# 1.数据库表描述

## 1.1 patrol

用于描述街道上巡逻的巡查员；每个巡查员会被分配一个巡逻区域，区域是事先划定好并存储在region表里面

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 键 | 是否为空 | 描述 |
| id | int | 主键 | NOT NULL | 自增id |
| name | varchar |  |  | 姓名 |
| department | varchar |  |  | 所属部门 |
| identity | enum |  |  | 身份类型（执法人员/协管人员） |
| title | varchar |  |  | 头衔职位 |
| telephone | varchar |  |  | 电话号 |
| wechat | varchar |  |  | 微信（暂时弃用） |
| related\_region | int | 外键：region->id |  | 该巡查员被分配的区域 |
| current\_region | int | 外键：region->id |  | 该巡查员当前所在的区域 |
| task | varchar |  |  | 上级分配任务的描述 |
| create\_time | datetime |  |  | 记录创建时间；自动生成 |
| update\_time | datetime |  |  | 记录更新时间；自动生成 |

## 1.2 patrol\_status

巡查员每天的状态，每条记录代表单个巡查员单独一天的状态

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 键 | 是否为空 | 描述 |
| id | int | 主键 | NOT NULL | 自增id |
| patrol\_id | int | 外键：patrol->id | NOT NULL | 对应巡查员 |
| date | date |  | NOT NULL | 记录的日期 |
| at\_work | tinyint |  |  | 是否在岗：0否1是 |
| vacation | tinyint |  |  | 是否休正假：0否1是 |
| vacation\_defer | tinyint |  |  | 是否补休：0否1是 |
| on\_work | datetime |  |  | 今日上班时间 |
| off\_work | datetime |  |  | 今日下班时间 |
| create\_time | datetime |  |  | 记录创建时间；自动生成 |
| update\_time | datetime |  |  | 记录更新时间；自动生成 |

## 1.3 patrol\_location

记录每个巡查员的坐标，实际使用时app端可能会隔一段时间提交一个坐标，需要将每个坐标都保存下来

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 键 | 是否为空 | 描述 |
| id | int | 主键 | NOT NULL | 自增id |
| patrol\_id | int | 外键：patrol->id |  | 对应巡查员 |
| location | json |  |  | 经纬度组成的字符串，例如：[104,209] |
| record\_time | datetime |  |  | 记录时间；自动生成 |

## 1.4 region

巡查区域（也叫围栏），由政府事先划定写死后存在数据库里；围栏是最小粒度的区域，此外共有13个街道，每个围栏都属于其中一个街道

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 键 | 是否为空 | 描述 |
| id | int | 主键 | NOT NULL | 自增id |
| name | varchar |  |  | 围栏名 |
| agency | varchar |  |  | 围栏所属街道 |
| creator | varchar |  |  | 创建者 |
| create\_time | datetime |  |  | 创建时间；自动生成 |
| point\_list | json |  |  | 多个经纬度列表组成的列表 |

## 1.5 bike

用于描述单车管理员；单车管理员和巡查员比较类似，相当于是简易版的巡查员

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 键 | 是否为空 | 描述 |
| id | int | 主键 | NOT NULL | 自增id |
| name | varchar |  |  | 姓名 |
| company | varchar |  |  | 所属公司：美团/青桔/哈啰 |
| telephone | varchar |  |  | 电话号 |
| wechat | varchar |  |  | 微信（暂时弃用） |
| related\_region | int | 外键：region->id |  | 该单车管理员被分配的区域 |
| current\_region | int | 外键：region->id |  | 该单车管理员当前所在的区域 |
| create\_time | datetime |  |  | 记录创建时间；自动生成 |
| update\_time | datetime |  |  | 记录更新时间；自动生成 |

## 1.6 bike\_location

用于记录单车管理员的坐标；类似于patrol\_location

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 键 | 是否为空 | 描述 |
| id | int | 主键 | NOT NULL | 自增id |
| bike\_id | int | 外键：bike->id |  | 对应单车管理员 |
| location | json |  |  | 经纬度组成的字符串，例如：[104,209] |
| record\_time | datetime |  |  | 记录时间；自动生成 |
| company | varchar |  |  | 所属公司：美团/青桔/哈啰 |

## 1.7 problem\_record

用于描述和记录巡查员在街道上巡逻时发现的问题

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 键 | 是否为空 | 描述 |
| id | int | 主键 | NOT NULL | 自增id |
| patrol\_id | int | 外键：patrol->id |  | 发布问题的巡查员 |
| type | varchar |  |  | 问题类型 |
| picture | varchar |  |  | 问题照片的URL |
| region | int | 外键：region>id |  | 发现问题处属于哪个围栏 |
| post\_date | datetime |  |  | 问题发布日期 |
| rectify\_date | datetime |  |  | 应完成整改的预定时间 |
| is\_rectified | tinyint |  |  | 是否整改：0否1是 |
| comment | varchar |  |  | 描述问题的相关备注 |
| create\_time | datetime |  |  | 记录创建时间；自动生成 |
| update\_time | datetime |  |  | 记录更新时间；自动生成 |

## 1.8 stranded\_msg

用于记录web端和app端通过websocket进行通信的通信内容

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 键 | 是否为空 | 描述 |
| id | int | 主键 | NOT NULL | 自增id |
| patrol\_telephone | varchar |  |  | app端用户的登录电话号 |
| admin\_telephone | varchar |  |  | web端用户的登录电话号，需加前缀admin |
| message | varchar |  |  | 消息内容 |
| is\_read | tinyint |  |  | 消息是否已读 |
| sender | varchar |  |  | 消息发送者（电话号） |
| record\_time | datetime |  |  | 记录时间；自动生成 |

# WebSocket接口

## app端

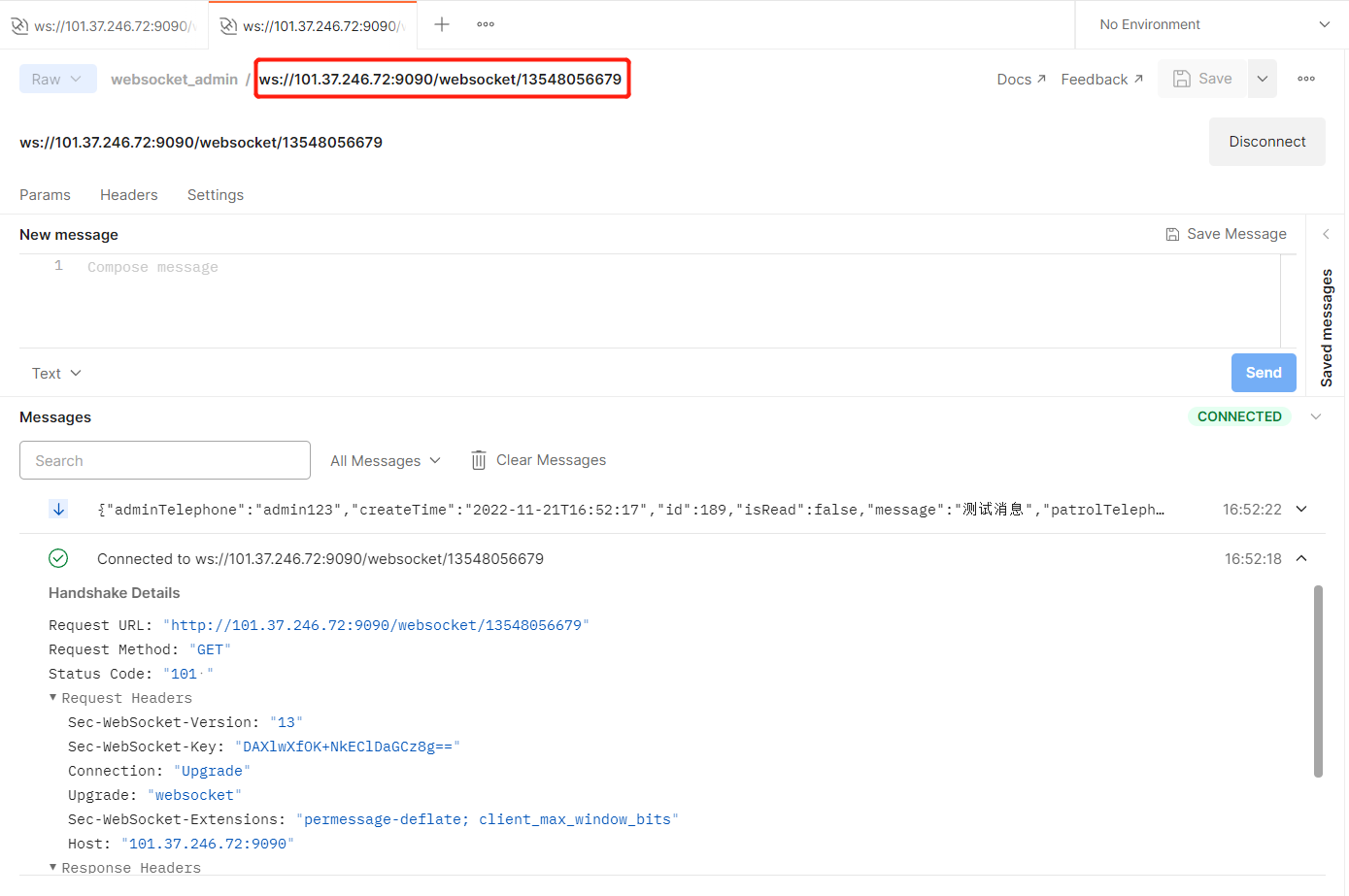
### 注册websocket服务

app端与后端建立websocket服务的URL为

**ws://101.37.246.72:9090/websocket/13548056679**

其中最后一项为app端的登陆账号（通常为电话号），此为服务端与app端的websocket通信依据。在postman中测试连接成功截图如下：

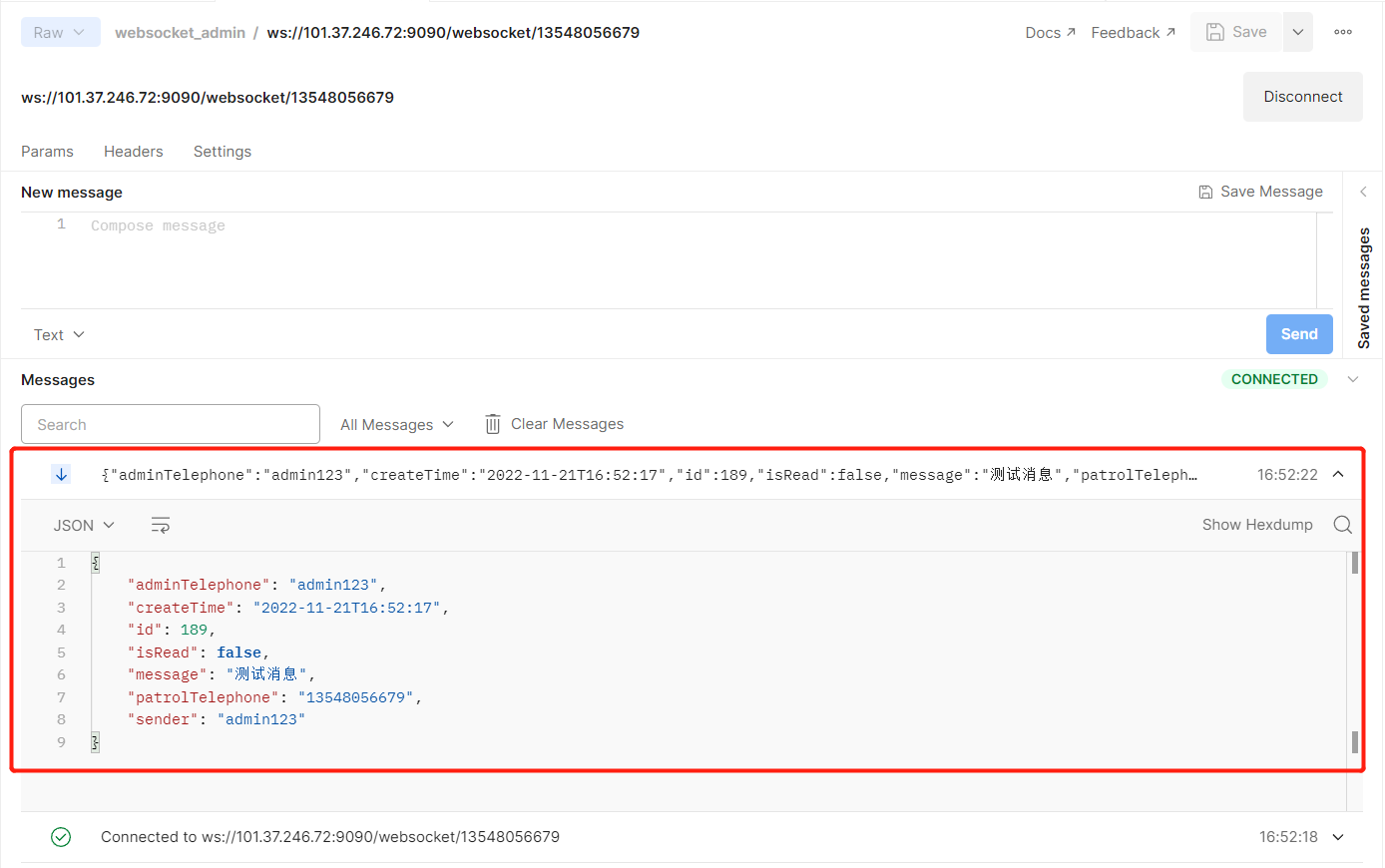
（在登录后就建立连接）



### 接收消息

在本项目目前的业务场景下，app端不需要主动和web端进行通信，只需要对web端的通信内容进行回复。app端接收到的websocket信息JSON格式为：

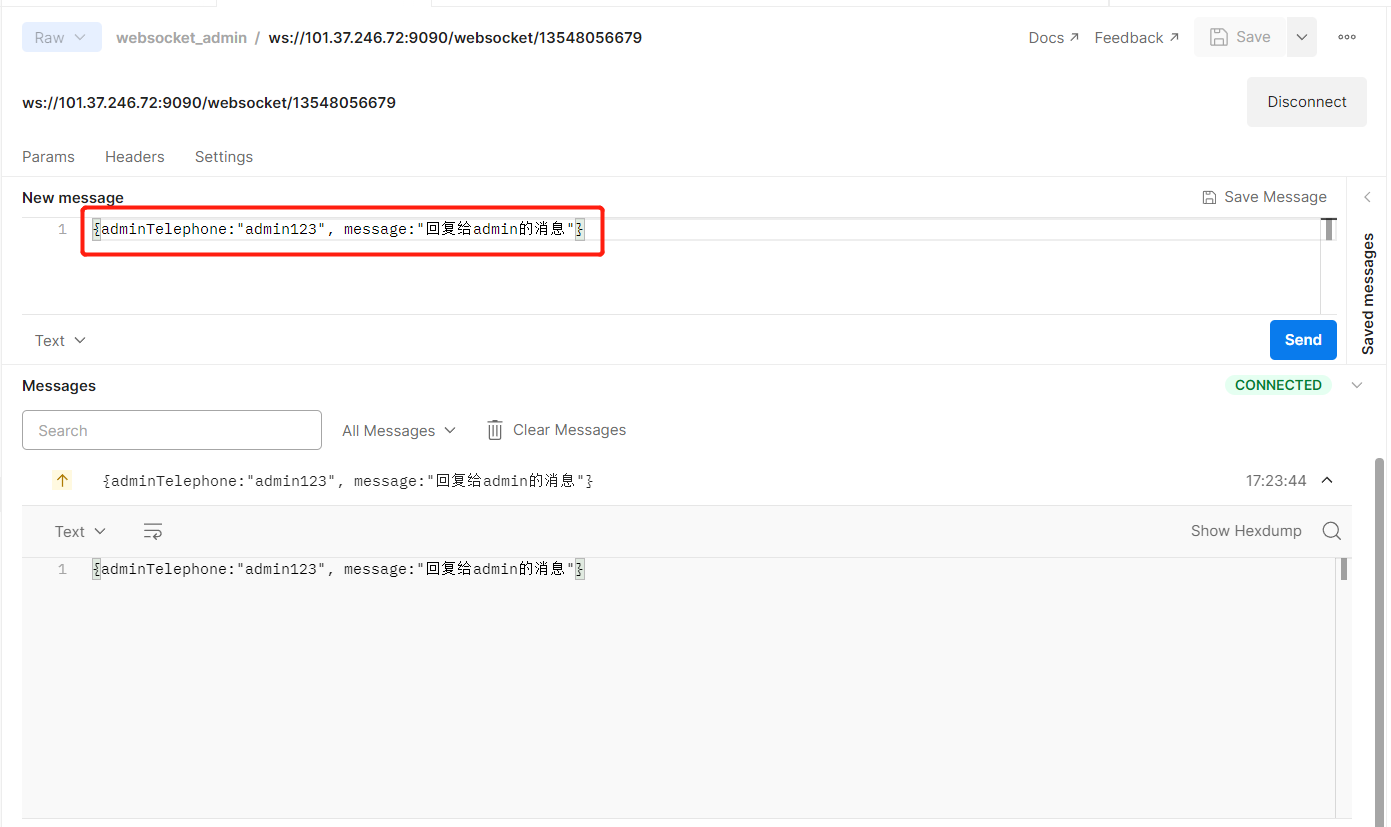
|  |  |
| --- | --- |
| 字段名 | 描述 |
| id | 自增id |
| patrolTelephone | app端用户的登录电话号 |
| adminTelephone | web端用户的登录电话号，需加前缀admin |
| message | 消息内容 |
| is\_read | 消息是否已读 |
| sender | 消息发送者（电话号） |
| record\_time | 记录时间；自动生成 |



### 发送消息

app端的回复消息只需要两个字段（如下图），adminTelephone字段指定要回复的web端管理员标识（以admin前缀开头），可以在接收到的消息中的adminTelephone获取；message字段即为回复的具体内容

|  |  |
| --- | --- |
| 字段名 | 描述 |
| adminTelephone | web端用户的登录电话号 |
| message | 消息内容 |



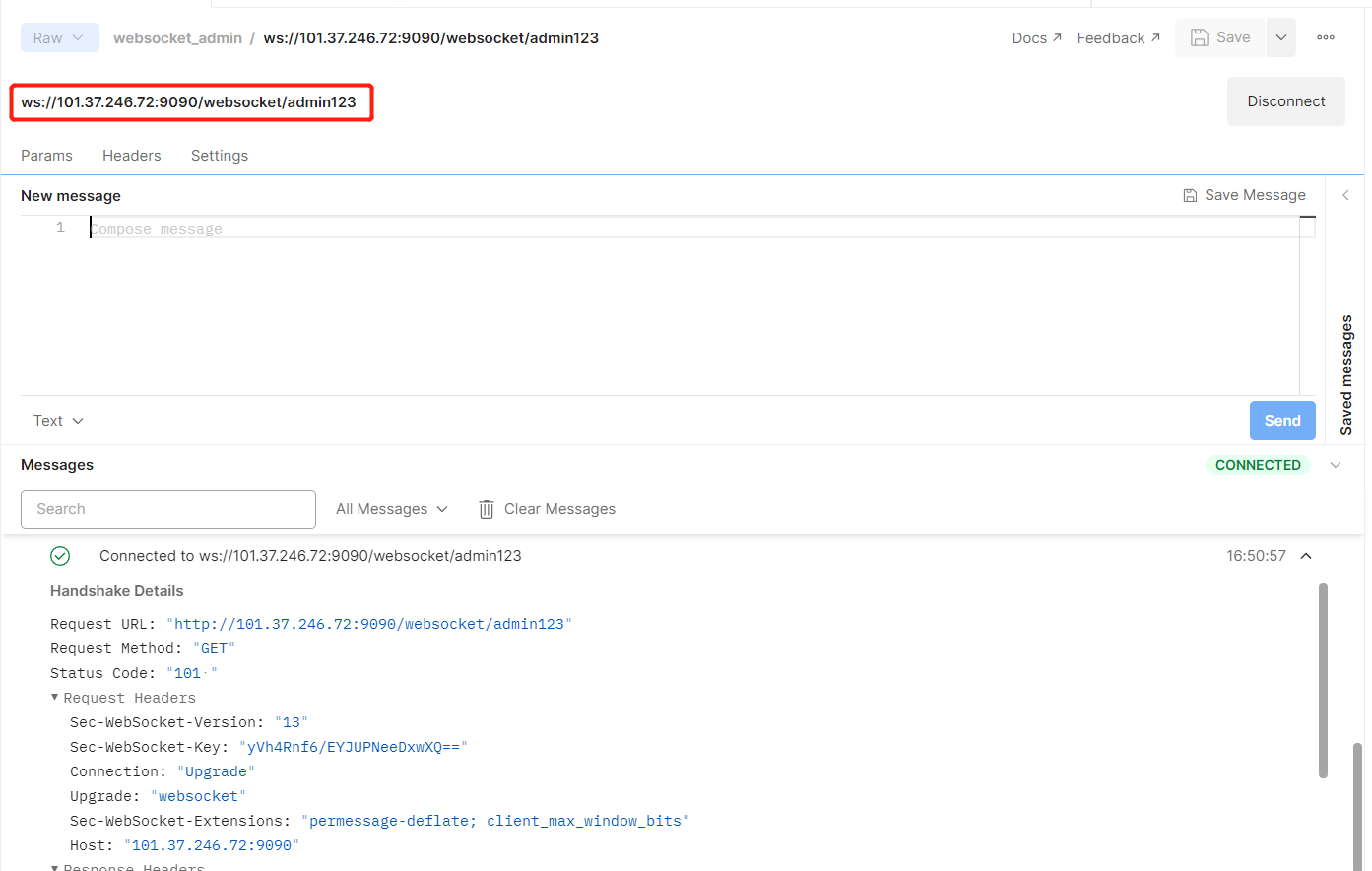
## web端

### 注册websocket服务

web端与后端建立websocket服务的URL为

**ws://101.37.246.72:9090/websocket/admin123**

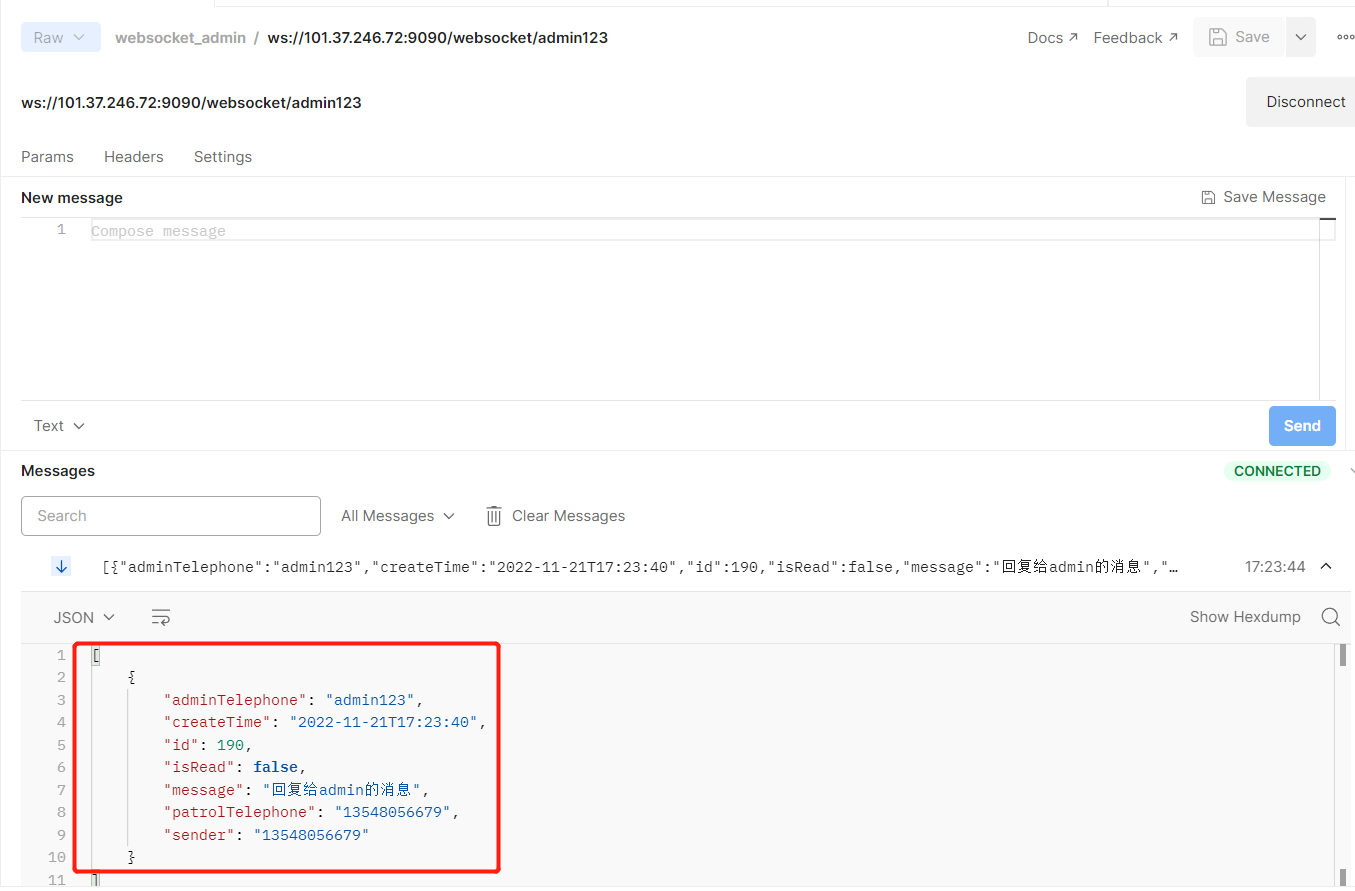
其中最后一项为web端的登陆账号加上admin前缀，此为服务端与web端的websocket通信依据。在postman中测试连接成功截图如下



### 接收消息

web端接收到的来自app端的消息为JSON数组（因为会有滞留消息重新发送），JSON消息格式为：

|  |  |
| --- | --- |
| 字段名 | 描述 |
| id | 自增id |
| patrolTelephone | app端用户的登录电话号 |
| adminTelephone | web端用户的登录电话号，需加前缀admin |
| message | 消息内容 |
| is\_read | 消息是否已读 |
| sender | 消息发送者（电话号） |
| record\_time | 记录时间；自动生成 |



### 发送消息

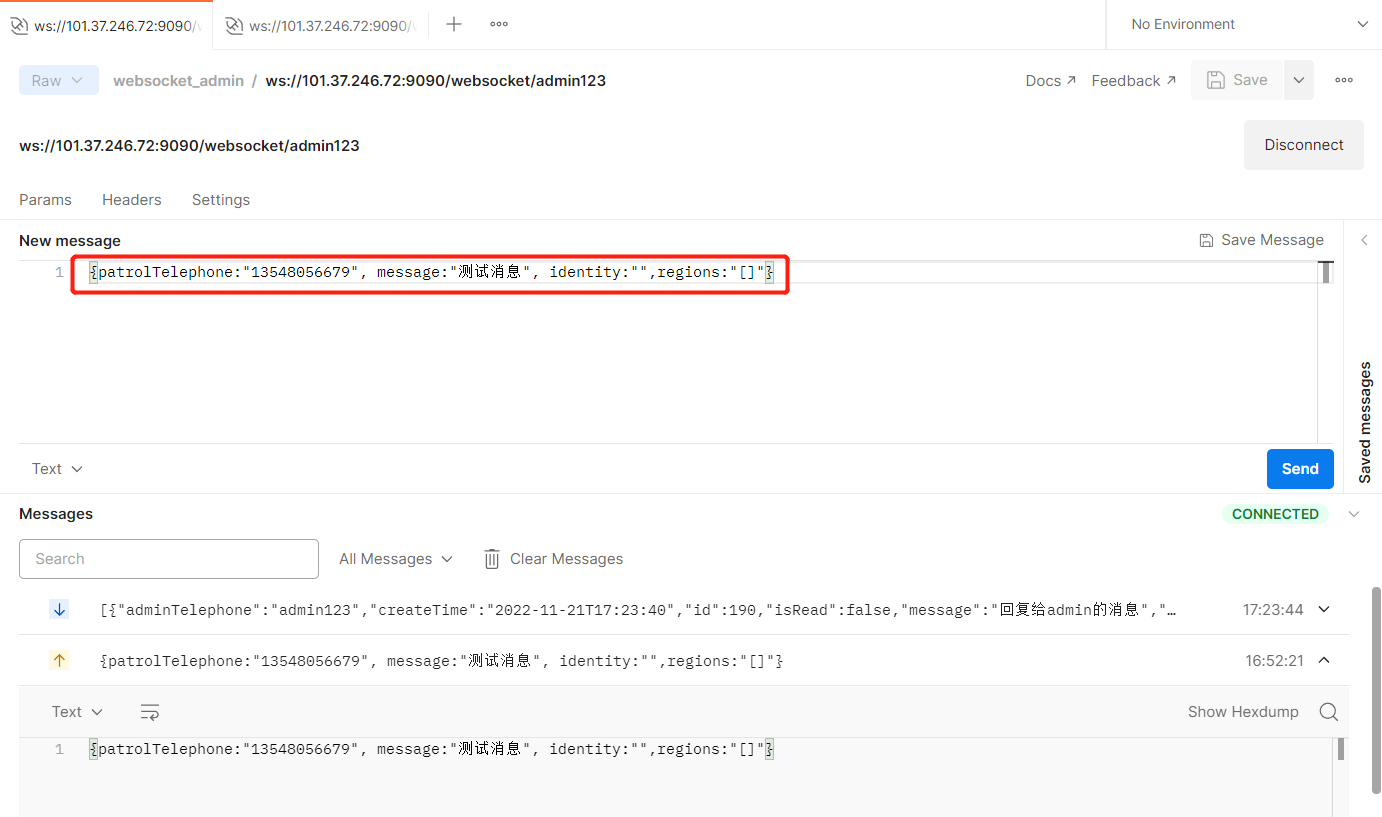
web端发送消息有两种模式，单独发送和群发；

1. 单独发送：当单独发送消息给某个巡查员时，只需指定patrolTelephone字段，idnetity和regions字段给空值和空列表（暂不可缺省）；
2. 群发：可以指定群发消息给某个类型的巡查员，也可以按照巡逻区域来群发消息，也可以同时给这两个字段赋值以便发送消息给指定区域内指定类型的巡查员。群发消息时，可以不指定patrolTepehone字段；

**群发消息最新的需求是按照街道来群发，街道名即为region表中的agency字段内容，因此查询region表中agency为指定街道名的全部围栏的id，将其封装为一个列表赋值给JSON中的regions字段**

web端发送消息给app端时，需指定以下字段

|  |  |
| --- | --- |
| 字段名 | 描述 |
| patrolTelephone | app端用户的登录电话号 |
| message | 消息内容 |
| identity | 人员类型：执法人员/协管人员 |
| regions | 围栏的列表：region表记录的id组成的列表 |



# HTTP接口

## 面向app端的接口

### PatrolLocationController

1. app端每隔一段时间上传巡查员的坐标，构造一个PatrolLocation实体并将其写入patrol\_location表；

**输入：**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 描述 |
| patrolTelephone | String | 巡查员的电话号码 |
| location | JSON | 巡查员的坐标，经纬度构成的字符串列表 |

**输出：**写入数据库patrol\_location表

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 描述 |
| patrolId | int | 巡查员表的id |
| location | JSON | 巡查员的坐标，经纬度构成的字符串列表 |

以上需求涉及到巡查员的电话号到巡查员的id之间的转换，需要先查询patrol表获取电话号对应的巡查员id

### PatrolStatusController

巡查员每天打卡上班时在patrol\_status表中创建一条记录，用来记录该巡查员这一天的状态；当他下班时，更新表中的off\_work字段为实际的下班时间；

1. 上班考勤打卡

接口名称：

PatrolStatusController/clockIn

接口描述：

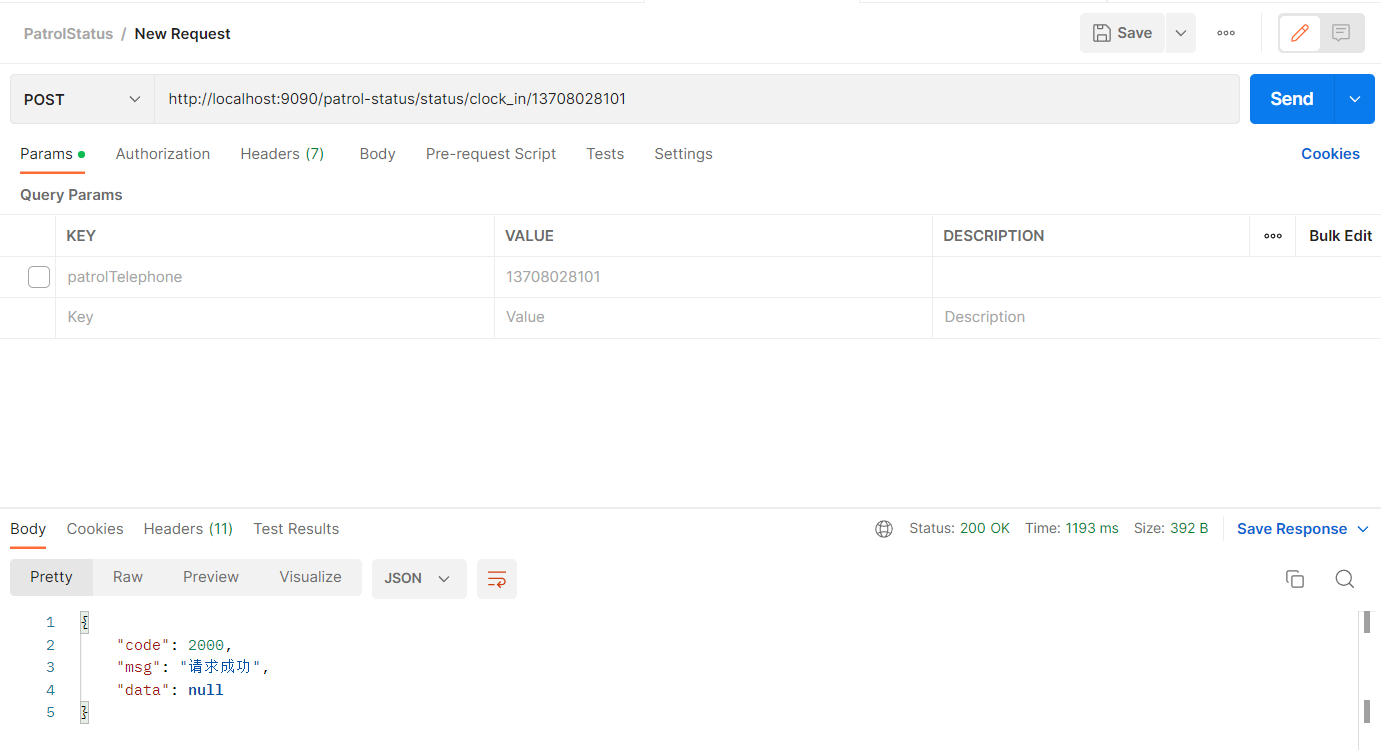
调用该接口时，以巡查员（以参数形式输入）和日期（自动生成当前日期）为索引在数据表patrol\_status中生成一条记录，并写入当前时间作为上班时间。

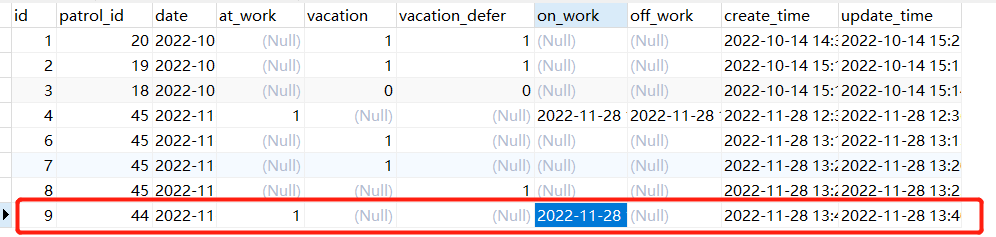
输入：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 描述 |
| patrolTelephone | String | 巡查员的登录电话号 |

输出：写入数据库

测试结果





1. 下班考勤打卡

接口名称：

PatrolStatusController/clockOut

接口描述：

调用该接口时，以巡查员（以参数形式输入）和日期（自动生成当前日期）为索引在数据表patrol\_status中查询记录，并将其中off\_work字段更新为当前时间。

输入输出：

同上

### ProblemRecordController

**问题记录相关接口需构思实现**

### PatrolController

1. 主要是patrol表中的current\_region字段的实时更新任务，目前的一个想法是，每当3.1.1中巡查员的坐标更新后，判断一下巡查员当前是否在自己所属的围栏，然后更新当前所处围栏信息;

## 面向web端的接口

### PatrolStatusController

1. 设置休假情况

接口名称：

PatrolStatusController/setVacation

接口描述：

指定巡查员、日期、和选项（休假/补休），若当日巡查员没有在岗，就将当日状态设为选项所指定的状态。（若数据表中当天该巡查员为在岗，则不会更新成功！）（同一天被设为休假后，无法再被设为补休）

输入：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 描述 |
| patrolTelephone | String | 巡查员的电话号 |
| date | String | 格式如：2022-11-28 |
| choice | String | 内容为：休假/补休 |

输出：

写入数据库状态

测试结果：

