# 预约充电

### 功能说明

用户手机上选择充电桩后进行预约操作。

### 序列图

预约操作序列图如下所示：



### 流程说明

1. 预约请求：APP用户点击预约按钮，发起预约申请，消息交由APP接口处理，消息接口格式参考接口定义。
2. 查看充电的桩的预约费用信息，并返回给APP，同时页面跳转到预约界面。
3. 在预约界面选择时长，点击确定预约，则APP接口则开始请求桩状态并验证。
4. 请求充电桩状态：APP接口调用Redis缓存，获取充电桩预约状态。

|  |  |
| --- | --- |
| **缓存状态定义** | **说明** |
| status | 充电桩充电状态 (PileStatusEnum) |
| appstatus | 预约状态 (PileAppStatusEnum) |
| userid | 预约用户编码 |
| appendtime | 预约结束时间 |

1. 验证桩状态：
2. 如果充电桩正在充电则不能预约。
3. 如果预约状态为已预约，且未到预约截止时间，则不能预约。
4. 如果预约状态为已预约，且已超过预约截止时间，则更新Redis缓存，预约状态变为未预约，预约用户编码userid为空。
5. 如果不可预约，则返回信息给APP,例如显示“充电桩正忙，请稍候再约”。
6. 查看用户是否已有预约（本桩或其他桩）：
7. 不可预约：提示用户您当前已有预约，不可重复预约。
8. 可预约：查看用户是否有未缴费。
9. 查看用户是否有未缴费预约记录：
10. 有未缴费：提示用户存在预约未缴费，不能再预约，请先缴清费用。
11. 没有未缴费：则修改Redis缓存里桩状态为“请求预约”，并设置userid为当前操作的用户，此操作的作用为锁定桩，避免其他用户进行预约请求，并产生为唯一的预约号。然后再进入下一步APP接口发送请求预约命令。

预约号产生规则：年（两位）+月（两位）+日（两位）+时（两位）+分（两位)）+序号（2位)

1. APP接口发送请求预约命令：APP接口向RecketMQ消息中间件发送预约消息命令，发送后定时循环判断Redis缓存中充电桩状态。

|  |  |
| --- | --- |
| **预约命令定义** | **说明** |
| userid | 用户编码 |
| pileid | 充电桩编码 |
| appointmentid | 预约号 |
| endtime | 预约结束时间 |

1. 下发预约命令：
2. 前置机收到请求预约命令后，先从Redis缓存中获取充电桩状态并验证。此时，若不可预约，则直接返回。若可预约，则向充电桩发送预约命令。
3. 向充电桩发送请求预约命令，并接受充电桩返回结果。如果成功则修改Redis缓存的预约状态为“预约成功”；如果失败，则判断Redis缓存中预约状态是否为“请求预约”且预约人编码是否为当前用户编码，若是，则修改缓存预约状态为当前的结果，若不是，则不管。
4. APP接口校验充电桩状态：如果Redis缓存的预约状态为“预约成功”且userid和当前请求一致，则判断预约成功。如果检验状态失败(超时)或预约失败，则修改Redis缓存里桩状态为“未预约”并设置userid为空，并返回提示用户预约失败，如果预约状态为“预约成功”，调用APP存储记录接口。
5. APP接口存储记录：APP接口计算预约费用，存储当前预约成功用户的预约信息和费用信息到数据库中，此时该预约记录为支付状态为支付中，并将预约记录ID（apprecordid）存入缓存中，再将预约结果和费用返回用户，方便用户缴费。
6. APP界面跳转到支付确认界面，点击“确定支付”，完成支付，则调用APP支付接口，根据支付结果更新预约记录的支付相关信息（支付方式、支付帐户信息、支付状态）。

### 参数说明

1. Redis缓存桩状态对象为PileStatusBean，与预约相关的参数与方法：

|  |  |  |
| --- | --- | --- |
| **参数名称** | **类型** | **说明** |
| id | Integer | 桩id |
| status | PileStatusEnum | 充电状态：用于判断桩是否在充电中 |
| appstatus | PileAppStatusEnum | 预约状态：用户判断桩当前是可预约或更新当前桩的预约情况 |
| userid | Integer | 已成功预约的用户编码 |
| appendtime | Date | 预约结束时间：用户获取缓存桩状态时，如果预约状态为已预约，则根据预约结束时间重新更新缓存里的预约状态。 |
| apprecordid | Integer | 预约记录ID：预约成功以后，产生预约记录后，将预约记录ID存入缓存 |
| updatetime | Date | 缓存数据的更新时间 |

获取充电桩状态对象缓存：

PileStatusBean pileStatusBean = ChargingCacheUtil.*getPileStatusBean*(pileid);

设置充电桩状态对象缓存：

ChargingCacheUtil.*setPileStatusBean*(pileStatusBean);

1. RecketMQ消息与预约相关的参数与方法：

|  |  |  |
| --- | --- | --- |
| **参数名称** | **类型** | **说明** |
| pileid | Integer | 桩id |
| userid | Integer | 用户ID |
| appointmentid | String | 预约号 |
| endtime | Date | 预约结束时间 |

发送前置消息的方法：

MsgAppointmentReq msg = new MsgAppointmentReq(pileid,userid,appointmentid,endtime));

mqMsgProduct.sendMQMsg(msg);

1. 数据库预约相关库表：

**表名:**BUS\_APPOINTMENT **解释:**预约表(BUS)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BUS\_APPOINTMENT(预约表(BUS))** | | | | | | | | |
| **是否主键** | **字段名** | **字段描述** | **数据类型** | **长度** | **可空** | **约束** | **缺省值** | **备注** |
| 是 | ID | 预约ID | INT(11) | 11 |  |  |  |  |
|  | USER\_ID | 用户ID | INT(11) | 11 |  |  |  |  |
|  | PILE\_ID | 桩ID | INT(11) | 11 |  |  |  |  |
|  | APP\_NO | 预约号 | VARCHAR(50) | 50 |  |  |  |  |
|  | START\_TIME | 预约开始时间 | DATETIME |  | 是 |  |  |  |
|  | END\_TIME | 预约结束时间 | DATETIME |  | 是 |  |  |  |
|  | APP\_LEN | 预约时长 | INT(4) | 4 | 是 |  |  |  |
|  | APP\_FEE | 预约费用 | DECIMAL(10,3) | 10 | 是 |  |  |  |
|  | APP\_STATUS | 预约状态 | SMALLINT(1) | 1 | 是 |  | 1 | 1.预约中2.履约3.取消4.过时 |
|  | PAY\_WAY | 支付方式 | SMALLINT(1) | 1 | 是 |  |  | 1.资金账户2.支付宝3.微信4.银联5.信用卡 |
|  | ACCOUNT\_INFO | 支付账户信息 | VARCHAR(100) | 100 | 是 |  |  |  |
|  | PAY\_STATUS | 支付状态 | SMALLINT(1) | 1 | 是 |  |  | 1.未支付2.支付中3.支付成功4.支付失败 |
|  | ADD\_TIME | 提交时间 | DATETIME |  | 是 |  |  |  |

# 终止预约

### 功能说明

用户在预约中界面进行取消预约操作。

### 序列图：

预约操作序列图如下所示：



### 流程说明

1. 取消预约请求：APP用户点击取消预约按钮，发起终止预约申请，消息交由APP接口处理，接口参数为预约记录id。
2. 根据预约记录id查询改条详细的预约记录，并判断预约状态
3. 如果该记录对应的userid不是当前操作用户，则提示“当前用户未预约”。
4. 如果预约状态为“过时”，则提示“预约已过期”；
5. 如果预约状态为“预约中”，则可允许取消预约，修改Redis缓存里桩状态为“请求取消预约”，然后再进入下一步APP接口发送请求取消预约命令。
6. APP接口发送请求取消预约命令：APP接口向RecketMQ消息中间件发送取消预约消息命令，发送后定时循环判断Redis缓存中充电桩状态。

|  |  |
| --- | --- |
| **预约命令定义** | **说明** |
| appointmentid | 预约号 |

1. 下发取消预约命令：
2. 前置机收到请求取消预约命令后，先从Redis缓存中获取充电桩状态并验证。此时，若状态为请求终止预约，则向充电桩发送取消预约命令，否则直接返回。
3. 向充电桩发送请求取消预约命令，并接受充电桩返回结果。如果成功则修改Redis缓存的预约状态为“未预约”；如果失败，则判断Redis缓存中预约状态是否为“请求取消预约”且预约人编码是否为当前用户编码，若是，则修改缓存预约状态为当前的结果，若不是，则不管。
4. APP接口校验充电桩状态：如果Redis缓存的预约状态为“未预约”，则判断预约成功。如果检验状态失败(超时)或取消预约失败，则修改Redis缓存里桩状态为原先的“预约成功”，并返回提示用户取消预约失败，如果预约状态为“未预约”，调用APP更新记录接口。
5. APP接口更新记录：将数据库中对应的预约记录的预约状态改为“取消”，并提示用户“取消预约成功”。

### 参数说明

1. RecketMQ消息与预约相关的参数与方法：

|  |  |  |
| --- | --- | --- |
| **参数名称** | **类型** | **说明** |
| appointmentid | String | 预约号 |

发送前置消息的方法：

MsgAppointmentReq msg = new MsgAppointmentReq(appointmentid);

mqMsgProduct.sendMQMsg(msg);

1. 数据库预约相关库表：

**表名:**BUS\_APPOINTMENT **解释:**预约表(BUS)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BUS\_APPOINTMENT(预约表(BUS))** | | | | | | | | |
| **是否主键** | **字段名** | **字段描述** | **数据类型** | **长度** | **可空** | **约束** | **缺省值** | **备注** |
| 是 | ID | 预约ID | INT(11) | 11 |  |  |  |  |
|  | USER\_ID | 用户ID | INT(11) | 11 |  |  |  |  |
|  | PILE\_ID | 桩ID | INT(11) | 11 |  |  |  |  |
|  | APP\_NO | 预约号 | VARCHAR(50) | 50 |  |  |  |  |
|  | START\_TIME | 预约开始时间 | DATETIME |  | 是 |  |  |  |
|  | END\_TIME | 预约结束时间 | DATETIME |  | 是 |  |  |  |
|  | APP\_LEN | 预约时长 | INT(4) | 4 | 是 |  |  |  |
|  | APP\_FEE | 预约费用 | DECIMAL(10,3) | 10 | 是 |  |  |  |
|  | APP\_STATUS | 预约状态 | SMALLINT(1) | 1 | 是 |  | 1 | 1.预约中2.履约3.取消4.过时 |
|  | PAY\_WAY | 支付方式 | SMALLINT(1) | 1 | 是 |  |  | 1.资金账户2.支付宝3.微信4.银联5.信用卡 |
|  | ACCOUNT\_INFO | 支付账户信息 | VARCHAR(100) | 100 | 是 |  |  |  |
|  | PAY\_STATUS | 支付状态 | SMALLINT(1) | 1 | 是 |  |  | 1.未支付2.支付中3.支付成功4.支付失败 |
|  | ADD\_TIME | 提交时间 | DATETIME |  | 是 |  |  |  |