# **小区门禁项目总结**

## **项目背景**

### 立项背景

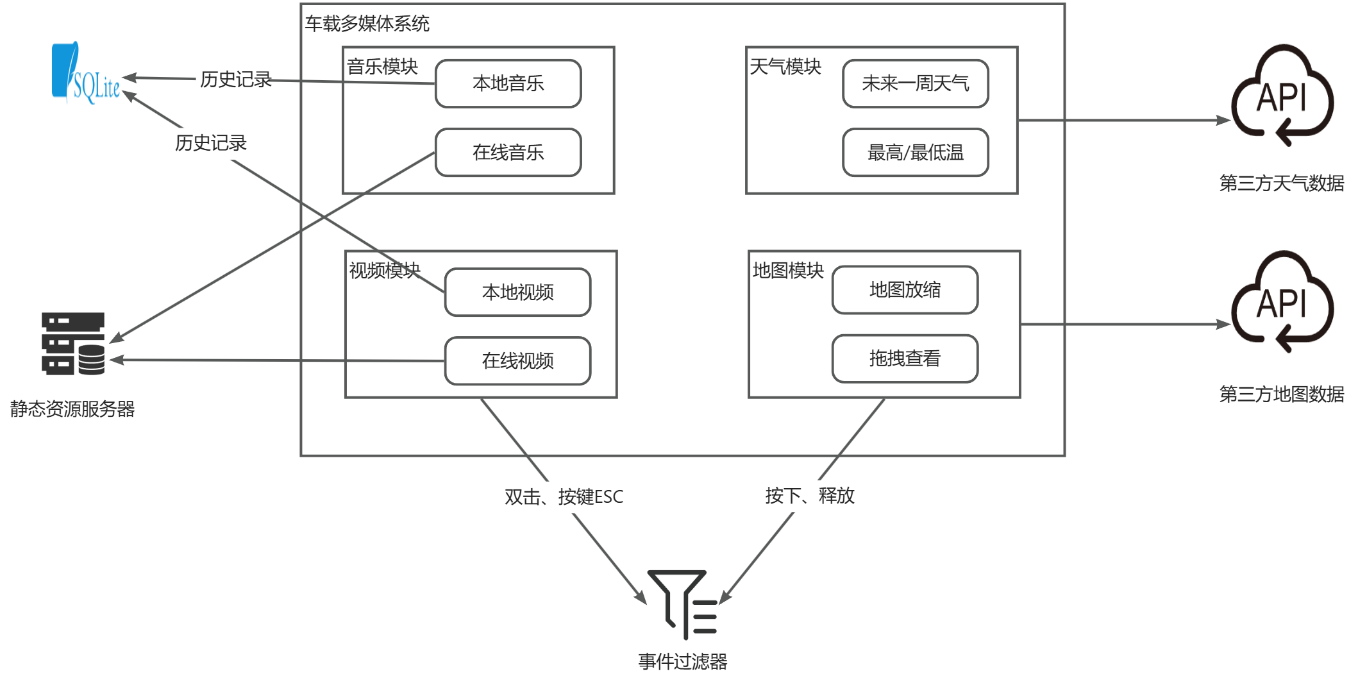
随

### 开发人员及其分工

项

## **项目架构**

项目架构示意图如图2-1所示：



**图2-1 项目架构图**

## **功能设计**

### 3.1 天气

#### 3.1.1 实现思路

先通过时间获取并将其拼接为对应格式的时间字符串，然后将其渲染到相应的label当中，然后构建请求向第三方获取天气信息，再将获取到的天气信息解析获取所需数据，包括天气、气温等数据，然后将对应数据展示到对应的布局当中。

#### 3.1.2 功能点

1. 时间显示：展示出当前年月日与星期信息，并以HH:mm:ss格式实时显示24小时制的当前时间
2. 天气显示：显示当前天气信息并展示对应图标，提示展示出未来六天的天气信息数据。
3. 气温显示：实时显示当前气温，并给出每一天的最高温与最低温。

#### 3.1.3 技术点

1. 定时器：实时刷新时间并动态显示在屏幕上。
2. 请求发送：向三方接口发送请求获取天气数据。
3. 响应信息的接收与解析：从json格式的响应数据中解析出需要的日期、天气、气温等数据并展示在屏幕当中。

### 音乐

#### 3.2.1 实现思路

先将大致布局与按钮设置好，然后给需要切换的图标设置mode并加载对应静态资源，然后编写本地加载部分，通过对应路径加载文件夹中指定后缀的文件，将加载到的数据分别取得其文件名与路径，文件名用于列表中展示，路径用于文件加载播放，同时将文件名与路径存到sqlite数据库当中作为历史记录，下次进入系统会在构造函数中先加载历史记录用于显示，避免用户每次都要重新加载歌曲文件。在线音乐部分则使用nginx模拟静态资源服务器，用户向对应地址发送请求并获取到对应的文件，同样获取歌曲路径与歌曲名分别用于播放与列表显示，而当在线列表与本地列表的tab切换时，需要将播放列表与显示列表进行重新加载，否则会出现一系列问题。音量控制与歌曲播放进度则通过相应的slider控件实现，用户可以拖动进度条进行播放进度的调整，因此需要设置事件过滤器进行按下与松开时间的捕获与其逻辑事件处理，因为qmediaplaylist提供播放模式的选择，因此单曲循环、列表循环等部分可以直接通过对应按钮的点击事件进行模式设置与图标切换。

#### 3.2.2 功能点

1. 播放/暂停/上一首/下一首：用户可以通过点击对应图标执行对应操作，图标样式会随着播放与暂停而改变。
2. 播放模式：有随机播放、顺序播放、单曲循环、列表循环四种播放模式，通过点击图标进行播放模式的切换。
3. 加载本地音乐：用户可以加载本地音乐到播放器当中，并且播放器会将音乐保存到历史记录中，下次用户再使用播放器时无需再次加载相同的音乐。
4. 加载在线音乐：用户可以查看并播放资源服务器中存储的音乐。
5. 滚动歌词展示：播放音乐时自动加载对应的歌词，并随着音乐进度进行滚动，当前播放语句进行字体放大处理。

#### 3.2.3 技术点

1. 样式表设置：图标需要根据点击情况更换对应图标、例如播放模式与播放/暂停等。
2. 数据库操作：加载本地音乐后将响应信息存储到sqlite数据库当中，当插入的历史记录相同时将忽略，不进行重复插入操作，下次进入播放器时先从sqlite数据库加载出历史播放列表数据。
3. 网络请求的发送与接收：通过ngnx模拟静态资源服务器，通过对应网址获取对应音乐文件。

### 地图

#### 3.3.1 实现思路

先将按钮样式与背景图设置好，然后通过向第三方发送对应请求获取当前位置的地图图片，然后将图片先加载到本地在渲染到页面窗口当中，当用户点击放大缩小时改变比例尺属性值重新发送请求获取静态图数据并显示达到地图放缩的功能，放缩在到达一定范围后会产生提示框提醒无法继续放缩。而拖拽功能需要先设置点击与释放的事件过滤器，并通过点击与释放的像素点坐标与比例尺反向计算出移动后的坐标，然后用该坐标重新获取相应静态图，达到拖拽后更换地图中心点的目的。

#### 3.3.2 功能点

1. 放大：调整地图放缩级别，使得用户能看到的地图信息更详细 。
2. 缩小：调整地图缩放级别，使得用户能看到的地图信息更广阔。
3. 地图拖动：用户可以拖动地图查看附近想要看的地点信息，拖动后以新的位置为屏幕中心。

#### 3.3.3 技术点

1. 事件过滤器：过滤出点击与释放时间，完成对拖拽操作的检测识别。
2. 请求发送与响应的接收：向第三方发送请求并获得对应的定位信息静态资源。
3. 文件读写：接收到的第三方静态资源需要先写到本地再进行读取。
4. 提示弹窗：在地图级别放缩到最大最小值时需要产生相应的提示信息。

### 视频

#### 3.4.1 实现思路

主体逻辑与音乐部分实现相同，区别在于视频可以实现全屏与退出全屏，因此需要安装事件过滤器对对应窗口的特定事件进行处理。并且视频可以调整播放速率，提供0.5倍速、1倍速、1.5倍速以及2倍速四种选择。

#### 3.4.2 功能点

1. 视频切换（向前/向后）：切换列表中播放的视频。
2. 暂停/播放：控制视频的暂停与播放，按钮根据播放与暂停展示响应样式图片。
3. 全屏/退出全屏：点击全屏将当前视频全屏播放，全屏模式下双击或点击ESC键退出全屏。
4. 播放速率：选择播放视频的速率，有0.5倍速、1倍速、1.5倍速、2倍速四种选择，默认情况下为1倍速。

#### 3.4.3 技术点

1. 事件过滤器：给全屏窗口安装事件过滤器，过滤出双击事件与ESC按键事件，用于全屏模式的退出。
2. 请求发送与响应的接收：向本地搭建的静态资源服务器发送请求获取相应的在线视频文件。
3. 文件读取：从本地获取相应的视频文件进行加载播放。
4. 数据库操作：读取本地视频时将相关信息存储到数据库中，下次优先从数据库中获取历史记录，避免每次进入都要重复的读取文件。

## **功能实现**

项目功能页面实现如下：



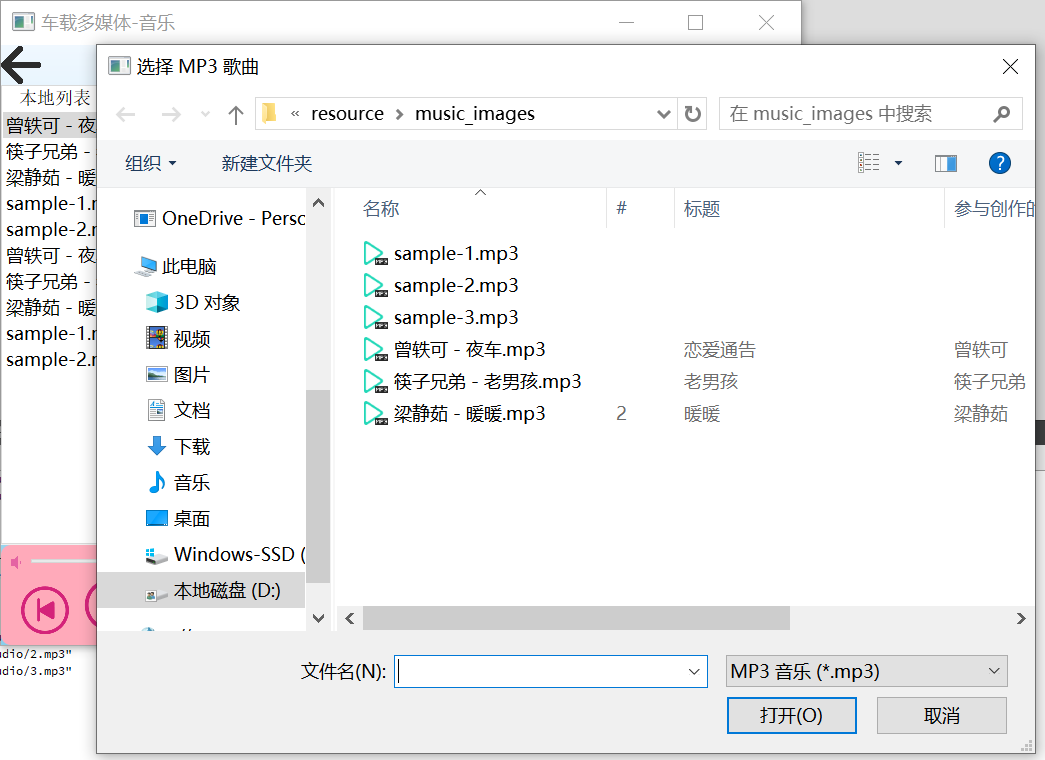
**图4-1 主页面实现**



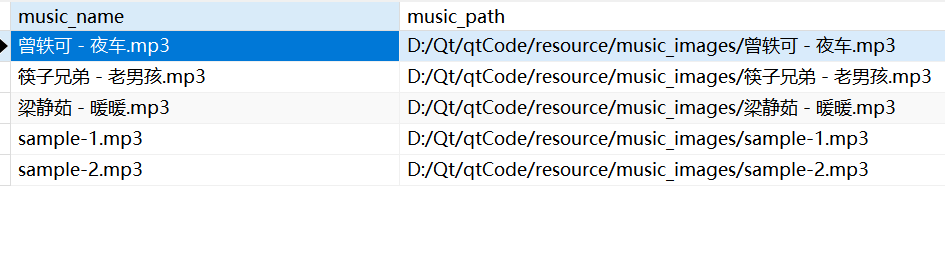
**图4-2 天气页面实现**



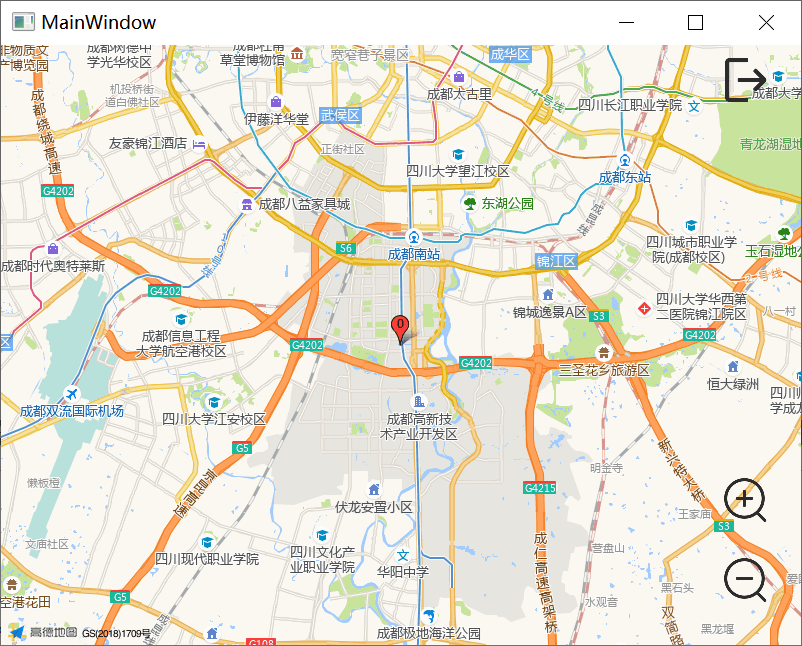
**图4-3 音乐页面实现**



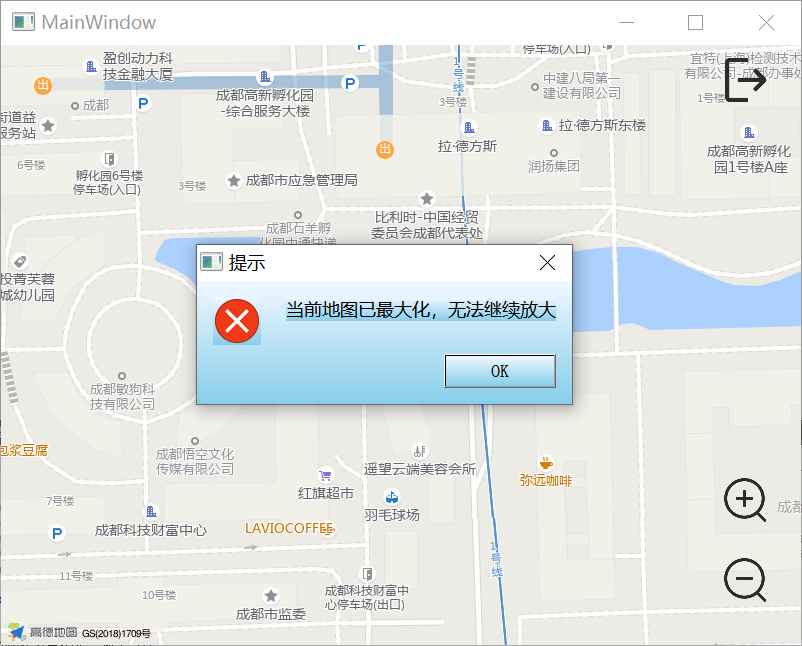
**图4-4 获取本地音乐文件**



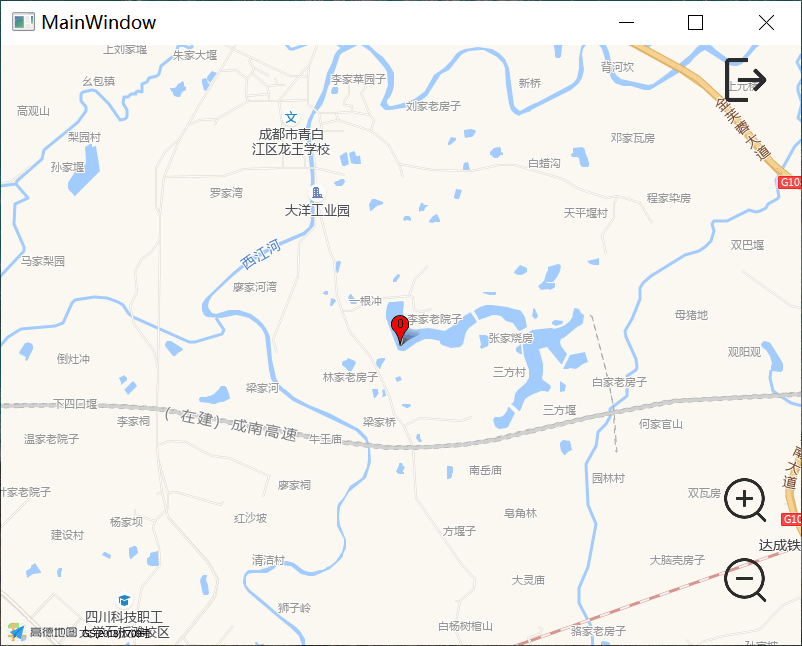
**图4-5 sqlite存储的历史记录信息**



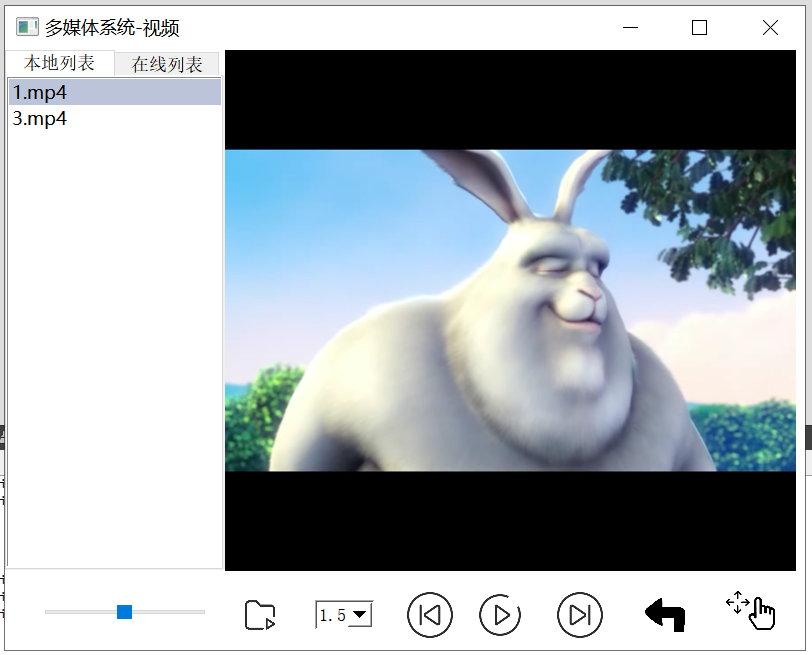
**图4-6 地图页面实现**



**图4-7 放缩极限提示框**



**图4-8 将中心点拖拽到附近**



**图4-9 视频播放页面**



**图4-10 全屏播放**

## **问题与解决方案**

|  |  |  |
| --- | --- | --- |
| BUG等级 | 描述 | 解决 |
|  | 描述：未开启nginx，即模拟远程服务器端口无法访问连接时，点击提交按钮没有任何反应  原因分析：QNetworkAccessManage不会主动超时，因此需要通过手动设置超时或捕获错误信号来做出相应页面提示 | 已解决，设置了网络错误捕获并将错误信息打印到控制台，页面给出错误提示框 |
|  | 描述：人脸识别页面在一开始未识别到人脸时，后续人脸进入摄像头区域无法正常识别  原因分析：将Query类定义在帧循环外，因此遍历完前几帧图像后query的next指针指向末尾，因此无法与之前数据继续正常比对脸部特征向量 | 已解决，可以选择在每次读取帧图像时重新查询，但会影响系统性能，因此选用每次遍历完query后使用first函数将next指针重新指向第一行数据，使用first前需要通过参数设置其支持查询双向滚动，默认为仅支持向前滚动 |
|  | 描述：人脸录入未完成就切换页面到人脸识别时，摄像头调用冲突  原因分析：人脸录入未完成时没满足循环退出条件，因此一直占用摄像头，此时进入人脸识别页面调用摄像头就会产生冲突 | 已解决，通过设置标记量，在点击返回按钮时改变标记量从而退出帧循环并释放摄像头资源 |

## **技术内容**

在项目实现过程中，运用了许多零散的小模块内容，例如网络请求的发送接收、nginx设置静态资源代理等，在这将这些内容进行汇总总结。

### 6.1 网络请求的接收与发送

网络请求的接收发送主要使用QNetworkAccessManager类，需要发送请求时使用其中的方法，例如GET请求对应QnetworkAccessManager::get()方法，然后连接finished信号与相应处理槽函数进行响应的解析操作，在这个槽函数中，QRequestReply作为相应参数，其中就包括了响应数据，此时要先解析其为QByteArray对象，再进行后续操作，如果响应数据为Json格式，则需要将QByteArray对象先转为QJsonDocument对象，再解析其中的QJsonArray与QJsonObject，最后对Json对象使用键值对获取数据。

### 6.2 Nginx设置静态资源代理

先在html目录下创建对应文件夹，然后在对应的config文件中增加内容，最后在对应位置增加Json文件作为响应数据。

### 6.3 本地文件读取

读取本地文件主要通过QFileDialog::getOpenFileNames实现，在使用前还要设置对应过滤器，例如针对.mp3或.mp4格式文件的过滤，然后填写对应参数。

### 6.4 屏幕事件过滤器

分为两步：过滤器的定义、给窗口安装过滤器。定义为eventFilter(QObject \*obj, QEvent \*event)，安装则为QMainWindow::installEventFilters()。

### 6.5 SQLite数据库操作

数据库操作前需要先添加SQLite数据库连接，连接成功后定义操作类QSqlQuery，至于具体的增删改查操作则是根据提交的sql语句完成，而获取查询数据则是通过操作类的next()函数进行遍历，然后以.value()的方式获取对应列数据。