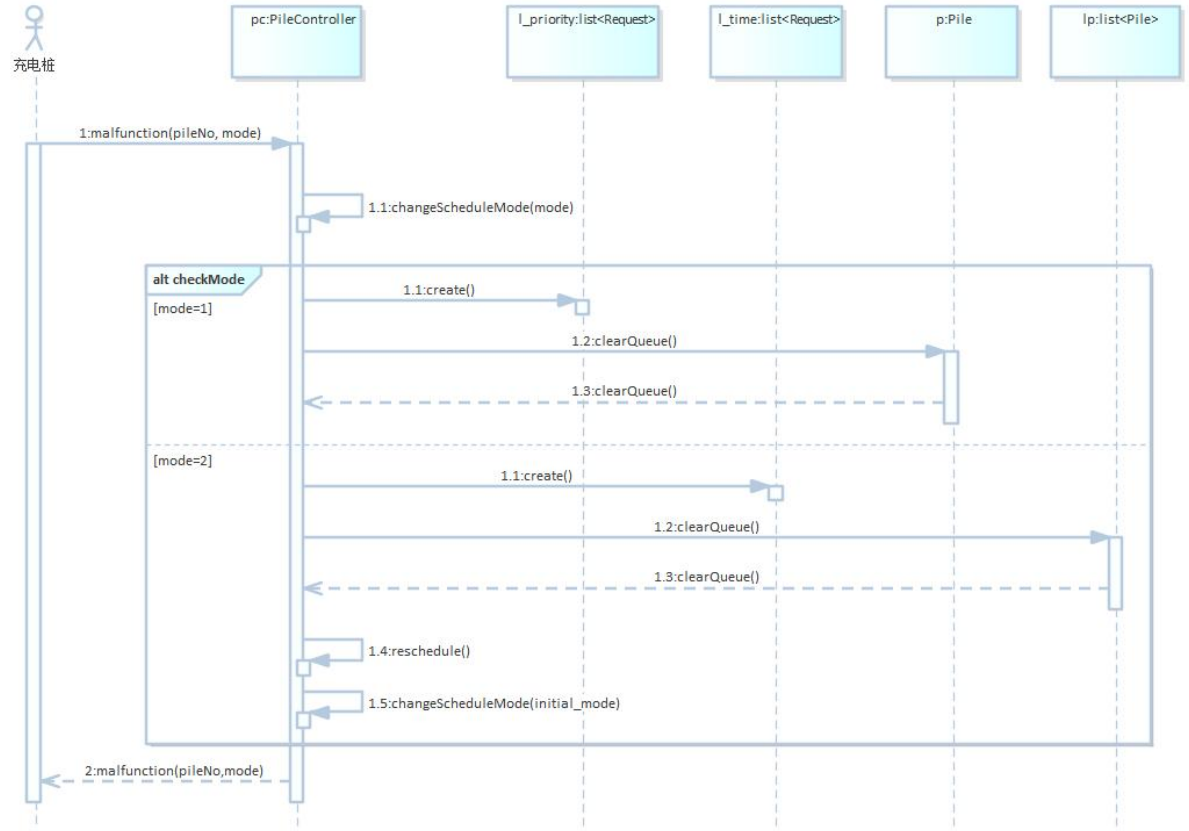
6. 报表展示




管理员发送 getReport(pileNo)消息给控制器 pic，pic 先用 select 获取需要的数据，再创建报表对象 r。

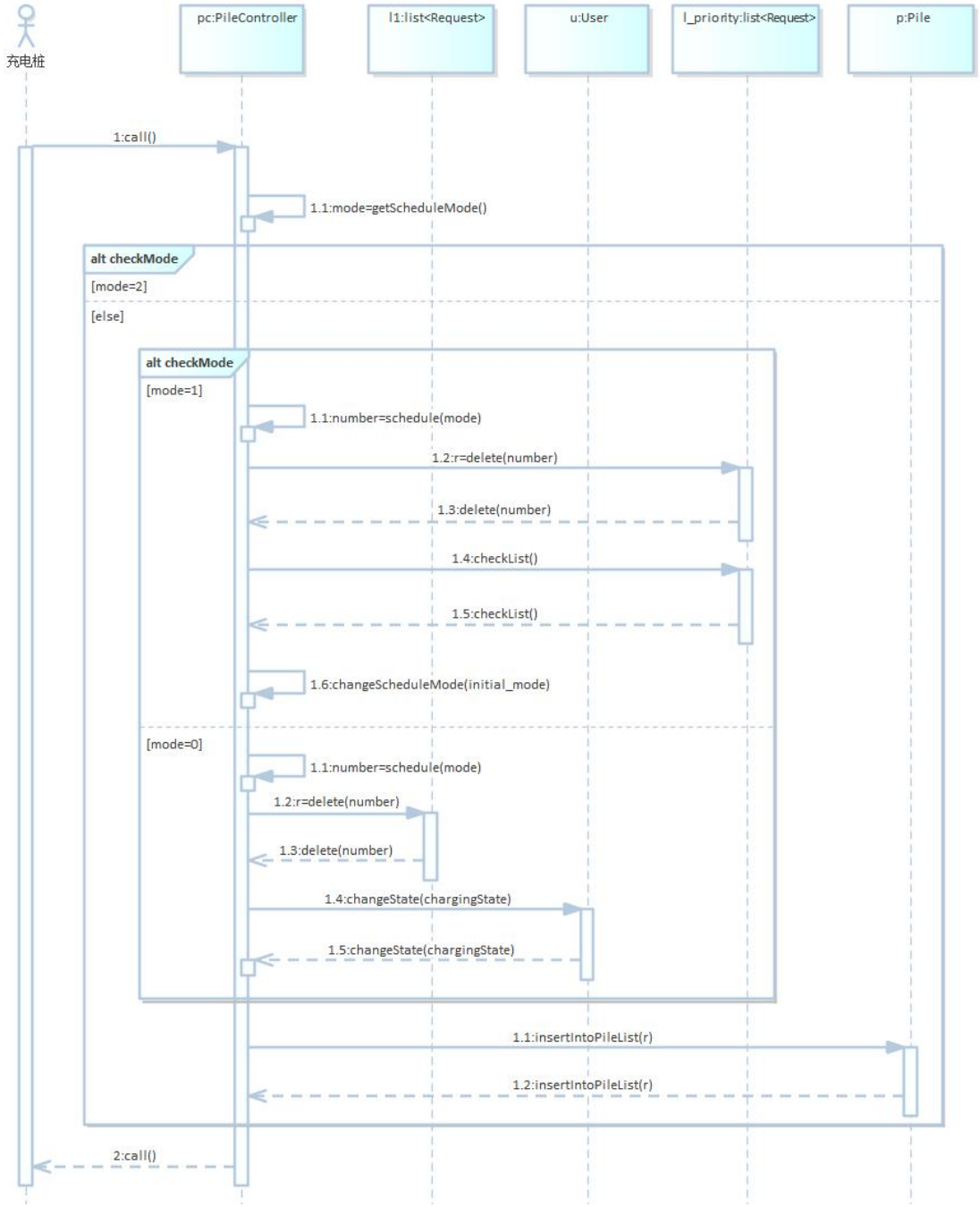
**充电桩相关：**

1. 充电桩故障




当充电桩出现故障时，通过 malfunction(pileNo,mode)通知控制器 pc，其中 pileNo 代表充电桩编号，mode 代表选择的调度方式（1 代表优先级调度，2 代表时间顺序调度）。控制器首先通过 changeScheduleMode(mode)来改变其内部属性 scheduleMode 的值（scheduleMode 有 3 个可能值，一般情况下为 0，也就是从等候区队列中调度；发生故障时可能为 1 或 2，含义如前所述），之后根据 mode 的值进行不同的操作。如果 mode 为 1，说明进行优先级调度，此时用故障充电桩的等候队列创建一个优先级队列 l_priority，并把故障充电桩的等候队列清空。如果 mode 为 2，说明进行时间顺序调度，此时用与故障充电桩充电模式相同的所有充电桩中等待充电的车辆创建一个时间顺序队列 l_time，之后用 reschedule()进行重新调度，调度完成后用 changeScheduleMode(initial_mode)将调度模式改为 0。（注：当充电桩故障恢复时，也是进行时间顺序调度，与 mode=2 的情况类似）

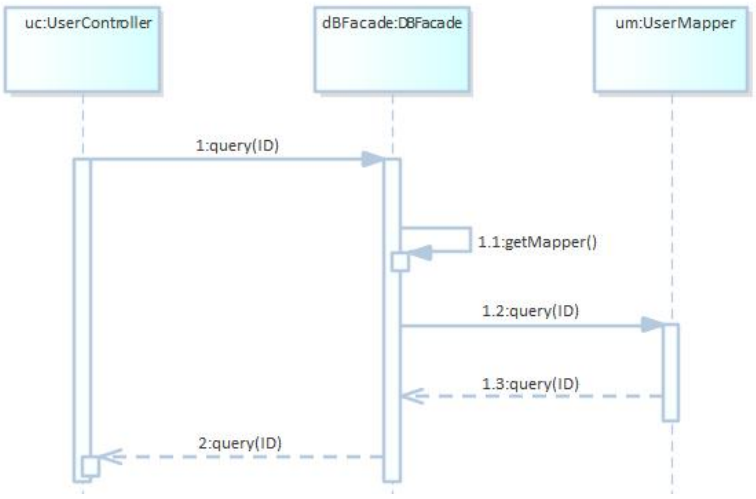
2. 充电桩叫号




当充电桩中出现空位时，发送call()消息给控制器pc进行叫号。pc首先通过getScheduleMode()获取当前调度模式。如果模式为2，说明为时间顺序调度，此时什么也不做。如果模式为1，说明为优先级调度，此时按此种调度方式从优先级队列中选一个充电请求并将其从优先级队列中移除，之后调用checkList()检查优先级队列是否为空，如果为空则将调度模式改为0，相当于开启等候区叫号服务。如果模式为0，则从等候区队列中选一个充电请求并将其从等候区队列中移除，之后根据获取的请求中的用户ID找到对应的用户对象并把用户状态置为充电区态。无论是模式0还是模式1最终都要把获取到的充电请求加入充电桩队列中。

**持久化层设计：**

1. 用户控制器查询数据库




2. 用户控制器写入数据库