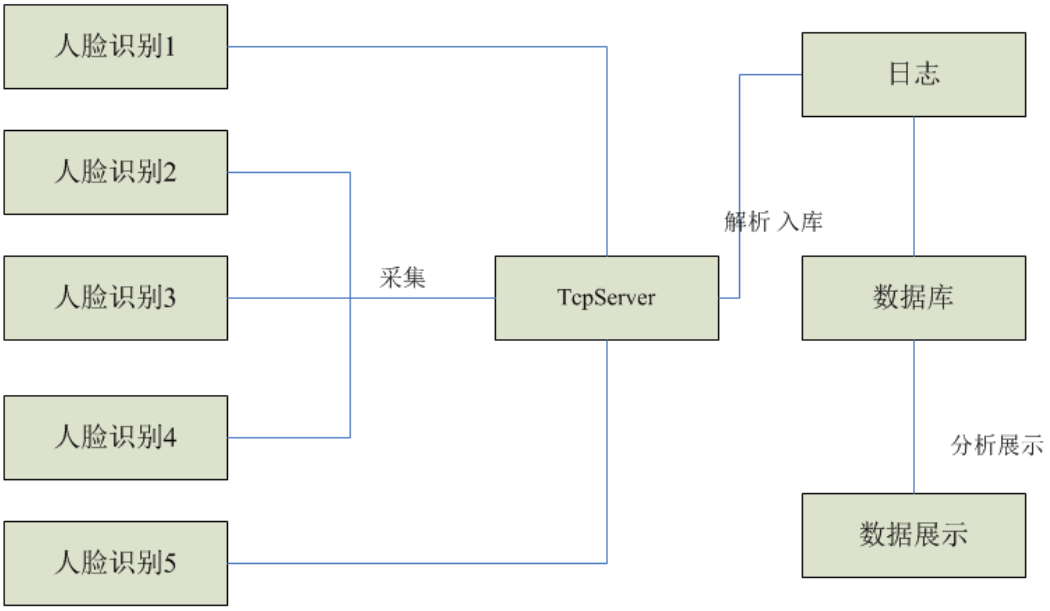
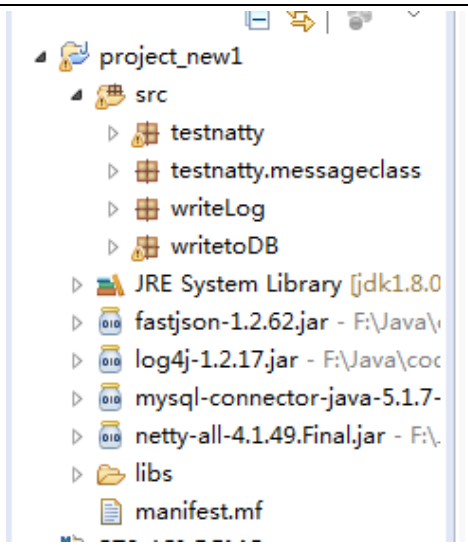
0 项目总体框架

(1)项目整体框架

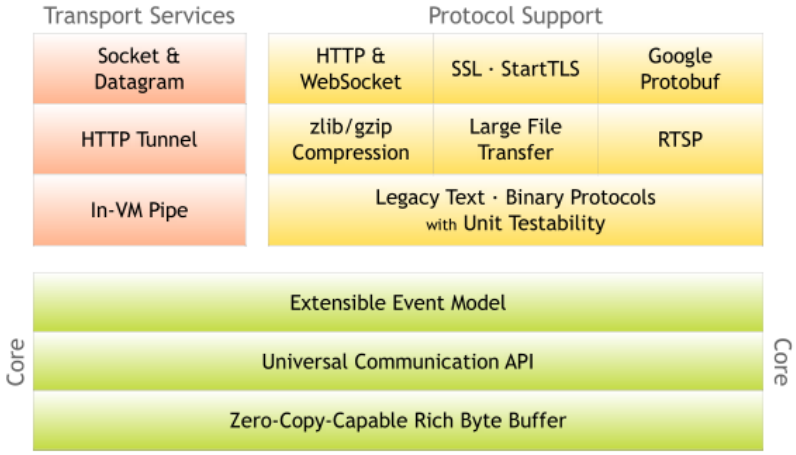


(2)具体实现框架

	Testnatty :Data Collection 数据收集 **testnatty.messageclass:Data Message 报文识别** writeLog:　Log Management 日志管理 writetoDB：Database operation 数据库操作(Mysql ,Sqlserver)

1 数据采集

(3)Netty 框架



Netty is an asynchronous event-driven network application framework. for rapid development of maintainable high performance protocol servers & clients.

(4)具体实现

1 创建线两个 event
一个是用于处理服务器端接收客户端连接的
一个是进行网络通信的（网络读写的）

```
EventLoopGroup pGroup = new NioEventLoopGroup();
EventLoopGroup cGroup = new NioEventLoopGroup();
```

2 创建辅助工具类，用于服务器通道的一系列配置

```
// 2 创建辅助工具类，用于服务器通道的一系列配置
ServerBootstrap b = new ServerBootstrap();
b.group(pGroup, cGroup)        // 绑定两个线程组
.channel(NioServerSocketChannel.class)        // 指定NIO的模式
.option(ChannelOption.SO_BACKLOG, 1024)        // 设置tcp缓冲区
.option(ChannelOption.SO_SNDBUF, 32*1024)    // 设置发送缓冲大小
.option(ChannelOption.SO_RCVBUF, 32*1024)    // 这是接收缓冲大小
.option(ChannelOption.SO_KEEPALIVE, true)    // 保持连接
.childHandler(new ChannelInitializer<SocketChannel>() {
    @Override
    protected void initChannel(SocketChannel sc) throws Exception {
        // 3 在这里配置具体数据接收方法的处理
        sc.pipeline().addLast(new ServerHandler());
    }
});
```

3 在这里配置具体数据接收方法的处理
4 端口进行绑定

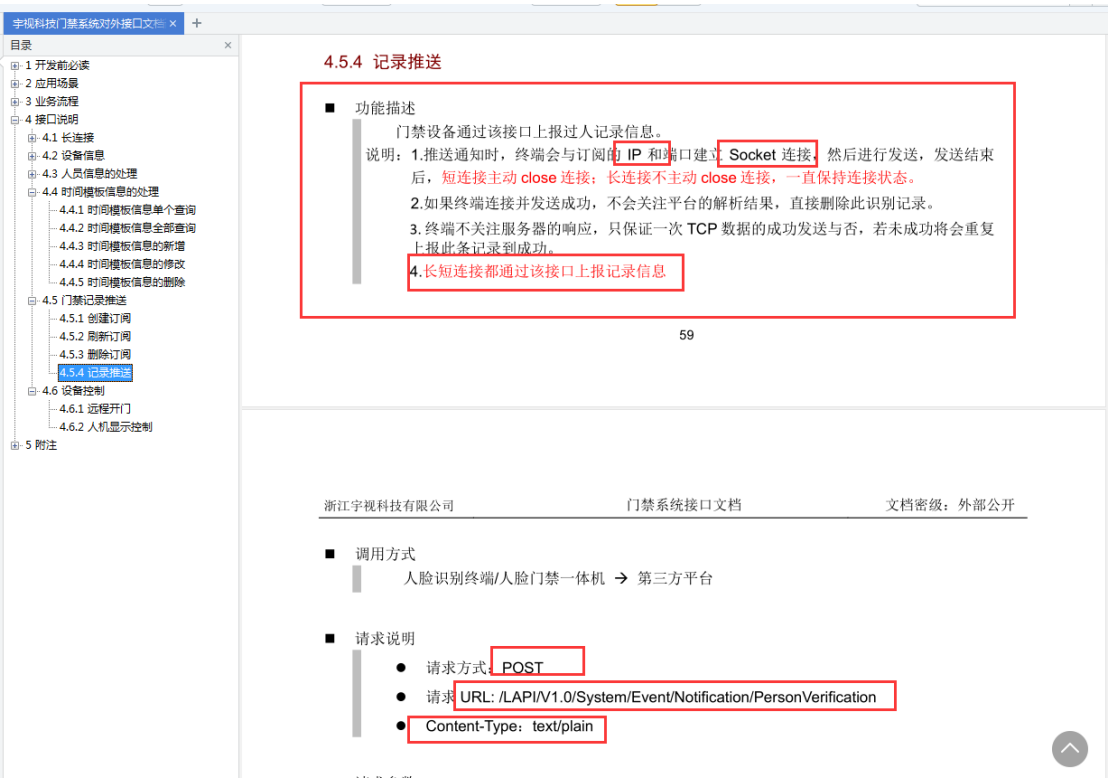
```java
// 4 进行绑定
ChannelFuture cf1;
try {
    cf1 = b.bind(8099).sync();
    cf1.channel().closeFuture().sync();
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
pGroup.shutdownGracefully();
cGroup.shutdownGracefully();
```

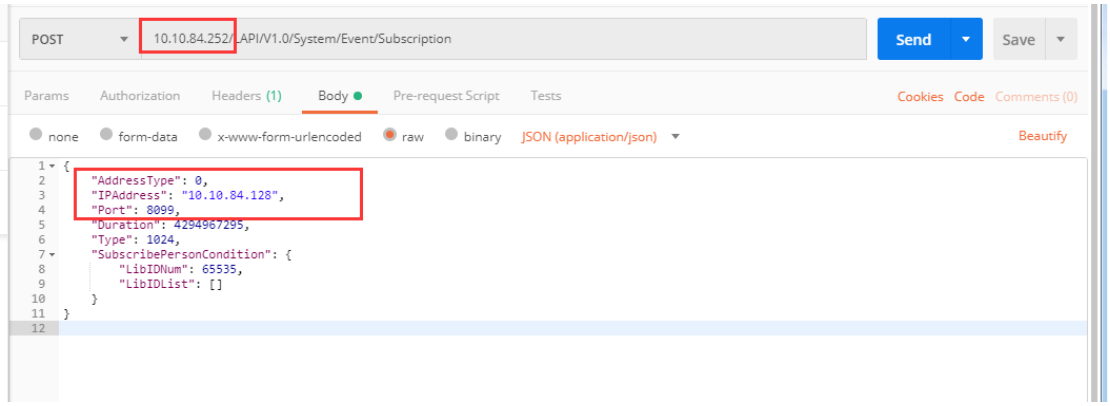
服务器端口设置为 8099

2 报文分析

(1)报文分析依据

根据《宇视科技门禁系统对外接口文档说明》





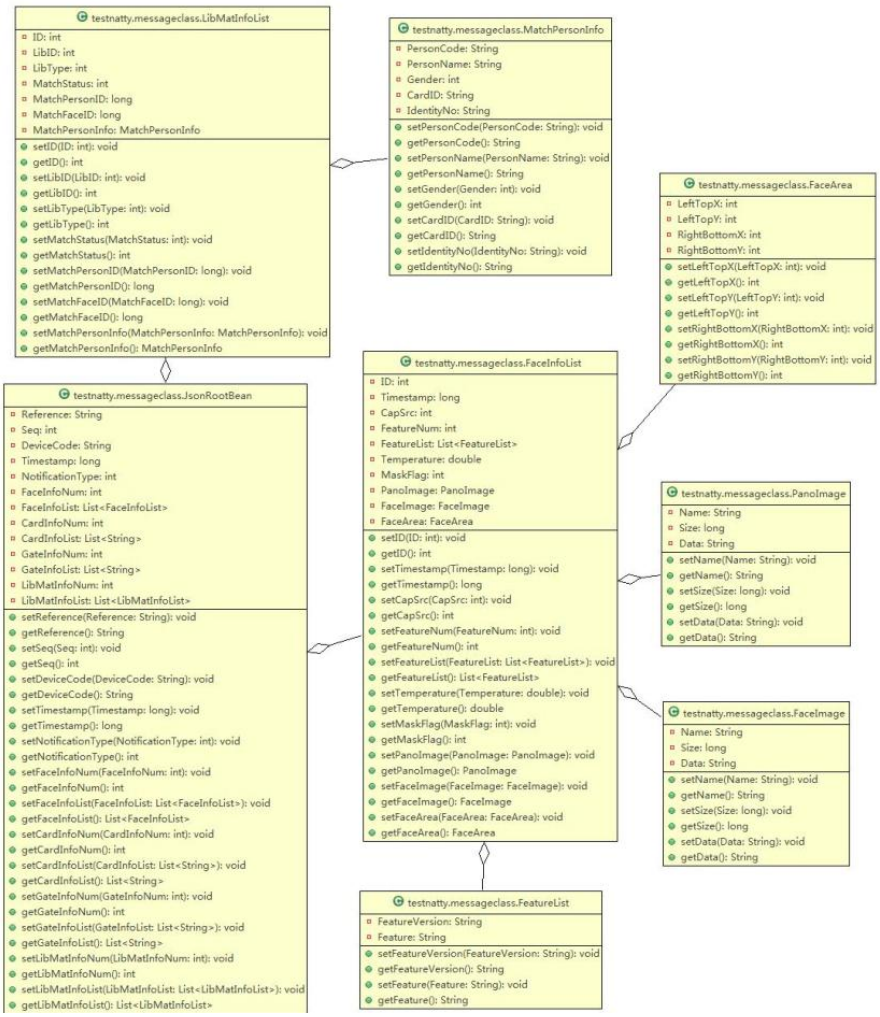
创建订阅->打开服务器及其接收端口->记录推送 ->删除订阅

(2)报文内容

{"Reference": "172.19.29.192:80/Subscription/Subscribers/1000","Seq": 7,"DeviceCode": "210235C4C53203011538","Timestamp": 1587973902,"NotificationType": 1,"FaceInfoNum": 1,"FaceInfoList": [{"ID": 7,"Timestamp": 1587426786,"CapSrc": 1,"FeatureNum": 0,"FeatureList": [{ "FeatureVersion": "", "Feature": "" }, { "FeatureVersion": "", "Feature": "" }],"Temperature": 35.9,"MaskFlag": 1,"PanoImage": { "Name": "1587426786_1_98.jpg", "Size": 172340, "Data": "" },"FaceImage": { "Name": "1587426786_2_98.jpg", "Size": 125776, "Data": "" },"FaceArea": { "LeftTopX": 3703, "LeftTopY": 5880, "RightBottomX": 5833, "RightBottomY": 7046 }}],"CardInfoNum": 0,"CardInfoList": [],"GateInfoNum": 0,"GateInfoList": [],"LibMatInfoNum": 1,"LibMatInfoList": [{"ID": 7,"LibID": 3,"LibType": 3,"MatchStatus": 10,"MatchPersonID": 317144637,"MatchFaceID": 317144637,"MatchPersonInfo": { "PersonCode": "0", "PersonName": "甘欣旺", "Gender": 0, "CardID": "1801213136", "IdentityNo": "" }}]}

(3) 报文结构

构造报文，报文格式如下图：




```
jrb= JSON.parseObject(JSON.parse((newMessage1)).toString(), testnatty.messageclass.JsonRootBean.class);
```

3 日志

(1)Log4j 介绍

采用 Apache logging services

The Apache Logging Services Project creates and maintains open-source software related to the logging of application behavior and released at no charge to the public.

(2) 具体操作

```
logger.warn(jrb.getReference());
logger.warn(jrb.getLibMatInfoList().get(0).getMatchPersonInfo().getCardID());
logger.warn(jrb.getLibMatInfoList().get(0).getMatchPersonInfo().getPersonName());
logger.warn(jrb.getTimestamp());
logger.warn(jrb.getFaceInfoList().get(0).getTimestamp());
logger.warn(jrb.getDeviceCode());
logger.warn(jrb.getFaceInfoList().get(0).getTemperature());
```

```
----+----1----+----2----+----3----+----4----+----5
日志级别: WARN
10.10.87.181:80/Subscription/Subscribers/1000
2009010219
贾海天
1588228620
1588228619
210235C4C53203010994
36.3
```

注意：

Timestamp	M	unsigned long	通知上报时间，UTC 格式，单位秒	1510925018
Timestamp	O	unsigned long	采集时间，UTC 格式，单位秒	1510925018

时间戳格式均为 UTC 格式

4 数据库操作(Mysql)

(1) 表结构

名	类型	长度	小数点	允许空值	
ID	int	11	0	☐	1
DeviceCode	varchar	20	0	☑	
BuildName	varchar	20	0	☑	

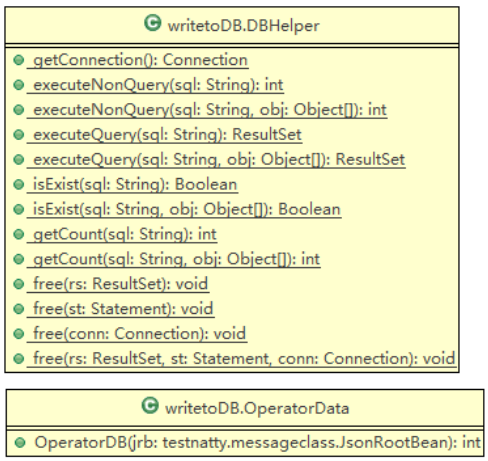
ID	DeviceCode	BuildName
1	210235C4C53203010994	3211

名	类型	长度	小数点	允许空值	
ID	int	11	0	☐	1
Reference1	varchar	100	0	☑	
CardID	varchar	10	0	☑	
PersonName	varchar	20	0	☑	
Timestamp1	bigint	20	0	☑	
Timestamp2	bigint	20	0	☑	
DeviceCode	varchar	20	0	☑	
Temperature1	double	0	0	☑	

ID	Reference1	CardID	PersonName	Timestamp1	Timestamp2	DeviceCode	Temperature1
1	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	1588580000	1588579999	210235C4C53203(	36.4
2	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	1588580671	1588580664	210235C4C53203(	36.5
3	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	1588580683	1588580682	210235C4C53203(	36.4
4	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	1588640528	1588640412	210235C4C53203(	34.1
5	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	1588640535	1588638354	210235C4C53203(	31.7
6	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	1588640542	1588637360	210235C4C53203(	31.6
7	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	1588640545	1588637356	210235C4C53203(	31.3

(2) JDBC 操作数据

数据库驱动 com.mysql.jdbc.Driver
数据库连接 jdbc:MySQL://IP:3306/hive?useUnicode=true&characterEncoding=utf-8
数据库操作类如下图所示



具体操作关键代码如下所示：

```
public class OperatorData {
    public int OperatorDB(testnatty.messageclass.JsonRootBean jrb)
    {
        //insert into JsonRootBean (Reference1,CardID,PersonName,Timestamp1,Timestamp2,DeviceCode,Temperature1)
         //Values ('10.10.84.252:80/Subscription/Subscribers/1000','2009010219','贾海天',1588580000,1588579999,'210235C4C53203010994',3
         String sql2="insert into JsonRootBean (Reference1,CardID,PersonName,Timestamp1,Timestamp2,DeviceCode,Temperature1)   Values
         //sql2="";
         Object[] obj = new Object[]{jrb.getReference(),jrb.getLibMatInfoList().get(0).getMatchPersonInfo().getCardID(),jrb.getLibMat
         //System.out.println(DBHelper.executeNonQuery(sql2,obj));
         //System.out.println("数据插入完成！");
         //return 1;
         return DBHelper.executeNonQuery(sql2,obj);
    }
}
```

5 数据库操作(Sqlserver)

(1)表结构

```sql
/****** Object:  Table [dbo].[buildinfo]    Script Date: 05/15/2020 11:55:27 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[buildinfo](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [DeviceCode] [nvarchar](20) NULL,
    [BuildName] [nvarchar](50) NULL,
 CONSTRAINT [PK_buildinfo] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[jsonrootbean]    Script Date: 05/15/2020 11:56:04 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[jsonrootbean](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [Reference1] [nvarchar](100) NULL,
    [CardID] [nvarchar](20) NULL,
    [PersonName] [nvarchar](20) NULL,
    [Timestamp1] [bigint] NULL,
    [Timestamp2] [bigint] NULL,
    [DeviceCode] [nvarchar](20) NULL,
    [Temperature1] [real] NULL,
 CONSTRAINT [PK_jsonrootbean] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

(2)JDBC 操作数据

数据库驱动 com.microsoft.sqlserver.jdbc.SQLServerDriver

数据库连接 jdbc:sqlserver://127.0.0.1:1433;databaseName=master;

数据库操作类如下图所示

```
writetoDBSqlserver.OperatorData
  OperatorDB(jrb: testnatty.messageclass.JsonRootBean): int
```

```
writetoDBSqlserver.DBHelper
  driver: String
  url: String
  user: String
  password: String
  DBHelper()
  getConnection(): Connection
  getStatement(): Statement
  getStatement(conn: Connection): Statement
  getPreparedStatement(cmdText: String, cmdParams: Object[]): PreparedStatement
  getPreparedStatement(conn: Connection, cmdText: String, cmdParams: Object[]): PreparedStatement
  ExecSql(cmdText: String): int
  ExecSql(conn: Connection, cmdText: String): int
  ExecSql(cmdText: String, cmdParams: Object[]): int
  ExecSql(conn: Connection, cmdText: String, cmdParams: Object[]): int
  ExecScalar(cmdText: String): Object
  ExecScalar(conn: Connection, cmdText: String): Object
  ExecScalar(cmdText: String, cmdParams: Object[]): Object
  ExecScalar(conn: Connection, cmdText: String, cmdParams: Object[]): Object
  getResultSet(cmdText: String): ResultSet
  getResultSet(conn: Connection, cmdText: String): ResultSet
  getResultSet(cmdText: String, cmdParams: Object[]): ResultSet
  getResultSet(conn: Connection, cmdText: String, cmdParams: Object[]): ResultSet
  buildScalar(rs: ResultSet): Object
  buildTableModel(rs: ResultSet): DefaultTableModel
  getTableModel(conn: Connection, cmdText: String): DefaultTableModel
  getTableModel(cmdText: String, cmdParams: Object[]): DefaultTableModel
  getTableModel(cmdText: String): DefaultTableModel
  getTableModel(conn: Connection, cmdText: String, cmdParams: Object[]): DefaultTableModel
  close(obj: Object): void
  closeEx(obj: Object): void
  closeConnection(obj: Object): void
```

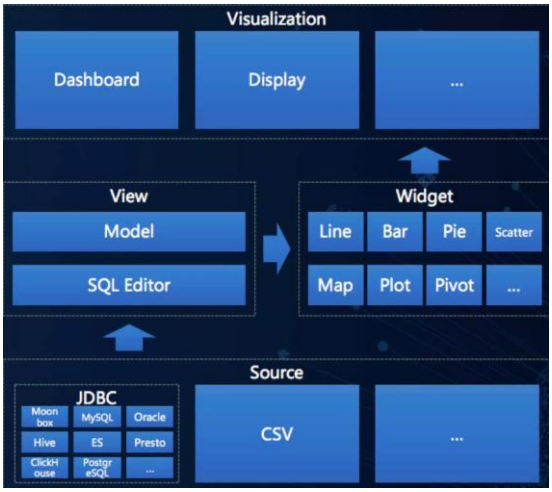
具体操作关键代码如下所示：

```
package writetoDBSqlserver;
import  testnatty.messageclass.*;


public class OperatorData {
      public int OperatorDB(testnatty.messageclass.JsonRootBean jrb)
      {
          String sql2="insert into JsonRootBean (Reference1,CardID,PersonName,Timestamp1,Timestamp2,DeviceCode,Temperature1)   Values (?,?,?,?,?,?,?)
          Object[] obj = new Object[]{jrb.getReference(),jrb.getLibMatInfoList().get(0).getMatchPersonInfo().getCardID(),jrb.getLibMatInfoList().get(
          return DBHelper.ExecSql(sql2,obj);
      }
}
```

6 数据展示

(1)展示平台介绍

	Davinci is oriented towards product managers, business people, data engineers, data analysts, data scientists, etc. It aims to provide a one-stop data visualization solution, which could be both independently used as public cloud/private cloud and integrated into third-party systems as plugin. A simple configuration on Davinci UI can meet multiple visualization requirements. It also supports other visualization features like advanced interaction, industry analysis, pattern searching, social intelligence, etc.

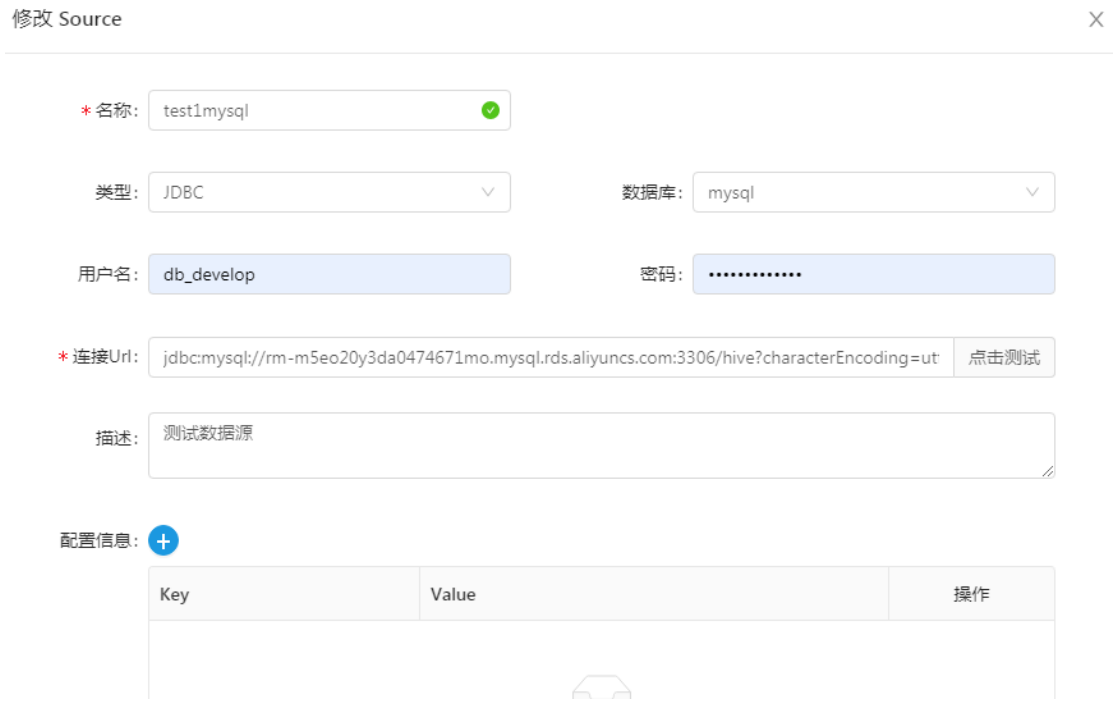
Davinci 是一个 DVAAS（Data Visualization as a Service）平台解决方案，致力于提供一站式数

据可视化解决方案。

(2)实现过程

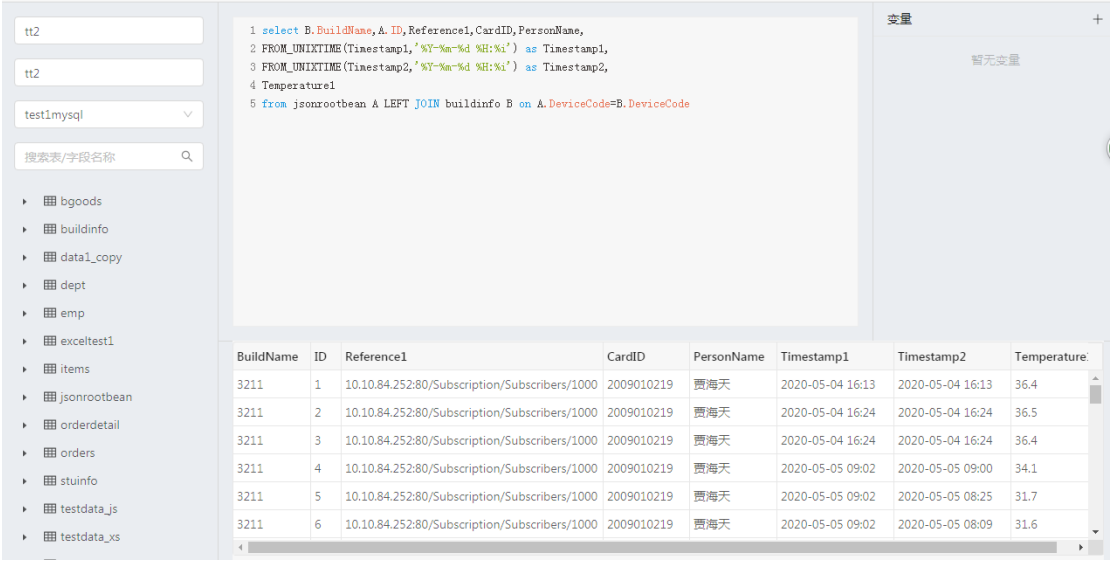
A 管理数据源

支持 JDBC 数据源和 CSV 文件上传



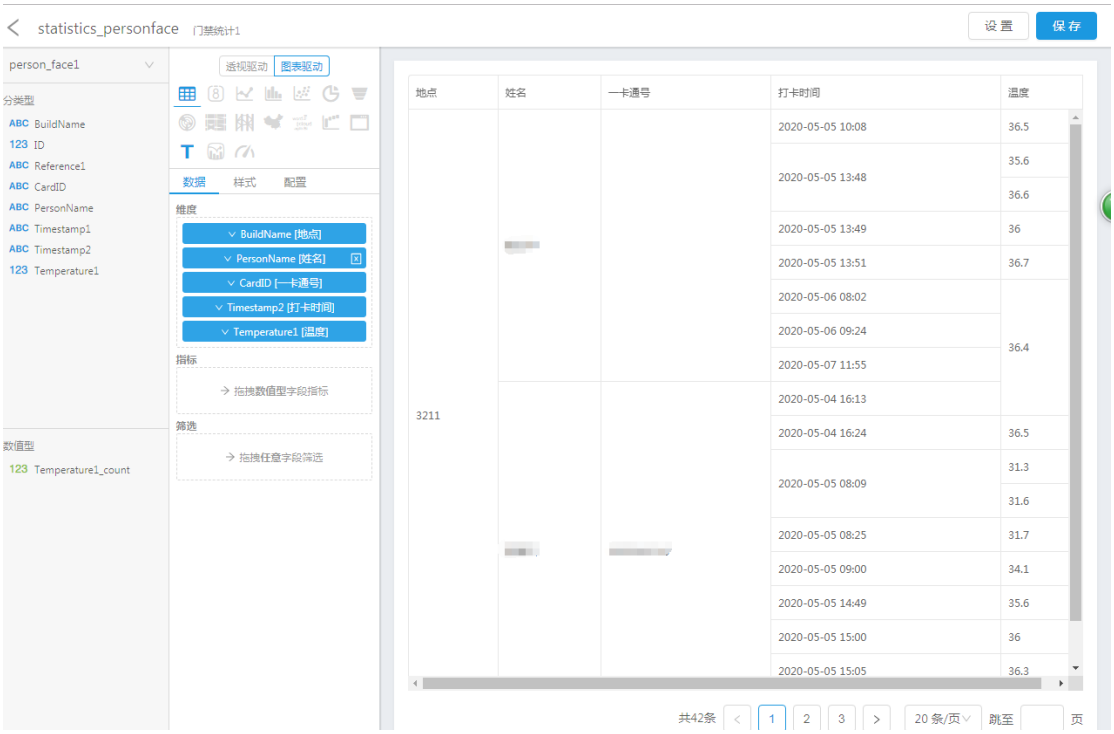
B 数据视图

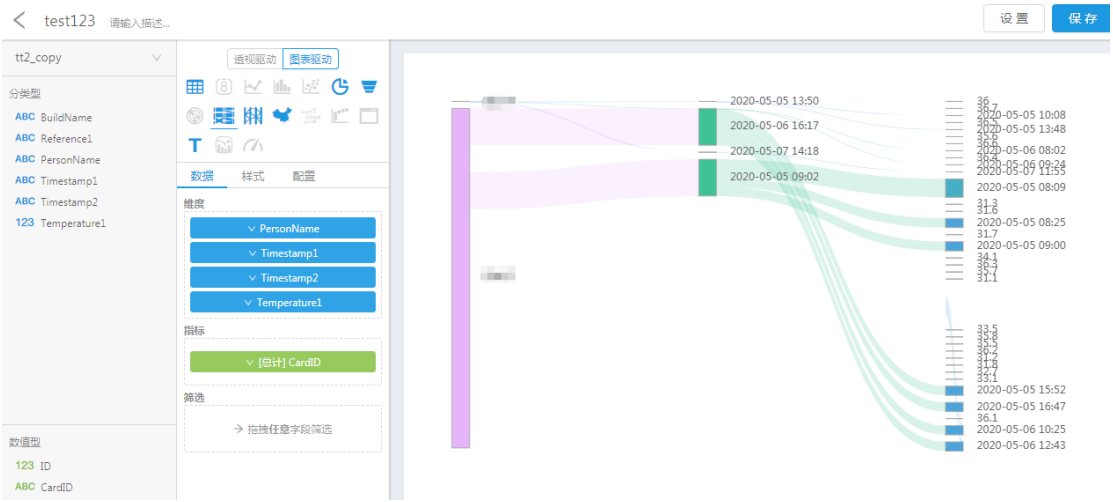
支持定义 SQL 模版、SQL 高亮显示、SQL 语法测试和回写操作



C 可视组件

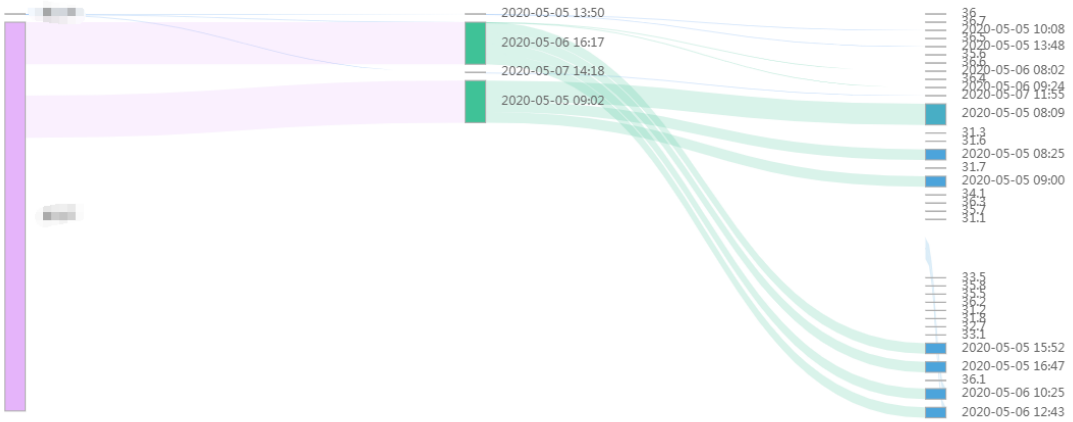
支持预定义图表、控制器组件和自动布局





D 交互能力

支持可视组件全屏显示、本地控制器、高级过滤器、组件间联动、群控控制器可视组件和大数据量展示分页和滑块



可以进行时间和人员的查看并统计

E 集成能力

支持可视组件 CSV 下载、公共分享、授权分享以及仪表板的公共分享和授权分享

7 项目部署

(1)程序部署

A 工程下创建 libs 目录

把所有的依赖包拷贝到 libs 目录下

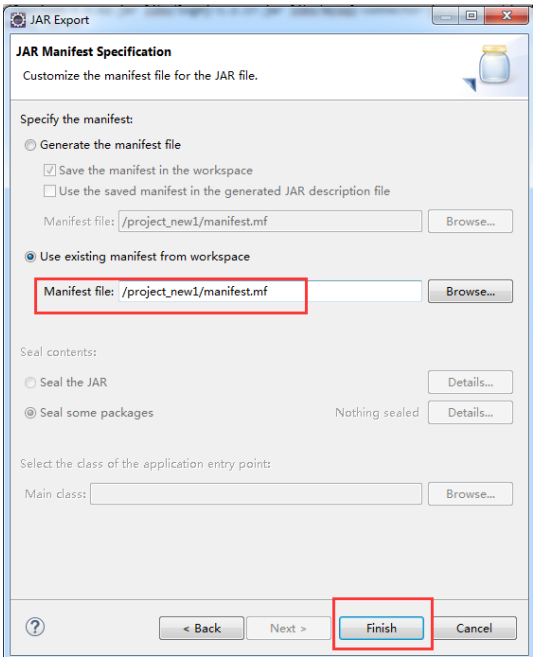
B manifest.mf 文件维护

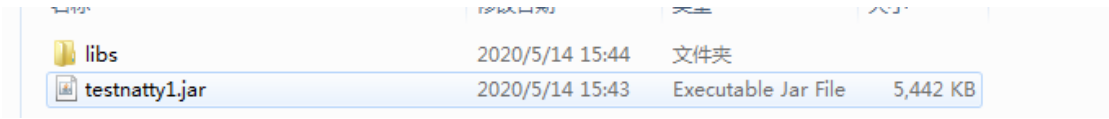
Manifest-Version 指程序的版本号
Main-Class 指程序主方法的入口类
Class-Path 指外来 jar 包的位置

```
Manifest-Version: 1.0
Class-Path: libs/fastjson-1.2.62.jar libs/log4j-1.2.17.jar libs/mysql-connector-java-5.1.7-bin.jar libs/netty-all-4.1.49.Final.jar
Main-Class: testnatty.server

```

C 程序打包部署

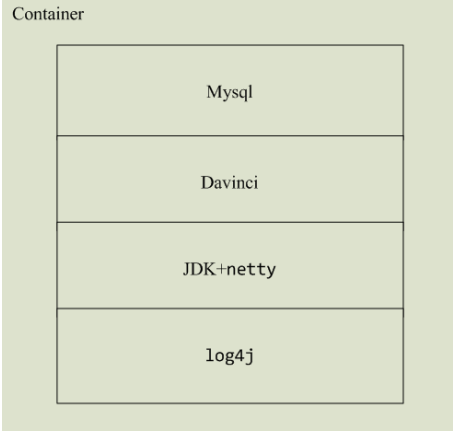




```
E:\logs>java -jar testnatty1.jar
log4j:WARN No appenders could be found for logger (io.netty.util.internal.loggin
g.InternalLoggerFactory).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more in
fo.
```

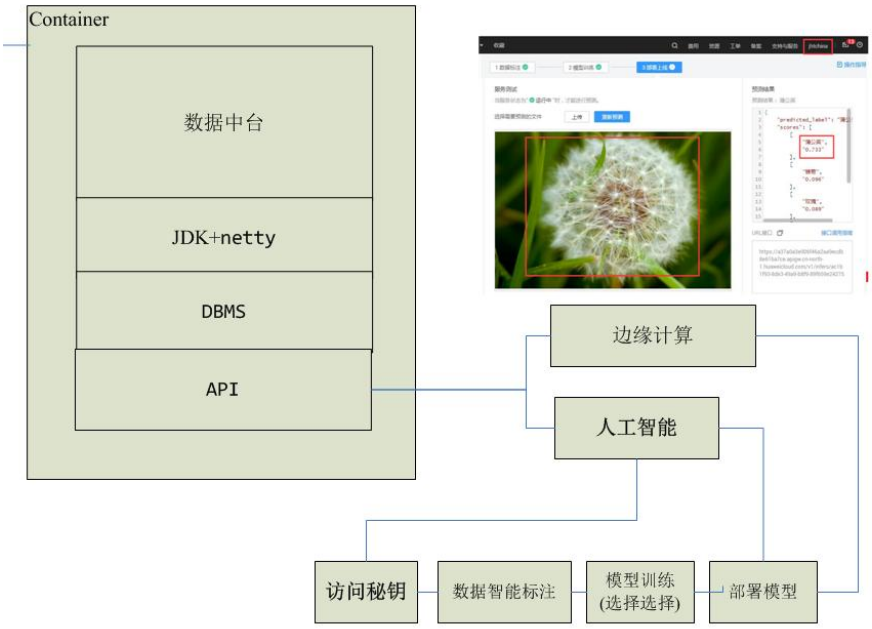
(2)容器化

工程采用 Docker 容器部署

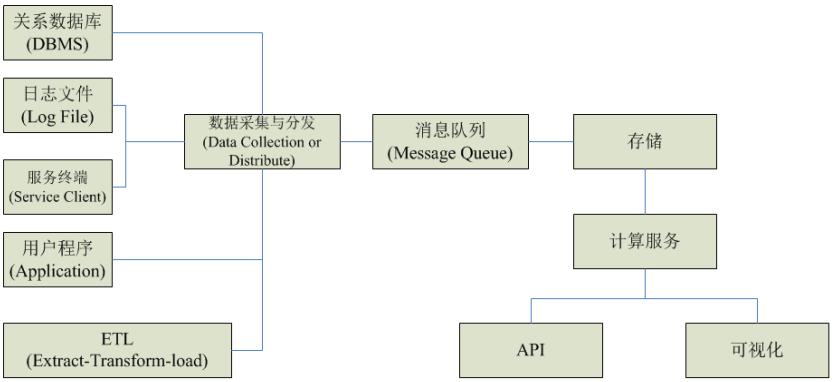
	Developing apps today requires so much more than writing code. Multiple languages, frameworks, architectures, and discontinuous interfaces between tools for each lifecycle stage creates enormous complexity. Docker simplifies and accelerates your workflow, while giving developers the freedom to innovate with their choice of tools, application stacks, and deployment environments for each project.

8 展望

框架引申



数据中台



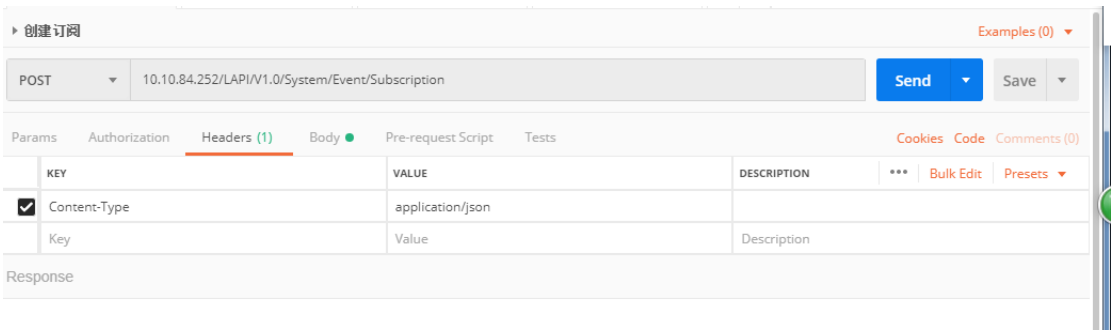
Devops（另外文档分析）

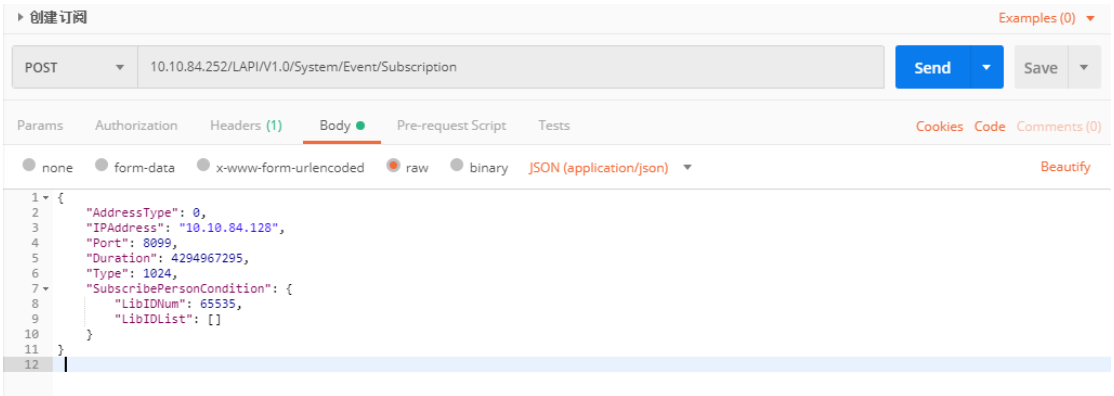
9 项目实施要求

A 客户端要求

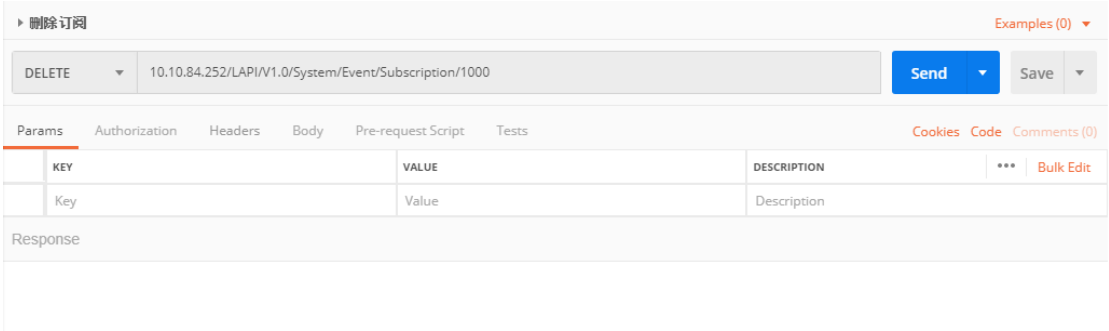
B 订阅设置

B.1 创建订阅





B.2 删除订阅



B.3 获取设备在线状态