浙江大学

本科实验报告

课程名称：	B/S 体系软件设计
姓　　名：	刘轩铭
学　　院：	计算机科学与技术学院
系：	计算机系
专　　业：	软件工程
学　　号：	3180106071
指导教师：	胡晓军

2021 年　05 月　01 日

IoT物联网设备管理APP详细设计

3180106071 刘轩铭 软件工程

版本记录

文档版本号	修订人	修订日期	变更内容	备注
1.0	刘轩铭	2021-03-17	新建	项目基础设计
1.1	刘轩铭	2021-03-31	修订	技术栈设计完成
1.2	刘轩铭	2021-04-13	修订	数据库和数据表设计完成
2.0	刘轩铭	2021-05-01	更新	对文档进行修缮和更新

需求概要分析

需求描述

物联网(IoT)正在积极塑造工业生产和消费世界，从零售到医疗保健，从金融到物流，智能技术已遍及每个业务和消费者领域。为了更好地将物联网和互联网进行结合，需要一种方式通过互联网对物联网设备进行管理，例如获取其提供的数据并对设备进行控制等。比较理想的方法是通过网页对物联网设备信息进行收集和展示。为了实现这一目标，本人计划开发一个基于B/S机构的Web APP应用，用于对物联网设备进行管理和数据分析。

该APP需要实现的需求如下：

1. 搭建一个MQTT服务器，能够接收指定的物联网终端模拟器发送的数据。
2. 实现用户注册、登录等功能，用户注册时需要填写必要的信息并验证，如用户名、密码要求在6字节以上，email的格式验证，并保证用户名和email在系统中唯一。用户登录后可以进行后续操作。
3. 提供设备配置界面，可以创建或修改设备信息，包含必要信息，如设备ID、设备名称等。
4. 提供设备上报数据的查询统计界面。
5. 提供地图界面展示设备信息，区分设备发出的正常和告警信息，并可以展示历史轨迹。

6. 首页提供信息统计功能（设备总量、在线总量、接收的数据量等），并以图表方式展示（柱状体、折线图等）数据信息。
7. Web APP样式适配手机端，能够在手机浏览器/微信等应用内置的浏览器中友好展示。

功能描述

该应用需要完成以下的功能点：

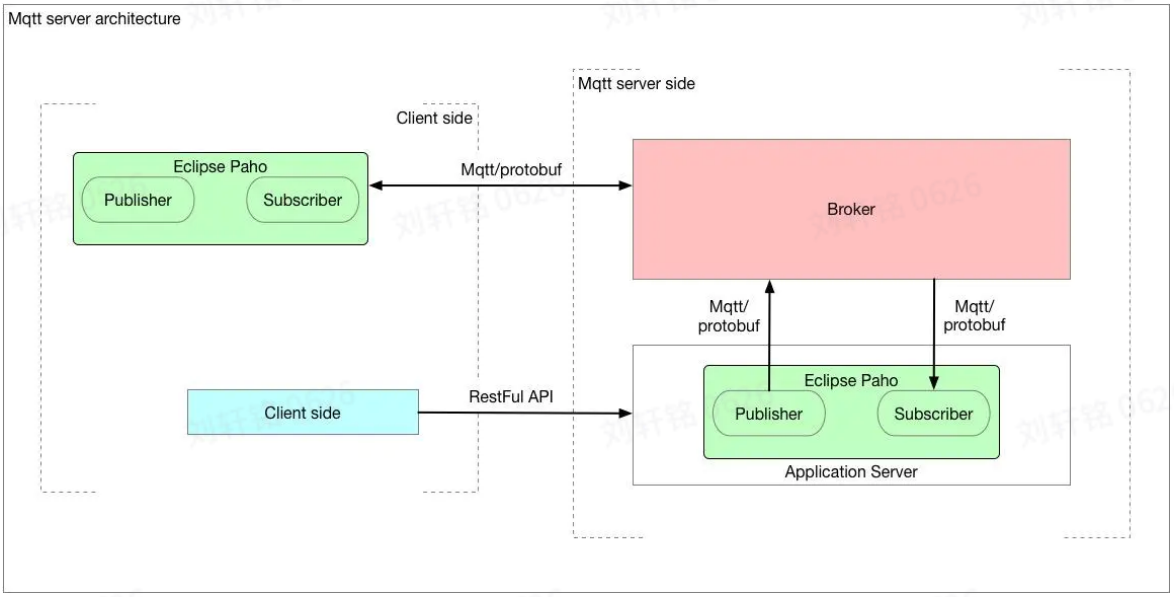
- 对于指定Server和Topic上MQTT消息的订阅功能
- 对于订阅获取到的消息进行储存和读取的功能
- 用户能在系统内获取特定账号并使用
- 用户能对物联网设备进行创建和编辑
- 用户能查看物联网设备发送的消息，并了解消息含义
- 用户能通过图表清晰地查看应用内物联网设备和消息的数量情况

设计概要分析

数据字典

- **MQTT**：MQTT（Message Queuing Telemetry Transport，消息队列遥测传输协议），是一种基于发布/订阅（publish/subscribe）模式的"轻量级"通讯协议，该协议构建于TCP/IP协议上。是一个基于客户端-服务器的消息发布/订阅传输协议。协议是轻量、简单、开放和易于实现的，这些特点使它适用范围非常广泛。在很多情况下，包括受限的环境中，如：机器与机器（M2M）通信和物联网（IoT）。其在，通过卫星链路通信传感器、偶尔拨号的医疗设备、智能家居、及一些小型化设备中已广泛使用。
- **数据库**：数据库是“按照数据结构来组织、存储和管理数据的仓库”。是一个长期存储在计算机内的、有组织的、可共享的、统一管理的大量数据的集合。一般分为关系型数据库和非关系型数据库。本应用采用MySQL关系型数据库进行数据储存和查取功能。

整体设计架构



本应用整体分为服务端和客户端两个部分。整体设计图如上所示。下面分别介绍两个部分的架构情况。

Server Side

根据以上架构图，分别设计如下的模块：

- **Broker**：MQTT消息管理服务器，类似消息 Pub-Sub 队列，用于物联网设备发出的消息接收和发布功能。这是服务端的核心部分。
- **Application Server**：用于处理RESTful的请求，对用户在Web端发送的请求进行响应。同时包装Pubisher和Subscriber，处理和转发MQTT消息。这也是服务端的核心部分。
- **Publisher**：本质是MQTT Client，用于发布Server模块的消息到Broker，本应用中由于没有这一部分的需求，故不涉及。
- **Subscriber**：本质是MQTT Client，用于从Broker订阅Client端消息，本应用中会将其作为一个模块集成到Server中，即在Server中单独设计线程对MQTT消息进行订阅和储存。

Client Side

根据以上架构图，分别设计如下的模块：

- **Publisher**：用于发布MQTT消息到Broker，本应用中用简单的Client进行模拟。
- **Subscriber**：用于从Broker订阅Server端的消息，本应用中由于没有这一部分的需求，故不涉及。
- **Client**：用于发送RESTful请求给Application Server触发消息Pub/Sub，本应用中即Web端用户的使用

技术栈设计

服务端(S)设计

根据以上架构设计，服务端主要涉及到MQTT服务器的搭建以及Web服务端的搭建两个部分。另外还涉及到使用数据库对MQTT消息以及用户和设备信息等进行储存和管理。

MQTT Broker

在MQTT服务器的选择上，本应用选择Mosquitto作为Broker。

Mosquitto是一款实现了消息推送协议 MQTT 的开源消息代理软件，提供轻量级的，支持可发布/可订阅的的消息推送模式，使设备对设备之间的短消息通信变得简单，比如现在应用广泛的低功耗传感器，手机、微型控制器等移动设备都应用了这一软件。该应用的特点是兼容性好，消息质量高，且可以被集成到Spring MVC框架中。

为了搭建该服务器，首先需要下载和安装该软件，然后进行配置。

之后在服务端内部架设线程，不断订阅指定的TOPIC，获取相应数据并进行处理。

Web 服务端搭建

本应用使用Java框架SpringBoot作为服务端，通过Maven集成MQTT消息收发框架来对Broker进行订阅。同时Spring MVC模式可以方便的提供RESTful的API给客户端进行使用。此外，通过集成Java框架MyBatis可以方便的对数据库进行管理。

数据库

由于本应用涉及到的数据量不大，而且数据格式比较确定，比较容易绘制E-R图等特点，选择关系型数据库（RDS）是一个较好的选择。为了简单起见，本应用选择MySQL进行常规数据储存。

另外，为了在用户访问网站时能够对用户信息进行鉴权，选择Redis进行用户登录Token的存放。

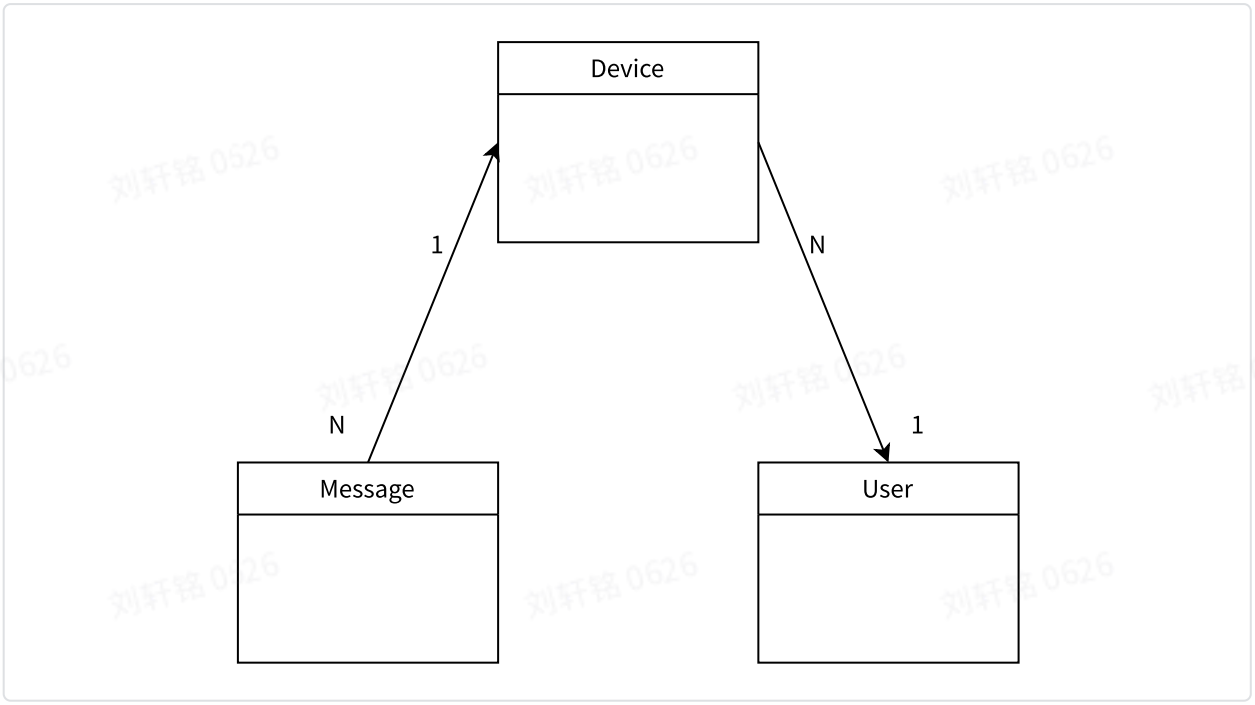
客户端(B)设计

根据以上架构设计，客户端是传统的Web形式。由于该网页可能有多页面存在，故选择React作为前端使用的技术框架。

数据模型设计

- 说明：部分字段在数据库建表时和设计中有命名上区别

E-R图设计



数据表设计

用户基本信息(t_user_info)：

	A	B	C	D
1	**中文名称**	**字段名**	**字段类型**	**字段描述**
2	用户ID	id	Integer	系统自动生成，主码
3	用户名	name	String	
4	用户密码	password	String	
5	邮箱	email	String	
6	用户头像	avatar	String	

物联网设备基本信息(t_project_state):

	A	B	C	D
1	**中文名称**	**字段名**	**字段类型**	**字段描述**
2	ID	id	Integer	系统自动生成，主码
3	设备编码	code	String	设备辨识符
4	设备名	name	String	可以被修改
5	描述	description	String	
6	创建时间	create_time	Datetime	
7	创建人	create_by	String	
8	最后修改时间	update_time	Datetime	系统自动更新
9	设备类型	type	Integer	枚举值

MQTT消息信息(t_device_message):

	A	B	C	D
1	**中文名称**	**字段名**	**字段类型**	**字段描述**
2	ID	id	Integer	系统自动生成，主码
3	设备编码	client	String	设备辨识符
4	传输信息	info	String	可以被修改
5	传输值	value	Integer	
6	是否报警	alert	Integer	布尔值
7	经度	longitude	Double	
8	纬度	latitude	Double	系统自动更新
9	时间戳	timestamp	Datetime	枚举值

建表SQL：

SQL

```sql
CREATE TABLE `t_user_info` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(128) NOT NULL DEFAULT '' COMMENT '名字',
  `password` varchar(128) NOT NULL DEFAULT '' COMMENT '密码',
  `email` varchar(128) NOT NULL DEFAULT '' COMMENT '邮箱',
  `avatar` varchar(128) DEFAULT '' COMMENT '头像URL',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT '用户基本信息';

CREATE TABLE `t_device_info` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `code` varchar(128) NOT NULL DEFAULT '' COMMENT '设备编号',
  `name` varchar(128) NOT NULL DEFAULT '未命名' COMMENT '设备名称',
  `description` text COMMENT '设备描述',
  `create_time` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `create_by` VARCHAR(128) NOT NULL DEFAULT '未分配' COMMENT '用户ID',
  `update_time` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '更新时间',
  `type` tinyint(4) NOT NULL DEFAULT 0 COMMENT '设备类型',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT '设备基本信息';

CREATE TABLE `t_device_message` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `client` varchar(128) NOT NULL DEFAULT '' COMMENT '设备编号',
  `info` varchar(128) NOT NULL DEFAULT '' COMMENT '数据信息',
  `value` int(11) NOT NULL DEFAULT 0 COMMENT '设备数据值',
  `alert` int(11) NOT NULL DEFAULT 0 COMMENT '是否报警',
  `longitude` DOUBLE(12,8) NOT NULL DEFAULT 0 COMMENT '经度',
  `latitude` DOUBLE(12,8) NOT NULL DEFAULT 0 COMMENT '纬度',
  `timestamp` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '时间戳',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT '传输数据信息';
```

接口设计

接口指的是客户端能够向服务端请求的数据请求API，本应用中均设计成RESTful风格的API。

由应用本身设计，主要分为用户信息模块和设备信息两个模块。

其中，需要的HTTP请求方法类型已经给出，需要的参数以Json形式传递并被Spring MVC对象化，返回结果也将以Json形式进行传递。

用户模块

```
/**
 * @description: user login
 * @author: liuxuanming
 * @date: 2021/3/28 9:23 上午
 * @params: []
 * @return: void
 */
@RequestMapping(value = "/login", method = {RequestMethod.POST})
public RpcResult<UserInfoDTO> login(@RequestBody(required = false) UserLoginQuery query) {
    return userLoginManager.loginValidation(query);
}

/**
 * @description: callback page of unauthorization.
 * @author: liuxuanming
 * @date: 2021/3/31 9:37 下午
 * @params: []
 * @return: com.hinsliu.iotapp.domain.RpcResult<java.lang.String>
 */
@RequestMapping(value = "/callback")
public RpcResult<String> callback() { return userLoginManager.unauthorizedCallback(); }

/**
 * @description: user register with form.
 * @author: liuxuanming
 * @date: 2021/3/31 9:37 下午
 * @params: [query]
 * @return: com.hinsliu.iotapp.domain.RpcResult<com.hinsliu.iotapp.domain.view.common.UserInfoDTO>
 */
@RequestMapping(value =  "/register", method = {RequestMethod.POST})
public RpcResult<UserInfoDTO> register(@RequestBody(required = false) UserRegisterForm query) {
    return userLoginManager.register(query);
}
```

设备信息模块

```java
@AuthToken
@RequestMapping(value = "/list", method = {RequestMethod.POST})
public RpcResult<Page<DeviceInfoDTO>> getDeviceList(@RequestBody(required = false) DeviceInfoQuery query) {
    return deviceManager.queryByPage(query);
}

@AuthToken
@RequestMapping(value = "/message", method = {RequestMethod.POST})
public RpcResult<Page<DeviceMessageDTO>> getDeviceMessage(@RequestBody(required = false) DeviceMessageQuery query) {
    return deviceManager.queryMessage(query);
}

@AuthToken
@RequestMapping(value = "/edit", method = {RequestMethod.POST})
public RpcResult<String> editDevice(@RequestBody(required = false) DeviceUpdateQuery query) {
    return deviceManager.editDevice(query);
}

@AuthToken
@RequestMapping(value = "/create", method = {RequestMethod.POST})
public RpcResult<String> createDevice(@RequestBody(required = false) DeviceUpdateQuery query) {
    return deviceManager.createDevice(query);
}
```

开发排期和时间轴

开发排期

- 服务器搭建：10天
- 数据库设计：2天
- 服务端设计和编写：10天
- 服务端测试：3天
- 客户端设计和编写：10天
- 应用线下联调测试：3天
- 系统上线测试和维护：5天

时间轴

- [x] ~~2021/03.22 - 2021.04.01 项目技术架构完成~~
- [x] ~~2021.04.01 - 2021.04.30 服务器搭建和服务端接口编写、数据库设计和搭建~~
- [ ] 2021.05.01 - 2021.06.01 客户端编写
- [ ] 2021.06.01 - 2021.06.15 整体测试和用户手册等文档撰