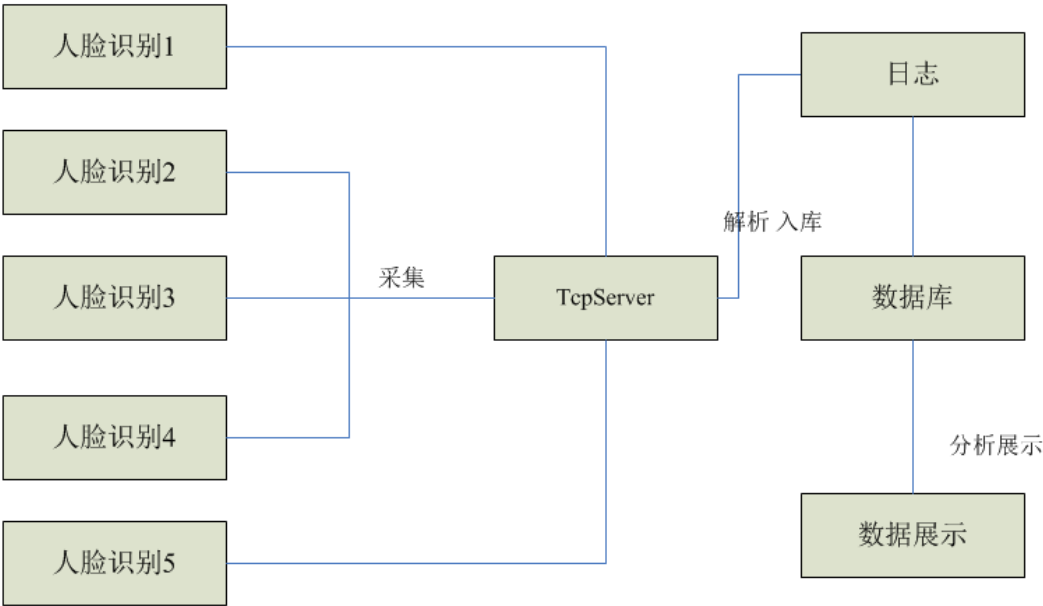


0 项目总体框架

(1)项目整体框架

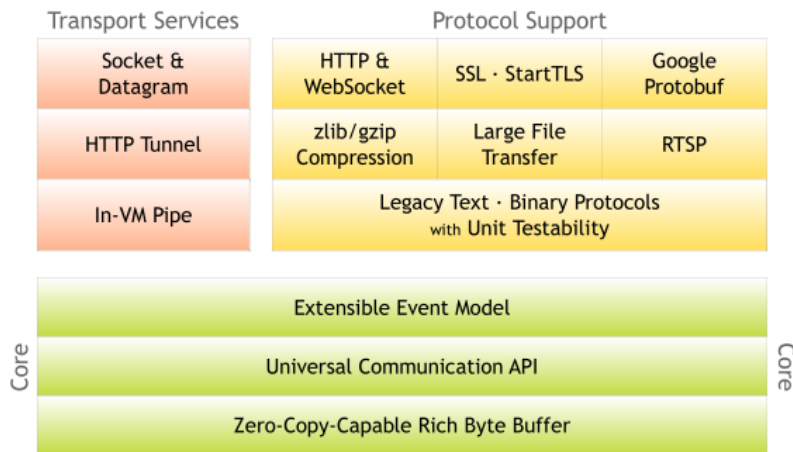


(2)具体实现框架

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

# 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) // 设置发送缓冲区大小
.option(ChannelOption.SO_RCVBUF, 32*1024) // 这是接收缓冲区大小
.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

```
// 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》

**4.5.4 记录推送**

■ 功能描述

门禁设备通过该接口上报过人记录信息。

说明：1.推送通知时，终端会与订阅的 IP 和端口建立 Socket 连接，然后进行发送，发送结束后，短连接主动 close 连接；长连接不主动 close 连接，一直保持连接状态。

2.如果终端连接并发送成功，不会关注平台的解析结果，直接删除此识别记录。

3.终端不关注服务器的响应，只保证一次 TCP 数据的成功发送与否，若未成功将会重复上报此条记录到成功。

4.长短连接都通过该接口上报记录信息

59

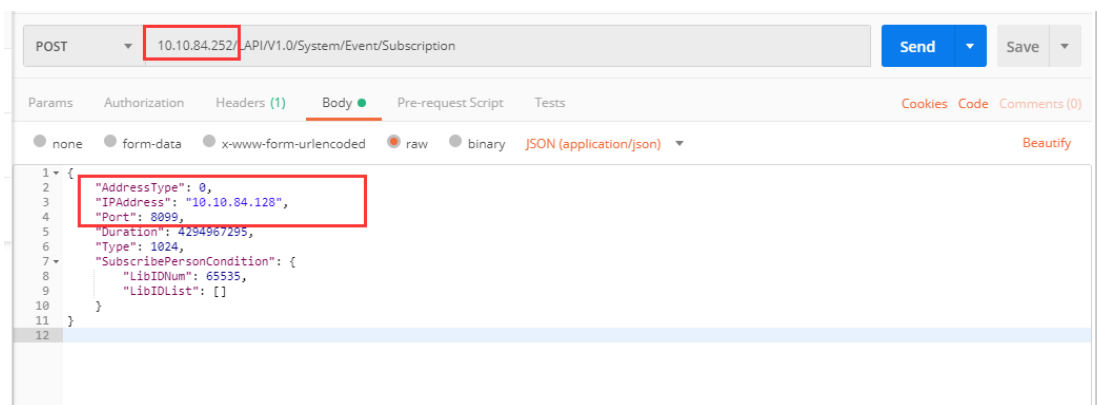
浙江宇视科技有限公司 门禁系统接口文档 文档密级：外部公开

■ 调用方式

人脸识别终端/人脸门禁一体机 → 第三方平台

■ 请求说明

- 请求方式：POST
- 请求 URL: /LAPI/V1.0/System/Event/Notification/PersonVerification
- Content-Type: text/plain



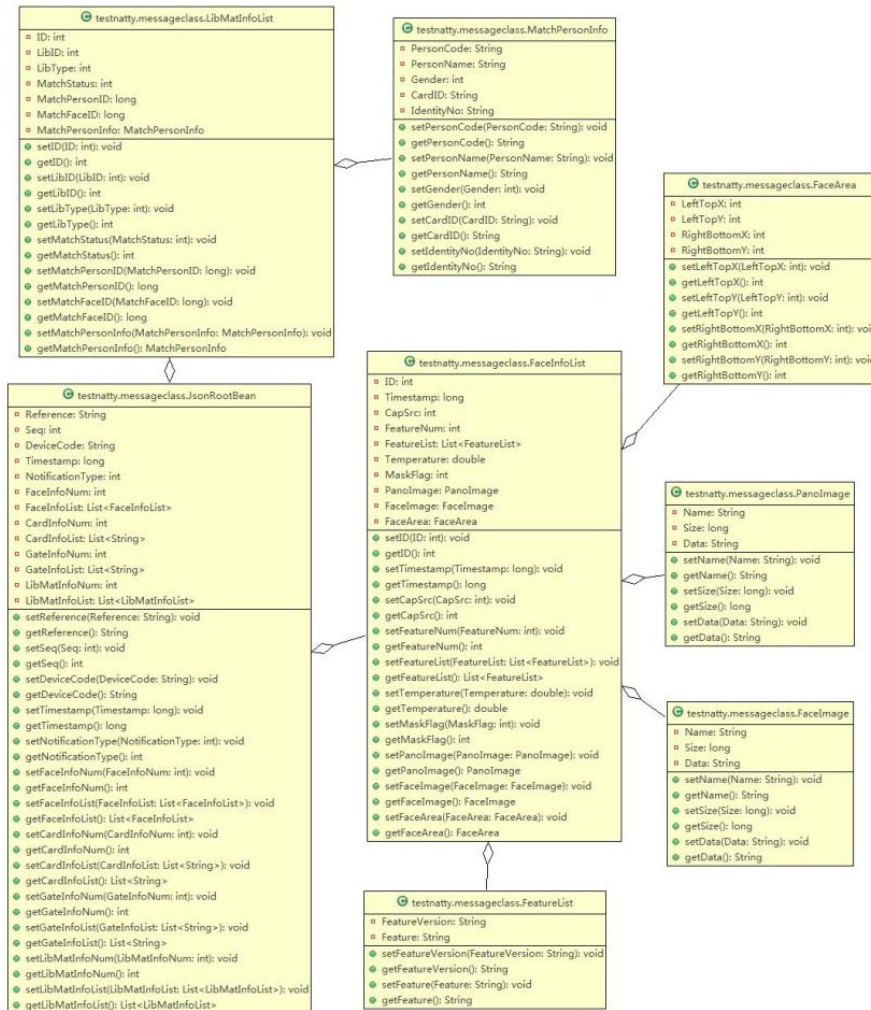
创建订阅->打开服务器及其接收端口->记录推送 ->删除订阅

## (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": ""}], "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：



```

testnatty.messageclass.JsonRootBean jrb=new testnatty.messageclass.JsonRootBean();
jrb= JSON.parseObject(JSON.parse((newMessage1)).toString(), testnatty.messageclass.JsonRootBean.class);
if (jrb!=null)

```

## 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());
```

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	<input type="checkbox"/>	1
DeviceCode	varchar	20	0	<input checked="" type="checkbox"/>	
BuildName	varchar	20	0	<input checked="" type="checkbox"/>	

ID	DeviceCode	BuildName
1	210235C4C53203010994	3211

名	类型	长度	小数点	允许空值	
ID	int	11	0	<input type="checkbox"/>	1
Reference1	varchar	100	0	<input checked="" type="checkbox"/>	
CardID	varchar	10	0	<input checked="" type="checkbox"/>	
PersonName	varchar	20	0	<input checked="" type="checkbox"/>	
Timestamp1	bigint	20	0	<input checked="" type="checkbox"/>	
Timestamp2	bigint	20	0	<input checked="" type="checkbox"/>	
DeviceCode	varchar	20	0	<input checked="" type="checkbox"/>	
Temperature1	double	0	0	<input checked="" type="checkbox"/>	

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

writetoDB.DBHelper
<ul style="list-style-type: none"> <li>getConnection(): Connection</li> <li>executeNonQuery(sql: String): int</li> <li>executeNonQuery(sql: String, obj: Object[]): int</li> <li>executeQuery(sql: String): ResultSet</li> <li>executeQuery(sql: String, obj: Object[]): ResultSet</li> <li>isExist(sql: String): Boolean</li> <li>isExist(sql: String, obj: Object[]): Boolean</li> <li>getCount(sql: String): int</li> <li>getCount(sql: String, obj: Object[]): int</li> <li>free(rs: ResultSet): void</li> <li>free(st: Statement): void</li> <li>free(conn: Connection): void</li> <li>free(rs: ResultSet, st: Statement, conn: Connection): void</li> </ul>
writetoDB.OperatorData
<ul style="list-style-type: none"> <li>OperatorDB(jrb: testnatty.messageclass.JsonRootBean): int</li> </ul>

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','费海天',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.executeQuery(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

```
1/***** Object: Table [dbo].[buildinfo] Script Date: 05/15/2020 11:55:27 *****/
2 SET ANSI_NULLS ON
3 GO
4
5 SET QUOTED_IDENTIFIER ON
6 GO
7
8 CREATE TABLE [dbo].[buildinfo](
9     [ID] [int] IDENTITY(1,1) NOT NULL,
10    [DeviceCode] [nvarchar](20) NULL,
11    [BuildName] [nvarchar](50) NULL,
12    CONSTRAINT [PK_buildinfo] PRIMARY KEY CLUSTERED
13 (
14     [ID] ASC
15 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
16 ) ON [PRIMARY]
17 GO
18
19 /***** Object: Table [dbo].[jsonrootbean] Script Date: 05/15/2020 11:56:04 *****/
20 SET ANSI_NULLS ON
21 GO
22
23 SET QUOTED_IDENTIFIER ON
24 GO
25
26 CREATE TABLE [dbo].[jsonrootbean](
27     [ID] [int] IDENTITY(1,1) NOT NULL,
28     [Reference1] [nvarchar](100) NULL,
29     [CardID] [nvarchar](20) NULL,
30     [PersonName] [nvarchar](20) NULL,
31     [Timestamp1] [bigint] NULL,
32     [Timestamp2] [bigint] NULL,
33     [DeviceCode] [nvarchar](20) NULL,
34     [Temperature1] [real] NULL,
35     CONSTRAINT [PK_jsonrootbean] PRIMARY KEY CLUSTERED
36 (
37     [ID] ASC
38 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
39 ) ON [PRIMARY]
40 GO
41
42 GO
43
44
```

### (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



<div> <div>writetoDBSqlServer.OperatorData</div> <div>OperatorDB(jrb: testnatty.messageclass.JsonRootBean): int</div> </div>
<div> <div>writetoDBSqlServer.DBHelper</div> <div> <div> <div>driver: String</div> <div>url: String</div> <div>user: String</div> <div>password: String</div> </div> <div>DBHelper()</div> <div>getConnection(): Connection</div> <div>getStatement(): Statement</div> <div>getStatement(conn: Connection): Statement</div> <div>getPreparedStatement(cmdText: String, cmdParams: Object[]): PreparedStatement</div> <div>getPreparedStatement(conn: Connection, cmdText: String, cmdParams: Object[]): PreparedStatement</div> <div>ExecSql(cmdText: String): int</div> <div>ExecSql(conn: Connection, cmdText: String): int</div> <div>ExecSql(cmdText: String, cmdParams: Object[]): int</div> <div>ExecSql(conn: Connection, cmdText: String, cmdParams: Object[]): int</div> <div>ExecScalar(cmdText: String): Object</div> <div>ExecScalar(conn: Connection, cmdText: String): Object</div> <div>ExecScalar(cmdText: String, cmdParams: Object[]): Object</div> <div>ExecScalar(conn: Connection, cmdText: String, cmdParams: Object[]): Object</div> <div>getResultSet(cmdText: String): ResultSet</div> <div>getResultSet(conn: Connection, cmdText: String): ResultSet</div> <div>getResultSet(cmdText: String, cmdParams: Object[]): ResultSet</div> <div>getResultSet(conn: Connection, cmdText: String, cmdParams: Object[]): ResultSet</div> <div>buildScalar(rs: ResultSet): Object</div> <div>buildTableModel(rs: ResultSet): DefaultTableModel</div> <div>getTableModel(conn: Connection, cmdText: String): DefaultTableModel</div> <div>getTableModel(cmdText: String, cmdParams: Object[]): DefaultTableModel</div> <div>getTableModel(cmdText: String): DefaultTableModel</div> <div>getTableModel(conn: Connection, cmdText: String, cmdParams: Object[]): DefaultTableModel</div> <div>close(obj: Object): void</div> <div>closeEx(obj: Object): void</div> <div>closeConnection(obj: Object): void</div> </div> </div>

具体操作关键代码如下所示：

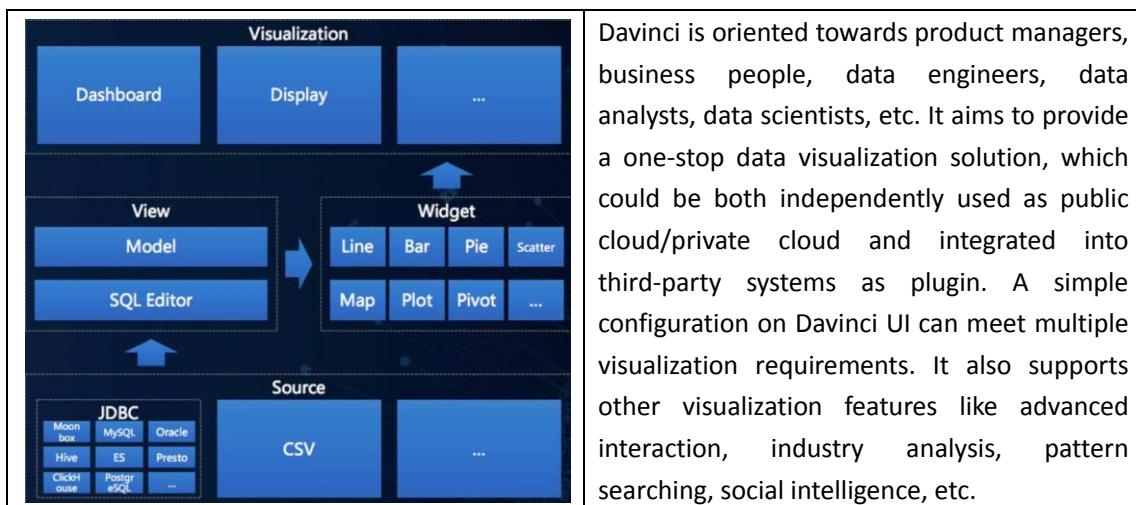
```

1 package writetoDBSqlServer;
2 import testnatty.messageclass.*;
3
4
5 public class OperatorData {
6     public int OperatorDB(testnatty.messageclass.JsonRootBean jrb)
7     {
8         String sql2="insert into JsonRootBean (Reference1,CardID,PersonName,Timestamp1,Timestamp2,DeviceCode,Temperature1) Values (?,?,?,?,?,?,?)
9         Object[] obj = new Object[]{}jrb.getReference(),jrb.getLibMatInfoList().get(0).getMatchPersonInfo().getCardID(),jrb.getLibMatInfoList().get(
10         return DBHelper.ExecSql(sql2,obj);
11     }
12 }

```

## 6 数据展示

### (1)展示平台介绍



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

修改 Source X

\* 名称: test1mysql ✓

类型: JDBC ▼

数据库: mysql ▼

用户名: db\_develop

密码: .....

\* 连接Url: jdbc:mysql://rm-m5eo20y3da0474671mo.mysql.rds.aliyuncs.com:3306/hive?characterEncoding=utf8

点击测试

描述: 测试数据源

配置信息: +

Key	Value	操作
<div></div>		

### 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

tt2

tt2

test1mysql

搜索表/字段名称

lgoods

buildinfo

data1\_copy

dept

emp

exceltest1

items

jsonrootbean

orderdetail

orders

stuiinfo

testdata\_js

testdata\_xs

1 select B.BuildName, A.ID, Reference1, CardID, PersonName,

2 FROM\_UNIXTIME(Timestamp1, '%Y-%m-%d %H:%S') as Timestamp1,

3 FROM\_UNIXTIME(Timestamp2, '%Y-%m-%d %H:%S') as Timestamp2,

4 Temperature1

5 from jsonrootbean A LEFT JOIN buildinfo B on A.DeviceCode=B.DeviceCode

变量

暂无变量

BuildName	ID	Reference1	CardID	PersonName	Timestamp1	Timestamp2	Temperature1
3211	1	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	2020-05-04 16:13	2020-05-04 16:13	36.4
3211	2	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	2020-05-04 16:24	2020-05-04 16:24	36.5
3211	3	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	2020-05-04 16:24	2020-05-04 16:24	36.4
3211	4	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	2020-05-05 09:02	2020-05-05 09:00	34.1
3211	5	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	2020-05-05 09:02	2020-05-05 08:25	31.7
3211	6	10.10.84.252:80/Subscription/Subscribers/1000	2009010219	贾海天	2020-05-05 09:02	2020-05-05 08:09	31.6

C 可视组件

支持预定义图表、控制器组件和自动布局

statistics\_personface 门禁统计1

设置 保存

person\_face1

透视图 图表驱动

分类型

ABC BuildName

123 ID

ABC Reference1

ABC CardID

ABC PersonName

ABC Timestamp1

ABC Timestamp2

123 Temperature1

数值型

123 Temperature1\_count

透视图 图表驱动

数据 样式 配置

维度

BuildName [地点]

PersonName [姓名]

CardID [一卡通号]

Timestamp2 [打卡时间]

Temperature1 [温度]

指标

拖拽数值型字段指标

筛选

拖拽任意字段筛选

地点	姓名	一卡通号	打卡时间	温度
3211	[模糊]	[模糊]	2020-05-05 10:08	36.5
			2020-05-05 13:48	35.6
			2020-05-05 13:49	36.6
			2020-05-05 13:51	36
			2020-05-06 08:02	36.7
			2020-05-06 09:24	36.4
			2020-05-07 11:55	36.4
			2020-05-04 16:13	36.5
			2020-05-04 16:24	31.3
			2020-05-05 08:09	31.6
			2020-05-05 08:25	31.7
			2020-05-05 09:00	34.1
			2020-05-05 14:49	35.6
			2020-05-05 15:00	36
2020-05-05 15:05	36.3			

共42条

1

2

3

20条/页

跳至

页



## 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

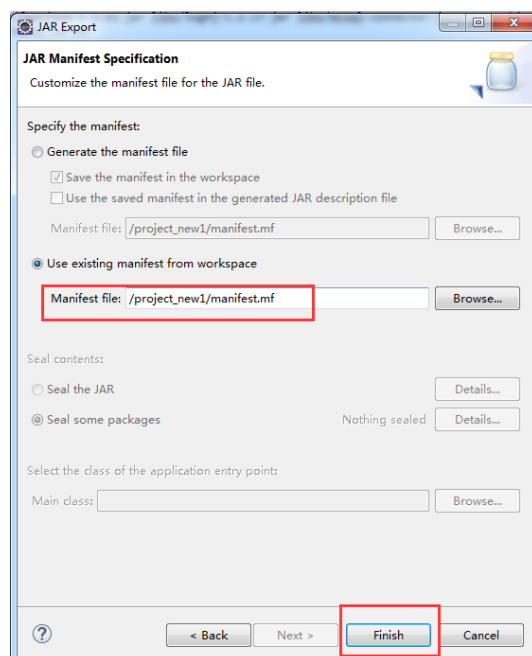
Manifest-Version 指程序的版本号

Main-Class 指程序主方法的入口类

Class-Path 指外来 jar 包的位置

```
1 Manifest-Version: 1.0
2 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
3 Main-Class: testnatty.server
4
```

#### C 程序打包部署

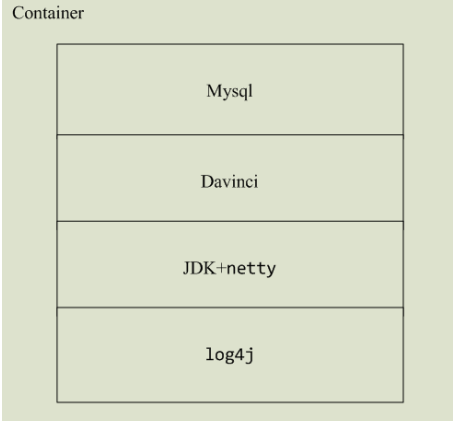


libs	2020/5/14 15:44	文件夹	
testnatty1.jar	2020/5/14 15:43	Executable Jar File	5,442 KB

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

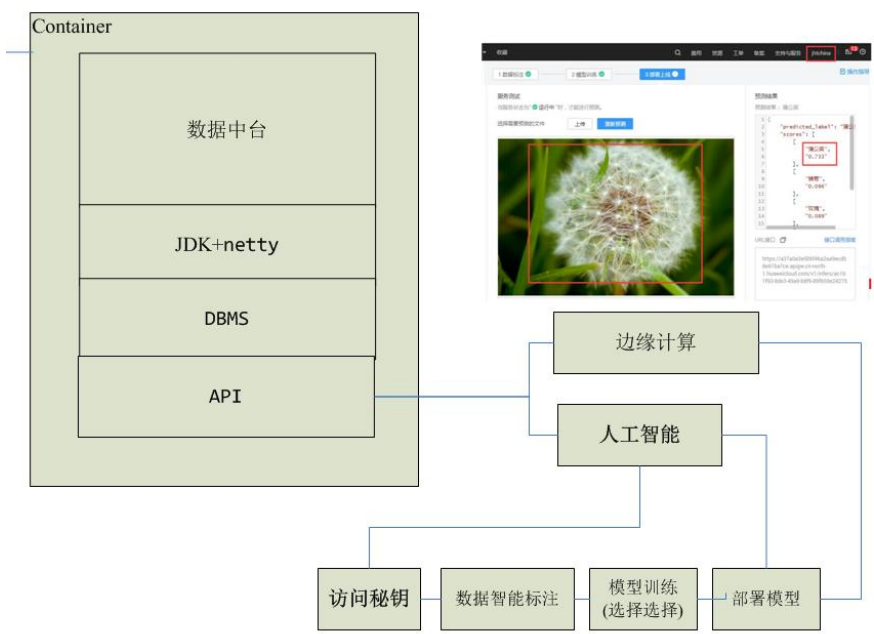
## (2)容器化

工程采用 Docker 容器部署

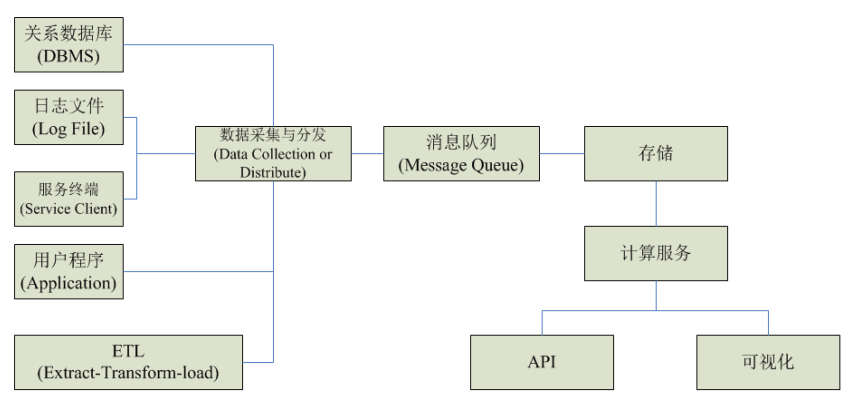
 <p>The diagram shows a large light green rectangle labeled 'Container' at the top left. Inside this container, there are four smaller light green rectangles stacked vertically. From top to bottom, they are labeled 'Mysql', 'Davinci', 'JDK+netty', and 'log4j'.</p>	<p>Developing apps today requires so 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.</p>
--	--

8 展望

框架引申



数据中台



Devops（另外文档分析）

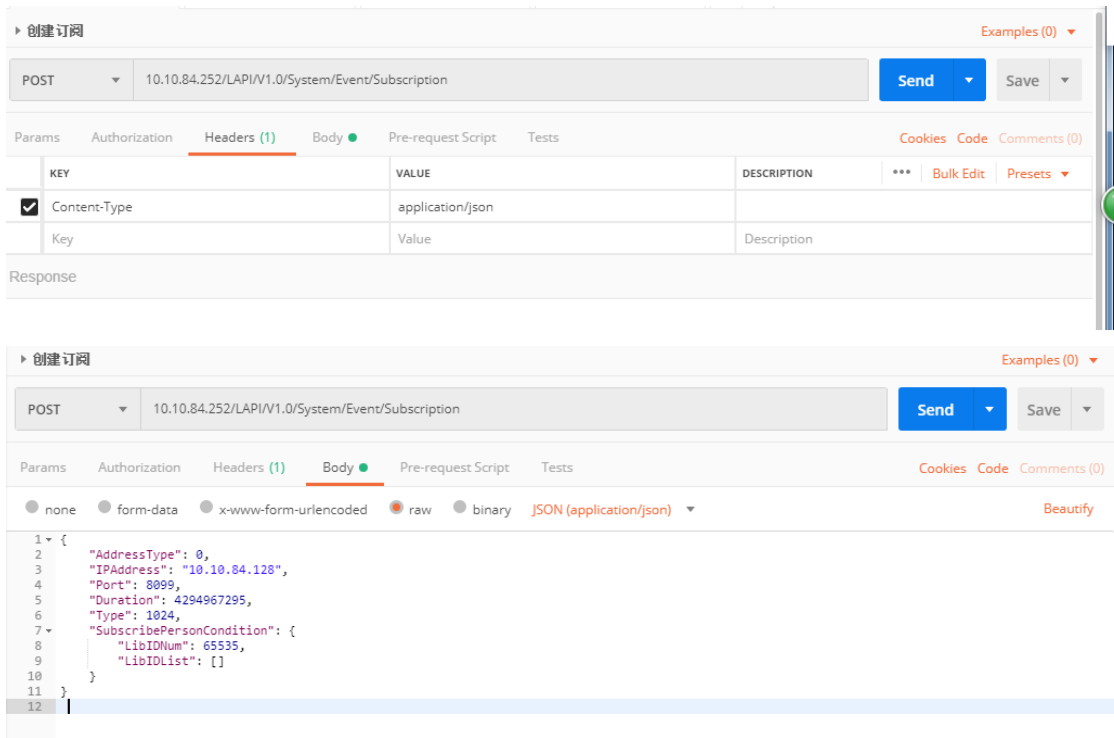
# 9 项目实施要求

## A 客户端要求



## B 订阅设置

### B.1 创建订阅





## B.2 删除订阅

删除订阅

Examples (0)

DELETE

10.10.84.252/LAPI/V1.0/System/Event/Subscription/1000

Send

Save

Params

Authorization

Headers

Body

Pre-request Script

Tests

Cookies

Code

Comments (0)

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Response

## B.3 获取设备在线状态

设备在线状态

Examples (0)

GET

http://172.19.29.192/LAPI/V1.0/System/KeepAlive

Send

Save

Params

Authorization

Headers

Body

Pre-request Script

Tests

Cookies

Code

Comments (0)

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Response