Tampere University

Lam Le

E-TICKET SYSTEM
As an example of Internet of Things application

Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
May 2019

ABSTRACT

Lam Le: E-ticket system as an example of Internet of Things application
M.Sc. Thesis
Tampere University
Master's Degree Programme in Information Technology
May 2019

We are entering a new revolution of technology called Internet of Things (IoT). It enables machines to be connected and exchange data. It brings a huge potential to different areas of technology. It is being used in several applications in different fields: building smart city, smart home, health care and agriculture. This thesis summarizes current IoT application development and the development of an E-ticket system as a demonstration of IoT application.

The literature research was taken place prior to the implementation phase. Several studies on existing scientific articles were done about current IoT applications and its use cases. After that, current IoT technologies were studied and the best technologies were chosen to implement the E-ticket system. It involves Amazon web services (AWS) [8], Java Spring framework [9] and REST [10] calls over 3G [11] connections.

As the result of the research and implementation, the E-ticket system was developed and demonstrated to the teacher. The solution provides police officers a quick and easy way to issue fine tickets and automates the fine payment process. Moreover, it also allows people to give feedback about performance of police officers. It is being sold to the government to solve cash payment issues in developing countries.

The research shows that IoT can be utilized in many different existing applications. It will improve the scalability of the application and allow collecting a huge amount of data, which will be the resource for big data analysis, machine learning, etc. IoT is a part of a big technology revolution we are going through at the moment.

Key words: IoT, internet of things, Amazon AWS, 3G, POS, web services

ACKNOWLEDEGEMENTS

I would like to thank my thesis advisor Dr. Timo Aaltonen, university lecturer of department pervasive computing at Tampere University of Technology. He was always responsive and supportive whenever I needed to discuss. Moreover, he also spent time to read through my thesis and gave constructive feedback on it. He allowed me to have decision of my own work, however steered me to the right direction when he thought I needed it.

I would also like to acknowledge my schoolmate Long Tran for supporting me on server-side implementation of this project. His contribution was very valuable since I had little knowledge about web application development.

Author

Lam Le

Contents

List of figures

List of symbols and abbreviations

IOT	Internet of Things
REST	Representational state transfer
AWS	Amazon Web Services
POS	Point of Sales
UMTS	Universal Mobile Telecommunications Service
BAN	Body Area Network
VPN	Virtual Private Network
SSL	Secure Socket Layer
VPC	Amazon Virtual Private Cloud
EC2	Amazon Elastic Compute Cloud
DB	Database
MVC	Model - View - Controller

1 Introduction

The phrase *Internet of Things* (IoT) [2] refers to a vision of the future Internet where every physical device is connected to the Internet and exchanges information about themselves and their surroundings. It allows digital and physical entities to link up and communicate by different means of information and communication technologies. This enables a new category of innovative applications and services where devices have the knowledge about their surroundings [2].

The term IoT was first introduced by Kevin Ashton in 1998 [1]. Since then, it has caught more and more attention from the educational institutes and industry. Recently, IoT has become the driving force in the development of Industry 4.0 [12], which focuses on making the next generation of intelligent manufacturing, or so-called modern manufacturing. It empowers the possibility that machine can learn, predict and prevent human errors.

Nowadays, the number of connected devices is increasing rapidly. They are computers, smart phones, tablet, smart watches, and many other embedded devices. Most of the devices are equipped with different sensors, such as light, noise, humidity, location, and send the collected information to the server other the Internet. IoT applications can utilize a network of such devices and analyze their enormous data to provide a new class of services with bigger benefits.

This thesis will discuss the general structure of IoT application, existing IoT applications and future development. Moreover, it describes the implementation of IoT E-ticket solution as an example of IoT's use cases. IoT has very big potential in developing smart applications in almost every field in our daily life.

2 Literature review

2.1 IoT architecture

Technology in general and IoT in specific has hugely evolved in the last couple of years. Many types of smart device with intelligent sensors, low power consumption and Ethernet connection were made and become a great source of information. At the same time, supercomputers [14] were boosted with substantial computing power enabling the capability to process big data. Cloud technology [13] with clustering and load balancing feature allows server to handle a huge amount of request concurrently. These changes place a great foundation for the advancement of IoT. The basic workflow of an IoT application [4] is described in figure 1 and explained in detail as the following:

- Smart device equipped with sensors collects environment data. The information can be temperature, humidity, noise, light, motion, etc. One smart device can be equipped with multiple sensors.
- The sensor data is transmitted to IoT server via wire or wireless communication protocol. It is stored in IoT server internal database. The design principle of IoT server allows it to handle a substantial number of requests concurrently, thus it is able to support big amount of smart devices.
- IoT server processes and analyzes the received data via a computation and processing system. The statistical result is displayed to end user via a web interface. Moreover, a decision-making and action-invoking unit can be added to notify administrator about current system state or when unexpected event happens.

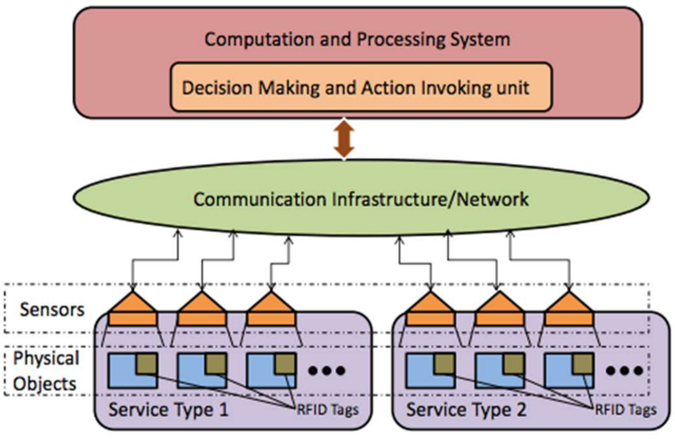

Computation and Processing System
Decision Making and Action Invoking unit
Communication Infrastructure/Network
Sensors
Physical Objects
Service Type 1
RFID Tags
Service Type 2
RFID Tags

Figure 1. Basic IoT work flow [4].

Architect of IoT application depends on the advancement of technology and the design of applications and business models. IoT needs to deal with challenges related to security and privacy. It is due to the fact that it connects to billions of devices over Ethernet, which implies cyber security risk. Moreover, substantial amount of data storage is also an important requirement for IoT application. Last but not least, scalability and reliability are also challenges that need to be addressed. As can be seen from figure 2, the structure of IoT is divided to five layers [4]:

- Perception layer: This layer, which is also called device layer, consists of physical devices and their sensors. There are different methods for identifying objects: RFID, barcode, infrared sensors, etc. This layer is responsible for identifying the objects and collecting their sensor information. The type of information varies and depends on the sensor and the intended use. It can be location, temperature, identification, humidity, etc. The gathered information is handed to network layer to be sent to IoT server.
- Network layer: This layer, which is also called transmission layer, is responsible for transferring the gathered information from perception layer to IoT server over a secured connection. The transmission protocol can be wire or wireless such as 3G, UMTS, Wifi, Bluetooth, infrared, ZigBee, etc.
- Middleware layer: This is the first layer in IoT server, responsible for collecting the data from network layer and storing it in a database. After that, it processes, analyzes the received information and takes actions based on the analysis result.
- Application layer: This layer implements different applications based on the information collected in middleware layer. The applications implemented in this layer will be discussed in the following chapter.
- Business layer: This layer manages the applications and provides services for end users. It includes business models, visual elements of the received and processed data. Business model plays an important role on the success of IoT application. It defines the action and business strategy taken from analysis results.

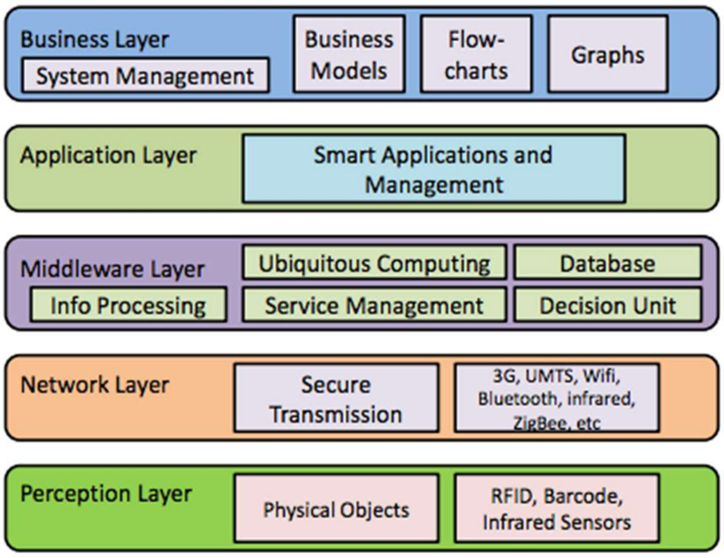


Figure 2. IoT architecture [4].

2.2 IoT applications

There are various application domains, which will be benefited and greatly improved as a result of the development of IoT infrastructure and technology. Below is a list of main sectors where IoT application can have a great impact.

2.2.1 Smart city

The concept of smart city focuses on using IoT to manage public services such as: energy, lighting, transport, waste management, and pollution control. The concept has been deployed successfully in many cities such as London, Amsterdam, Singapore, etc [5]. It includes many public sectors, where a few popular ones are smart environment, smart mobility, smart governance, smart grid, smart traffic, smart home.

Many ongoing smart city IoT projects are focusing on "identification of smart city scenarios, the integration and interaction of various IoT data sources and systems, big data representations and analytics, security and privacy issues" [5]. The target is to understand different system structures, businesses, and citizens to be able to model a smart city system that is applicable to each particular city.

One of the biggest Smart city IoT project is FP7 SmartSantander project [6]. In the project, a large number of IoT devices were deployed in many different urban scenarios, which are distributed over several cities. The project tested the analysis and evaluation of IoT application in large-scale, real-life situation.

2.2.2 Smart home

Smart home is a concept of having a home automation system controlling all devices and operations inside a house. It also enables remote control capability of a smart home. Some advanced system allows using smartphone as a remote controller managing all devices and appliances at home. Smart home works best when allocated within a smart city where an apartment can be integrated as part of communication facility of a building, or a residential area. This provides the infrastructure for a wide range of applications in different areas, such as security, information sharing, fault detection, service automation, and entertainment system.

Saving power is an important area of IoT smart home application. The smart home can be integrated with smart grid to optimize power consumption cost. The application can provide a service, which monitors the power consumption of all household devices and re-schedules, arranges their operations to avoid peak period with more energy cost while still maintains the performance.

2.2.3 Health service

IoT health care service focuses on monitoring patient health remotely and continuously. It brings the benefit of preventing serious patient condition, improving life quality of elderly people through a set of smart services and health statistics. The medical and healthcare sector will be greatly benefited from the development of IoT [2].

Health service works via a set of sensor devices. Advanced sensing devices allow continuous monitoring of medical parameters and vital functions of patient such as blood pressure, heart rate, and sugar level. The collected data is sent to a gateway device via low-power communication protocol, eg Zigbee [15] or BLE [16]. Gateway device is responsible for forwarding the data to IoT server to process. The analysis result is used by medical staff to diagnose and manage patient's health.

One of the popular terms in IoT health service is Body Area Network (BAN) [17], a network of inter-connected wearable devices, allowing doctor to monitor patient's health remotely. BAN devices may be implanted inside patient's body or worn on hands or kept in the pocket. BAN devices may need to be connected to other devices such as mobile phone to send the collected data.

At the moment, IoT health application has not reached its full potential. The concept of a connected health care system and medical devices has a huge potential to improve people's health and their well-being. However, its potential has not been fully realized and implemented yet. It is expected that big IoT start-up in health sector with application deployed in large scale will rise in the near future.

2.2.4 Agricultural

IoT agricultural application controls and monitors agricultural production and feed by using sensor system. The system consists of different sensors, which can measure data, perform data processing, and notify the farmer via a communication facility. For example, device sends text message informing that a portion of land needs more water.

Intelligent farming system will help farmer to have more efficient planting practice by providing statistics of land condition and climate variability during the year. This will increase the productivity of the farm.

2.3 Microservices

Microservices [18] is an application implementation approach focusing on decomposing applications into multiple single-function modules which have their own interfaces and can be deployed independently. This approach allows each module to be managed independently by a small team. As a result, it reduces the needed communication between teams and risk of a change. Consequently, it also speeds up project delivery. Last but not least, Microservices architecture allows scalability of the application, in case high load of a particular service.

Below are the main features of Microservices architecture.

- Decomposing:

As opposed to traditional application with large implementation, Microservices decomposes it into many independent small services (so-called microservice). Each microservice is responsible for a specific business domain (eg: organizations, departments, employees). It has its own user interface, business logic, database connection like a small application.

As in below figure 3, the application is split into three microservices: organization service, department service, employee service. Each handles its own respective business logic and is communicated through a gateway. It is similar as traditional software architecture. However, these three services are running as standalone applications, opposed to traditional software where they are part of a single application.

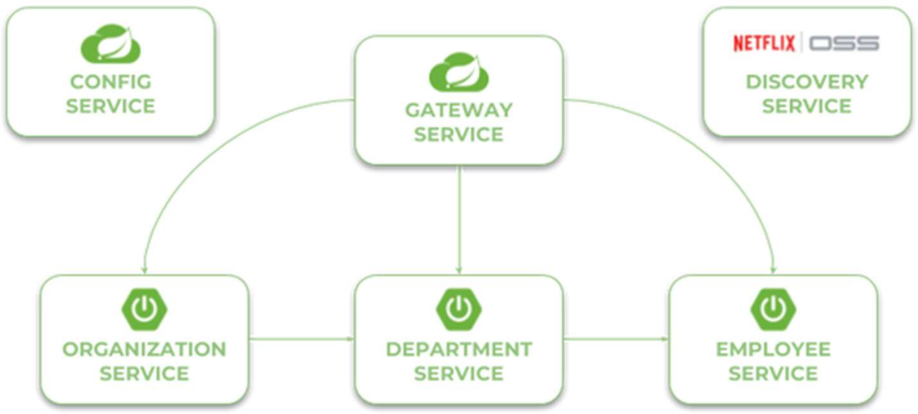


Figure 3. Microservices implementation from Netflix OSS [19].

❖ Single- function:

Each microservice implements a specific entity or business domain. In other words, each microservice handles a single function. It can implement many tasks; however all of them should be related to this single function.

❖ Independent

Microservice does not need to know about implementation of other microservices. This enables each service to be tested and deployed independently. However, one micro-service can use the service of other microservices. For this to work, each service should have a clear interface defining how it can be communicated. The interface of a service should be kept intact while updating its implementation to avoid affecting other services.

2.4 Microservices with Spring Boot and Spring Cloud

Spring Boot [20] is one of the most well-known and widely-used java frameworks for implementing Microservices. It is used to build stand-alone Spring applications [23] which are ready to be run in any production server. Spring Boot makes it easy for developers to understand and develop Spring applications. Thus it increases productivity and reduces development time.

Spring Cloud [21] is a framework built on top of Spring Boot. It contains a collection of design patterns to implement cloud-native applications. In other words, it provides an architectural design and supported libraries to implement cloud-based applications which can be scaled up if needed. This allows developer to focus on core business without worrying about cloud infrastructure aspect. Below are few basic, widely-used Spring Cloud modules.

- Spring Cloud config service: This service is used to store configuration of other microservices. This allows various application configurations, such as database URL, to be defined in one central place. Consequently, configuration options can be updated merely by restarting this service and not affecting other services.
- Spring discovery service: This service, which is also called service registry or eureka server, is a central place where services register themselves. Each service is given a name. Several instances of the same service will have the same name. Discovery service allows other service (gateway) to access an available instance of a service based on its name. It abstracts away the IP addresses of services; other services do not need to know about IP addresses.
- Spring gateway service: This service is the first entry point to access the application from outside. It has a mapping between URL prefix and corresponding service name. When receiving a request from outside, it will recognize needed service name, and ask discovery service to provide corresponding available service based on that. In summary, gateway service is an intermediate layer between users and services. Zuul [6] is an implementation of gateway service. It provides dynamic routing, monitoring services. Moreover, it also load balances between multiple instances of the same service using Ribbon [7].
- Spring Cloud security: In case microservices need to be protected from anonymous access, Spring Cloud provides single sign-on feature using OAuth2 [22]. This distributes the authentication context across services.

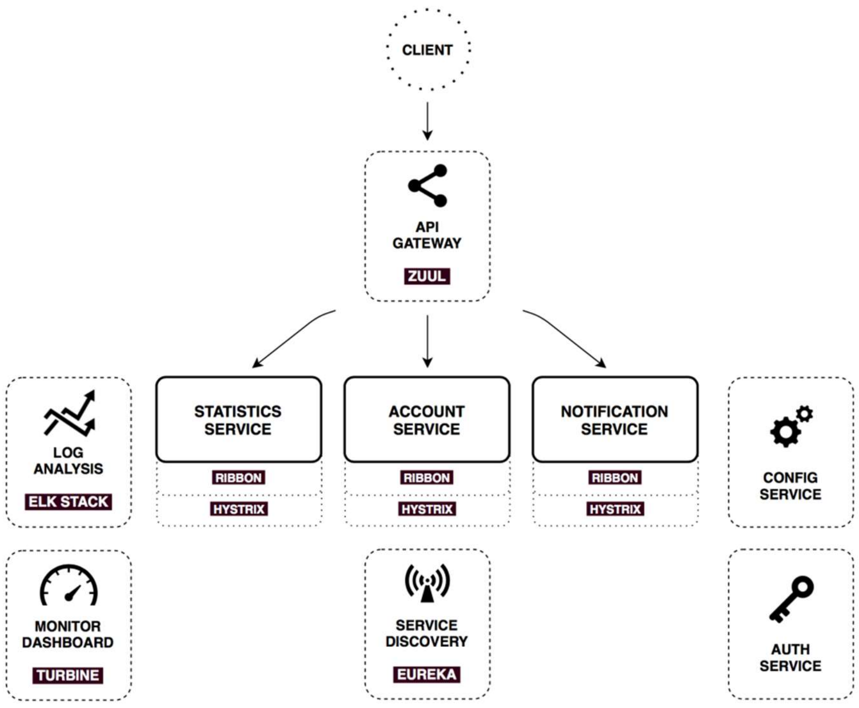


Figure 4. Microservices architecture using Spring Cloud[7].

Above figure 4 demonstrates the general structure of Microservices architecture using Spring Cloud. In this example, all services, such as statistics, account, and notification are standalone services. They are accessed through a gateway. Services are discoverable via discovery service. Apart from those, other services provide enhancement features like configuration, authentication, logging, dashboard monitoring, etc.

3 Software design

3.1 Application overview

E-ticket solution aims to provide cities a convenient way to issue and manage fine tickets. Moreover, it also helps citizen pay for the ticket and give feedback of the performance of police officer. Feedback will eventually help to improve the service of policeman and hence contribute positively to the relation between government and citizens.

At the highest level of abstraction the architecture of the solution consists of two main parts: the IoT cloud server and handheld device.

The handheld device is used for issuing the tickets. It looks similar like normal POS machine with a screen, a keypad, barcode scanner and thermal printer. It contains software required to display the data to the user, as well as the logic to enable the user to communicate with the IoT server. Policeman will use the E-ticket device to query needed information about the driver and vehicle from the IoT server. At the same time, he can create and print a fine ticket to the driver committed a fault. The fine ticket information is stored in the IoT server to be processed further.

The IoT cloud server contains two web applications: E-ticket Client and E-ticket Admin.

- E-ticket Client: public web application for everyone to check their own fine tickets and pay for them. After paying the fine ticket, citizen will have the possibility to give feedback about policeman’s service.
- E-ticket Admin: private web application used by system admin to manage fine tickets, feedback and statistics. It also provides communication with handheld device. This web application can only be accessed via a computer inside VPN network and with user credentials.

The two web applications share the same database. However, due to their respective purpose, they have different access rights. E-ticket Client can be publicly accessed while E-ticket Admin can only be access via VPN and only for authenticated users.

The communication interface between handheld device and E-ticket Admin is REST on top of SSL and key authentication. Moreover, the device uses a special sim card that enables it to connect to the same VPN network with E-ticket Admin. This ensures that the connection between them is secure.

Below figure 5 demonstrates the overview of the whole E-ticket solution.

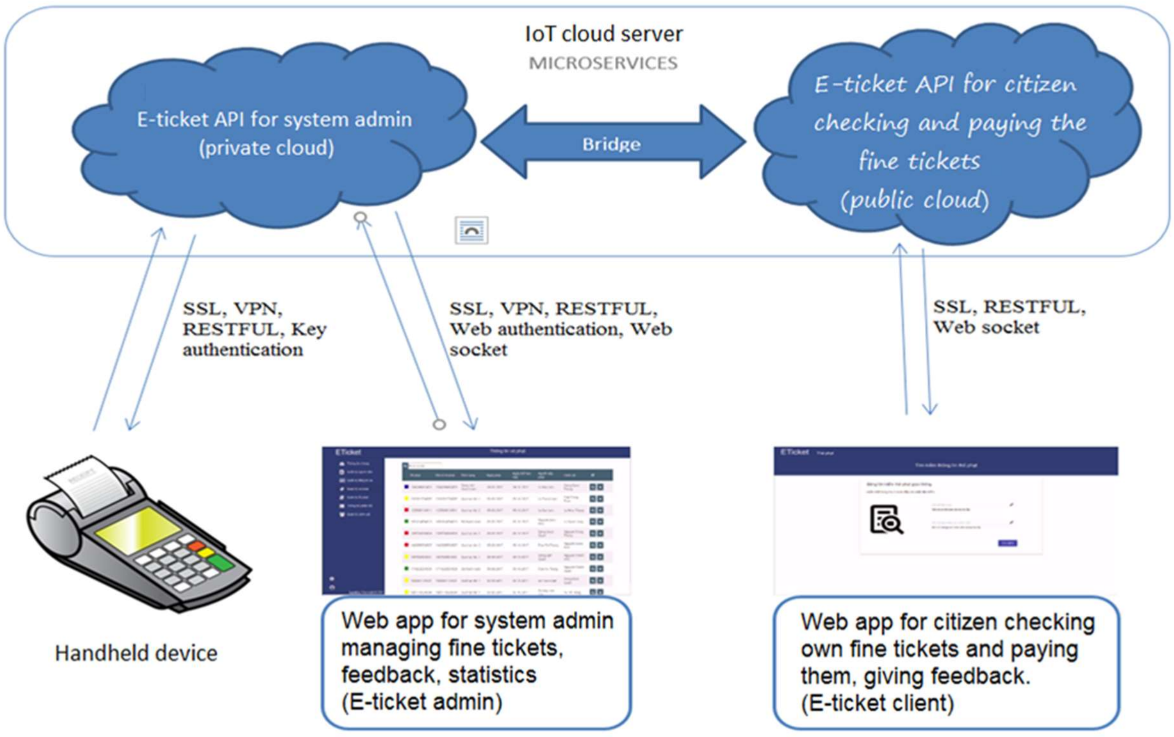

IoT cloud server
MICROSERVICES
E-ticket API for system admin (private cloud)
Bridge
E-ticket API for citizen checking and paying the fine tickets (public cloud)
SSL, VPN, RESTFUL, Key authentication
SSL, VPN, RESTFUL, Web authentication, Web socket
SSL, RESTFUL, Web socket
Handheld device
Web app for system admin managing fine tickets, feedback, statistics (E-ticket admin)
Web app for citizen checking own fine tickets and paying them, giving feedback. (E-ticket client)

Figure 5. Overview of E-ticket solution.

IoT cloud server is split into two sections: public cloud providing services for public access (normal user access), and private cloud providing services for authenticated access on top of VPN network. The private cloud serves both E-ticket devices and E-ticket Admin web application. Both private and public clouds are bridged and share the same database.

3.2 Use cases

The use case diagrams for this solution illustrate the interactions between actors and use cases (actions) in the solution. There are three actors in this solution: police issuing fine tickets, citizen paying the fines, and administrator monitoring tickets' status and feedbacks. Below figures show the three use case diagrams for each actor.

Figure 6 depicts the actions a police officer should perform while giving a fine ticket. As can be seen from the diagram, the police needs to login first and all actions happen in a hierarchical order from top to bottom. First, the police will check the citizen info: basic information, driving license and unpaid tickets. Then, he will select a new fine type and create a new ticket. Finally, he will print out the ticket and give it to the person committed the fault to pay.

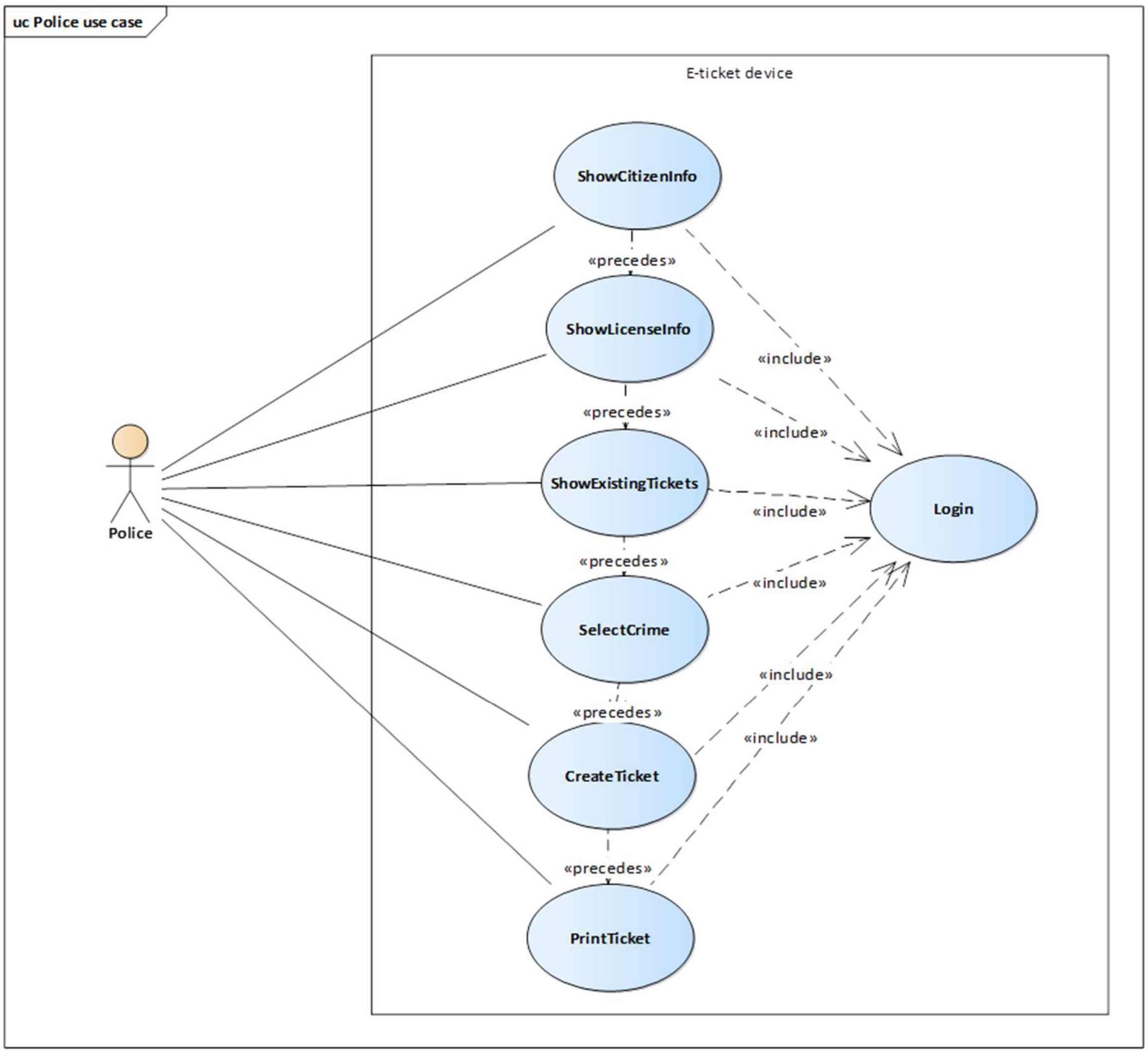


Figure 6. Police officer use case diagram.

Figure 7 illustrates citizen use case diagram. It describes different actions a citizen who committed a fault can do in the E-ticket public web application. Again the actions happen in a hierarchical order from top to bottom. First user can search for the ticket information based on their social security number or ticket‘s number. Then he can pay the fine ticket online via the web application. Finally, he can give the feedback about police performance only after the payment is complete.

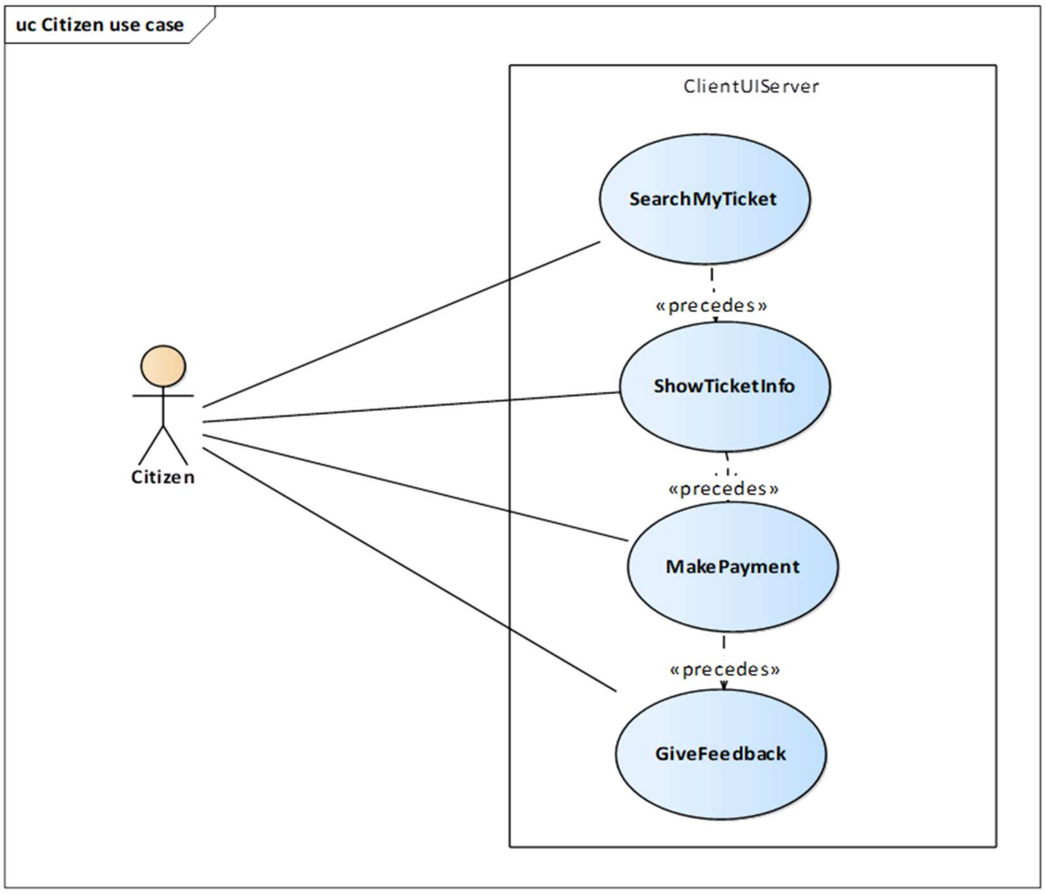


Figure 7: Normal user use case diagram.

The last use case diagram in figure 8 shows various actions an administrator can perform on private admin web application. All actions are protected behind login page. On this page, administrator can manage citizens, their driving licenses, vehicles and existing fine tickets. Moreover, they can update fine tickets's information based on the payments received in a designated bank account. Last but not least, they can review polices' performance and get a report out of that.

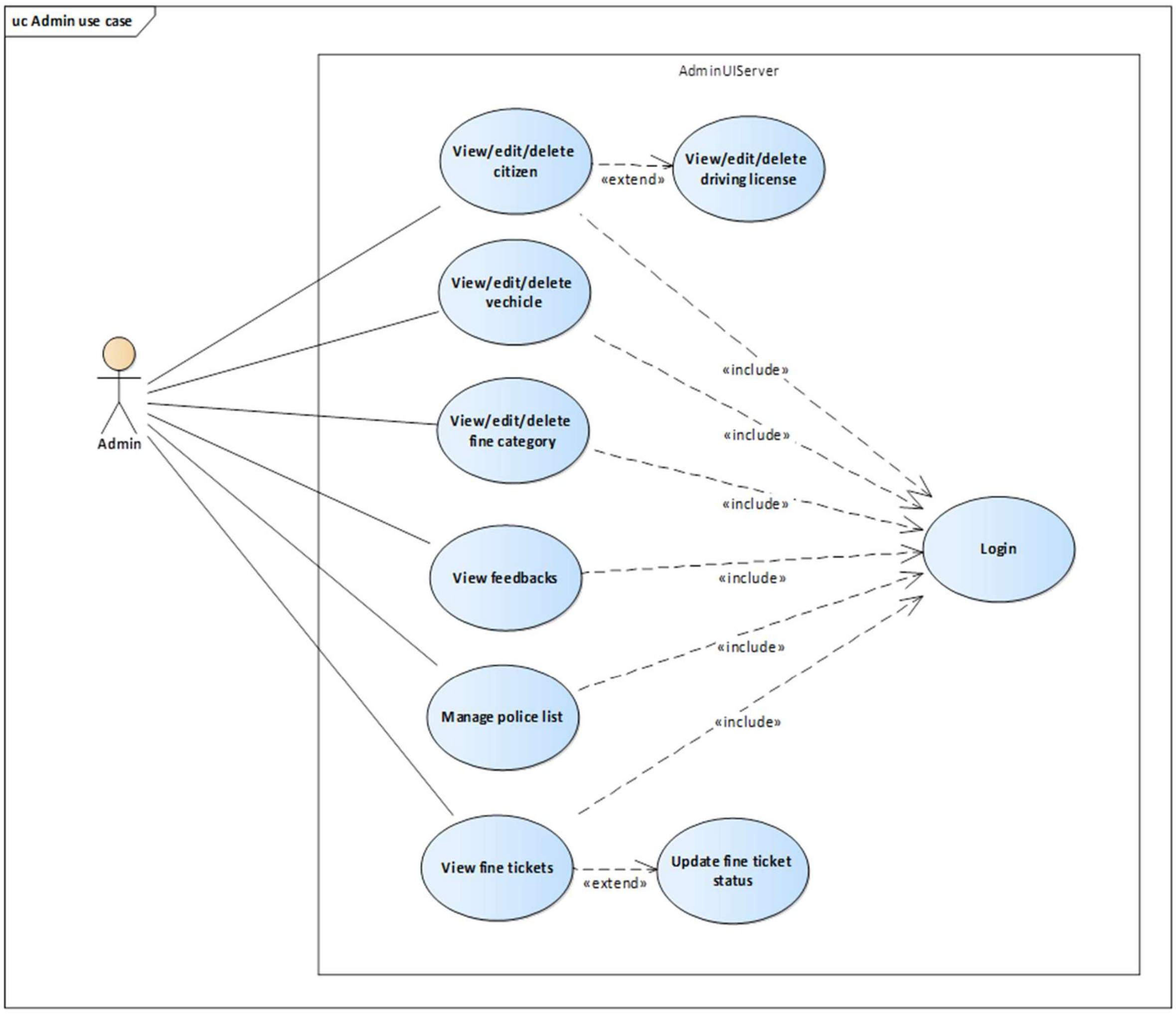


Figure 8. System admin use case diagram.

Above three diagrams show the basic functionality of the solution. There are other use cases which are not mentioned here, such as check vehicle information, search payment station, payment procedure. More detail on the application can be found in the appendices or in the following chapters.

3.3 Software structure and components

Below figure 9 describes the general setup of the solution as well as main basic components. On top layer, there is an E-ticket device with 3G connection. At the center, there is IoT cloud with its components, including REST server and two UI hosting servers. The bottom layer represents end users. There are two groups of users with different access rights: system admin and normal user.

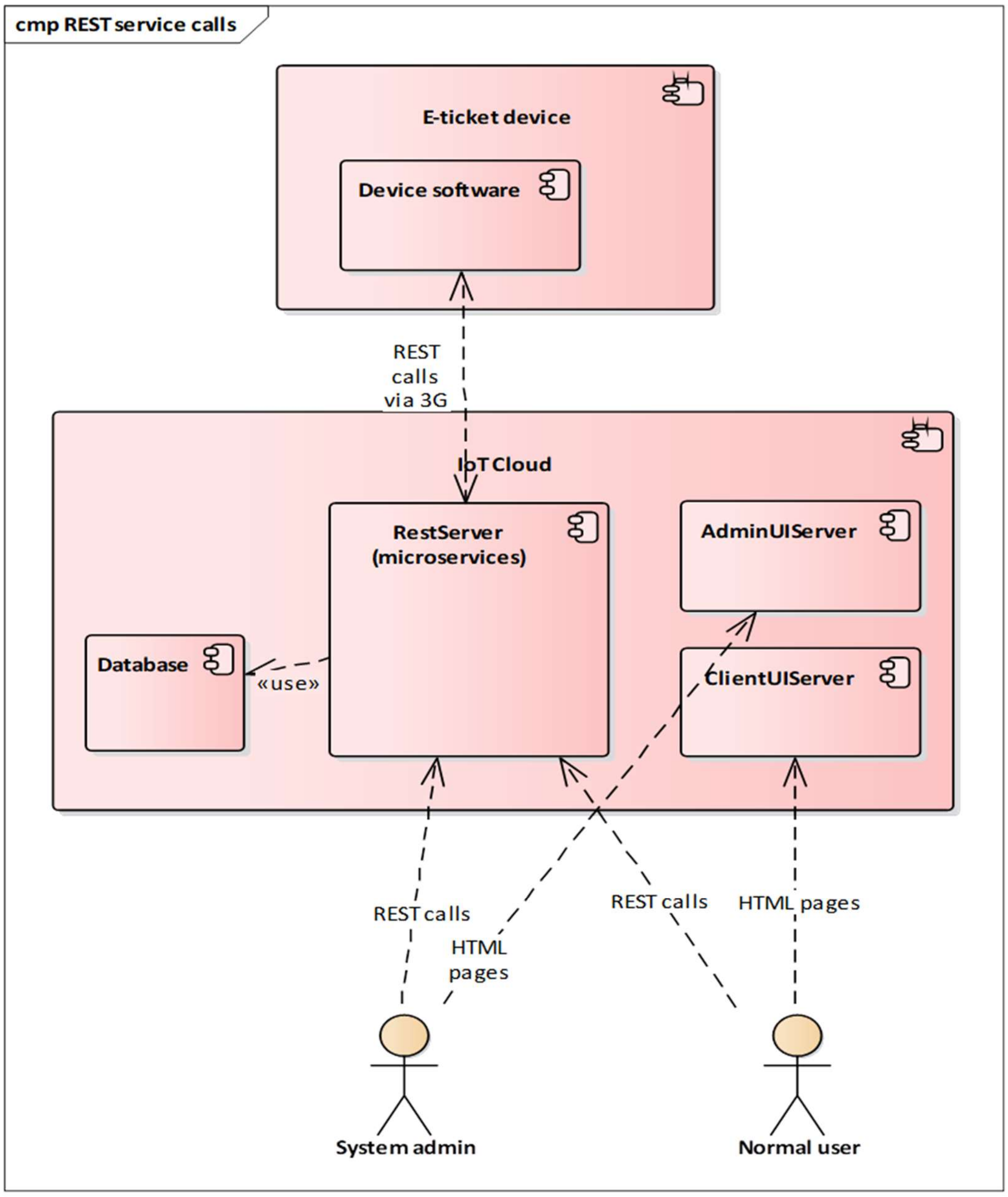


Figure 9. Main software components.

IoT cloud consists of 3 running servers.

REST server: the server is actually a collection of microservices. They provide REST web service for device software application and for the two web applications. They have their own database.

- AdminUIServer: nodejs server providing the HTML pages to end users via a web browser. It does not serve any data. Instead, the web application gets its data via REST web services mentioned above. This server is targeted for system admin and available only under VPN connection and with sufficient user privileges.
- ClientUIServer: same as AdminUIServer but this server provides HTML pages for normal users. The server is publicly available for all users (vehicle owners).

Device software communicates with REST server via REST API calls. Upon an event, for example issuing new ticket, the device software will enable its 3G connection and make REST calls to REST server to query data or to create new fine ticket. After the process is done, the device will close the 3G connection. The device's SIM card is a special one with VPN connection built-in. Once connected, the device is in the same VPN network with the private section of IoT cloud server. On top of that, the transferred data between them is encrypted using SSL. This ensures the connection from E-ticket device to IoT server is secure.

3.4 Application security

Security is a very important aspect of this IoT application. Amazon cloud service AWS [8] was selected to host this application. As the result, all security settings for the application are based on Amazon AWS. Figure 10 illustrates the security setup of E-ticket solution.

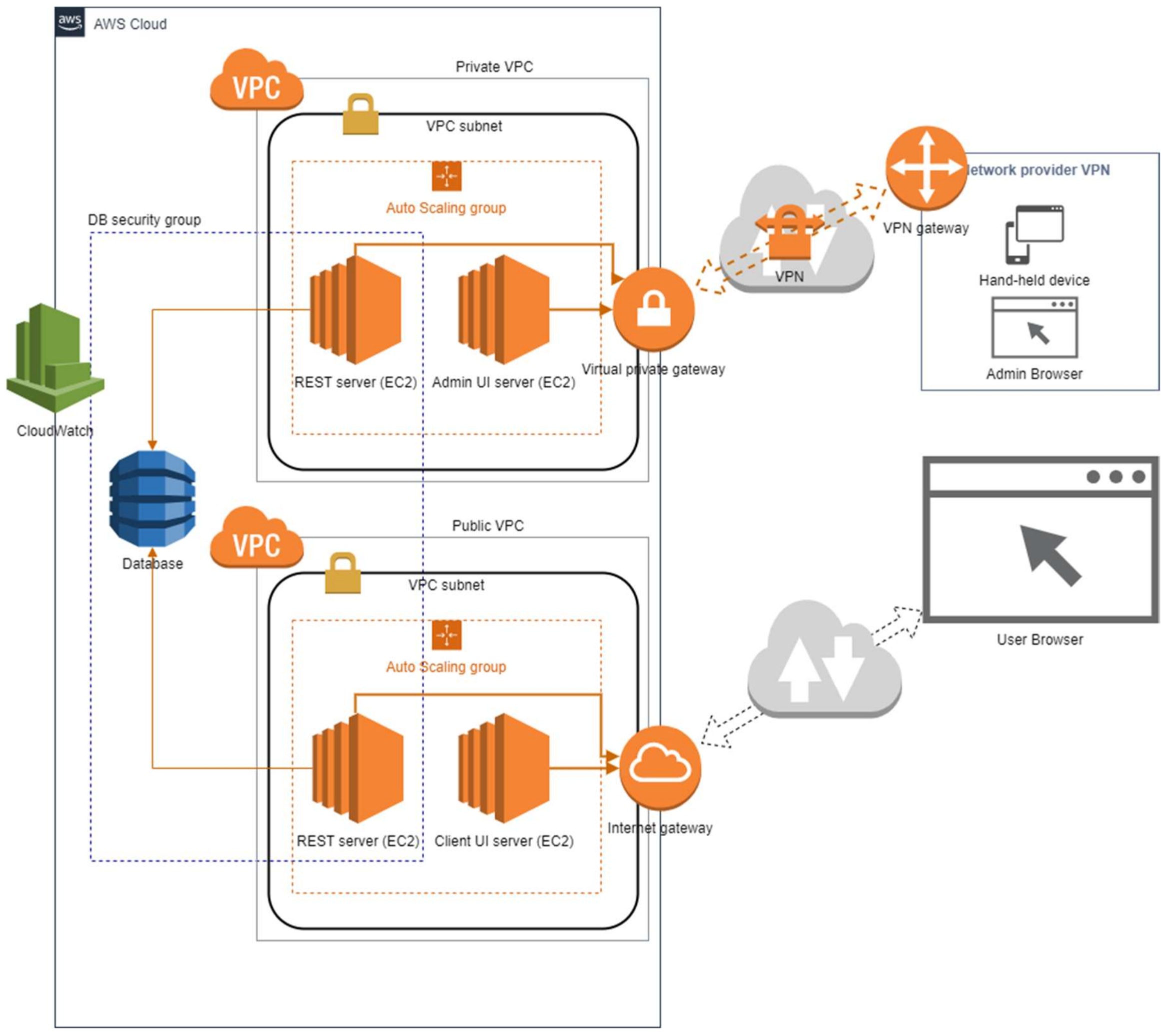


Figure 10. Solution security setup.

Since there are two separate use cases for this application, secured connection with handheld device and public web application access for normal user, the application is designed and deployed in two different VPCs [24]: public VPC and private VPC. Within each VPC, micro-services are deployed in different EC2s [25] inside a VPC subnet. This provides maximum security for the solution: each application is restricted in its own EC2 sandbox and is not allowed to access elsewhere.

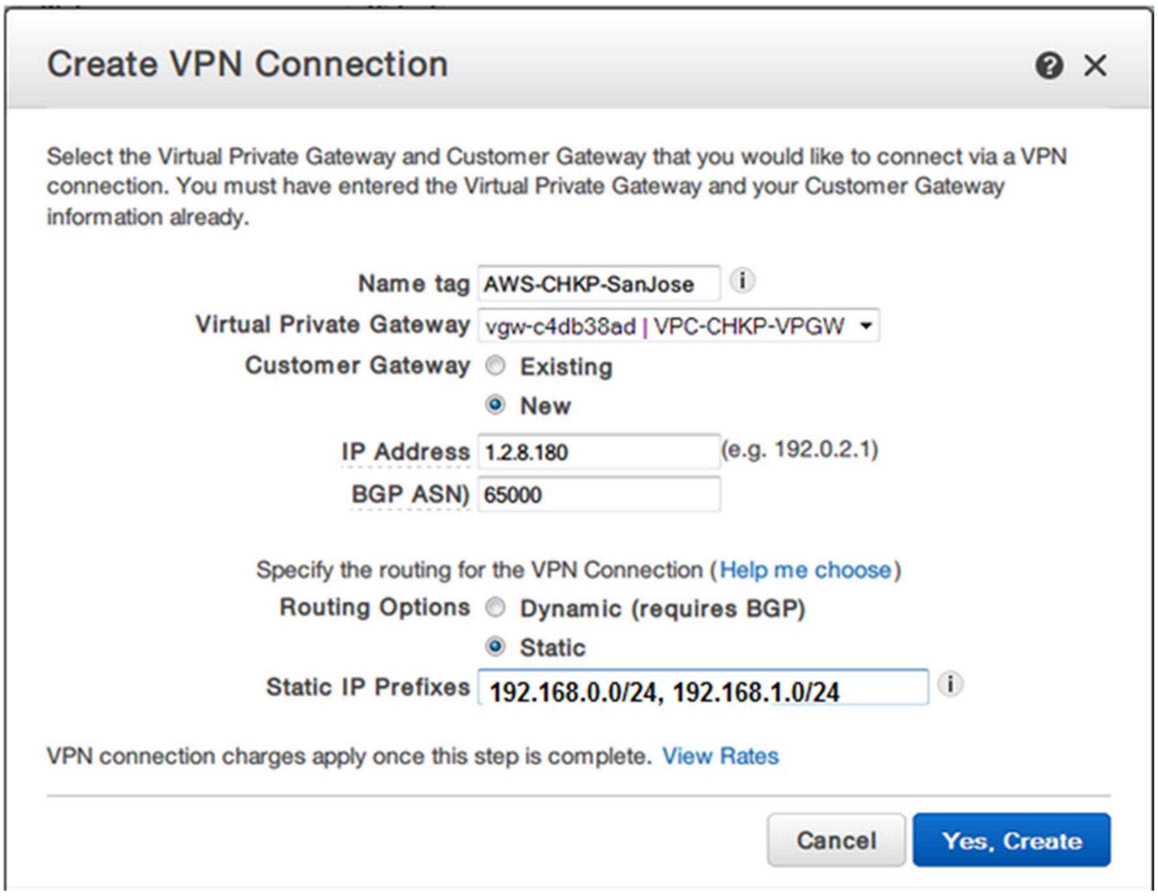
- Public VPC: hosts REST server including micro-services and Client UI server. All of them are public services and are available over internet via an internet gateway.
- Private VPC: hosts REST service including micro-services and Admin UI server. They are private services and communicate with outside via a Virtual private gateway. The gateway is configured to have bridged connection with the VPN network provided by network provider. This setup allows a secure connection from hand-held device to private VPC services.

Both REST servers use the same database. This is the only shared information between both VPCs. The security for database connection is done by DB security group. Both REST servers and database are part of this security group. Only member of this group can access the database.

The traffic of the whole solution is monitored by Amazon CloudWatch component. It is configured to log input and output data to the VPC via either virtual private gateway or internet gateway. It can be used to diagnose communication issue or detect fishing traffic. Moreover, it also allows setting alarm on the system if certain condition is satisfied.

Moreover, the solution supports clustering and auto-scaling. Each EC2 has auto-scaling feature to duplicate itself to multiple instances in case the coming traffic is high and reduce the instances in case of low traffic.

Below figure 11 shows configuration step to configure bridged VPN connection between private VPC and VPN of network provider.


Create VPN Connection

Select the Virtual Private Gateway and Customer Gateway that you would like to connect via a VPN connection. You must have entered the Virtual Private Gateway and your Customer Gateway information already.

Name tag	AWS-CHKP-SanJose
Virtual Private Gateway	vgw-c4db38ad | VPC-CHKP-VPGW
Customer Gateway	Existing [✓ New]
IP Address	1.2.8.180 (e.g. 192.0.2.1)
BGP ASN)	65000

Specify the routing for the VPN Connection (Help me choose)

Routing Options	Dynamic (requires BGP) [✓ Static]
Static IP Prefixes	192.168.0.0/24, 192.168.1.0/24

VPN connection charges apply once this step is complete. View Rates

Figure 11. Bridged VPN connection setup.

Bridged VPN configuration allows machines within a VPC connected to VPN network of network provider. The bridged VPN configuration is easy: two most important fields are Virtual private gateway we want to map and the IP address of the VPN gateway of the network provider. Moreover, it also allows configuring IP address range of EC2s within the VPC.

4 Implementation

4.1 Device software

4.1.1 Components overview

Device software is the software running on the E-ticket device. It implements perception layer and one side of the network layer in IoT architecture we discussed in literature review chapter. The software application is written in C language and run under Linux OS on ARM processor.

The application collects inputs from policeman and connects to server via 3G to send the command creating a fine ticket. After that it prints out the fine ticket to be given to the person committed to the fault. The application is also used to query and check vehicle data such as inspection info, owner or driver data such as contact info, driving license, unpaid tickets, etc.

Detail information about device components and services is listed in below figure 12.

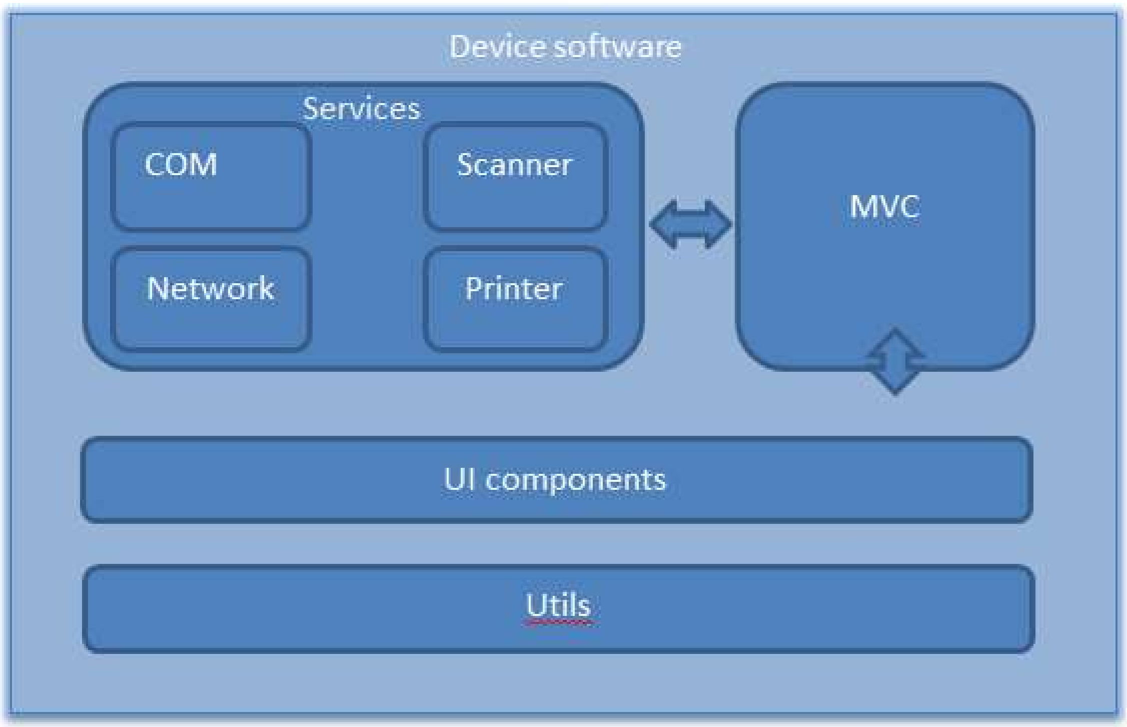


Figure 12. Device software components outline.

The application consists of 4 main packages: UI components, services, mvc and utils. Each package is made as a separate module with own functionality. Detail description on the services is as the following.

- UI components: this package contains basic UI components of a page such as button, label, textfield, etc. Since the application UI is built based on directfb, which is quite low level framework, all UI components need to be built from scratch.
- Services package: this package contains services needed for the application to interface with the hardware and communicate with the server. There are four services. Scanner service is used to scan barcode. Printer service is for printing tickets. Network service allows configuring 3G connection. COM service is responsible for REST communication with the IoT server.
- MVC (model – view - controller): this package contains the main UI application code. It is responsible for displaying UI pages together with logic of switching between pages and handle user actions. It composes the pages from the UI components and uses needed services from service package.
- Utils: utility package.

4.1.2 Components interfacing

The components interact with each other via API function calls. MVC is the main component; other services from services package are called from MVC when required. Figure 13 describes the interaction between MVC and services.

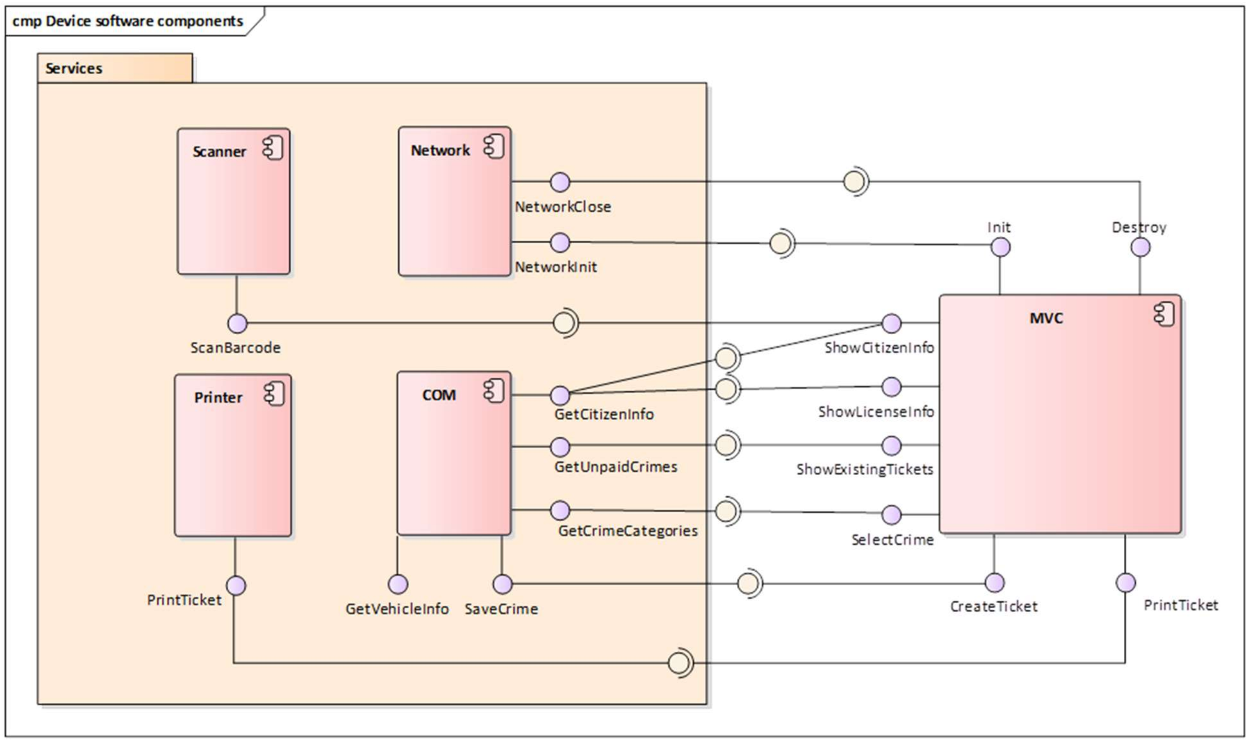


Figure 13. Interface between MVC and services.

As can be seen from above figure, services are independent and only interface with MVC. Each service handles a respective job. Below is the detail on each service and its functions.

- Scanner service is used by MVC to scan driver's license number.
- Printer service provides a function to print newly created ticket.
- COM service is the REST service providing user data such as: get citizen information, get citizen's unpaid tickets, get categories, and get vehicle information. This service is the main communication channel between the device and IoT cloud server. It also implements SSL encryption to secure the connection between them.
- Network service is used to initialize or close the data connection. It also handles the re-connection when connection interruption happens. In the background, it uses pppd agent, which is a process controlling the modem via a collection of AT commands called chatscript. Network service uses a configuration file to store network provider configuration (username and password). Below is the chat script network service uses to initiate the 3G connection.

```
/**
 * Chatscript to issue AT command and check the reply.
 * For different provider, the APN(internet), callnumber(*99#),
 * and username/password shall be updated accordingly.
 * Read more about chatscript in ppp website
 */
char GPRS_CHAT_SCRIPT[] = {
                "ABORT        'NO CARRIER'"                       "\n"
                "ABORT        'NO DIALTONE'"                      "\n"
                "ABORT        'ERROR'"                            "\n"
                "ABORT        'NO ANSWER'"                        "\n"
                "ABORT        'BUSY'"                                         "\n"
                "ABORT        'Invalid Login'"         "\n"
                "ABORT        'Login incorrect'"       "\n"
                "TIMEOUT       '60'"                                          "\n"
                ""'          'ATZ'"                                           "\n"
                "'OK'         'AT+CGDCONT=1,\"IP\",\"internet\"'"  "\n"
                "'OK'         'ATDT*99#'"                         "\n"
                "'CONNECT'     ''"
                "\n"
};
```

4.1.3 Sequence diagram

The main usage of E-ticket device is to check vehicle info, driver info and issue fine tickets. Examples of vehicle information which is checked are vehicle year, owner and inspection data. Vehicle information is retrieved based on the license plate number. For a driver, information such as name, age, address, driving license is checked. Driver information is fetched based on his social security number. The following figure 14 shows the sequence when police issues a fine ticket.

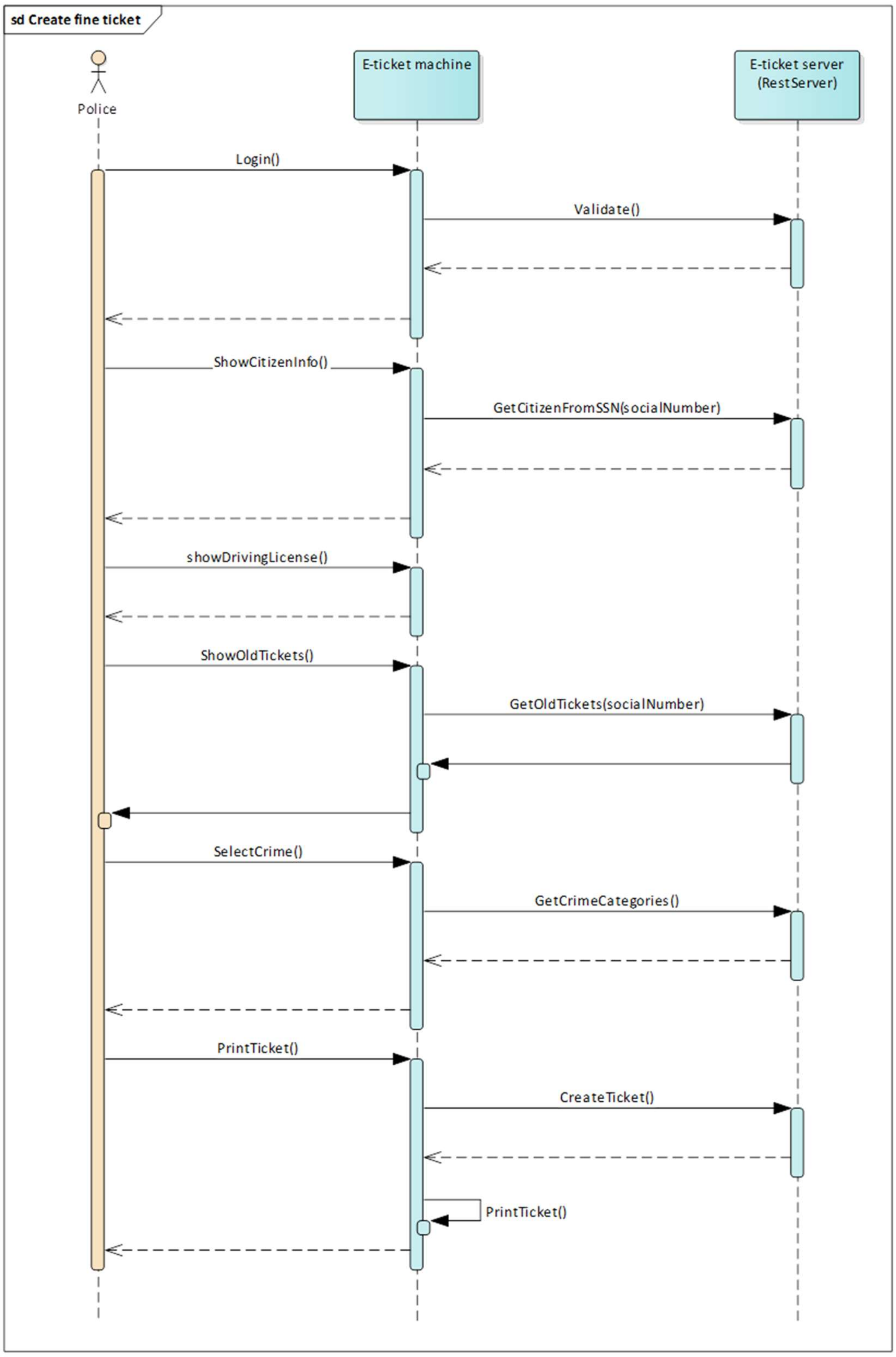


Figure 14. Create fine ticket sequence diagram.

E-ticket machine is used by police officer to issue fine tickets. The device does not constantly have the 3G Ethernet connection to IoT could server; the connection is switched on only when needed and off after the action is done. However, it is not described in above diagram to reduce the complexity. In order to access E-ticket functionality, at first police officer needs to login using his own credentials. Secondly, he scans the driver's driving license number and fetches the driver's info and unpaid tickets. Finally, he selects a new fault for the driver and prints a new ticket. If there are unpaid tickets, all of them will be summed up in the newly created ticket.

4.2 Server-side software

4.2.1 Components overview

As mentioned in theory part, the cloud server of this solution is implemented using Spring Boot [20] and Spring Cloud [21] frameworks. It utilizes the implementation of basic cloud services instead of reinventing the wheel. This allows the application to inherits all the nice features of IoT cloud service such as clustering, load balancing, service gateway, etc. Below figure 15 describes in detail the implemented services and their purposes.

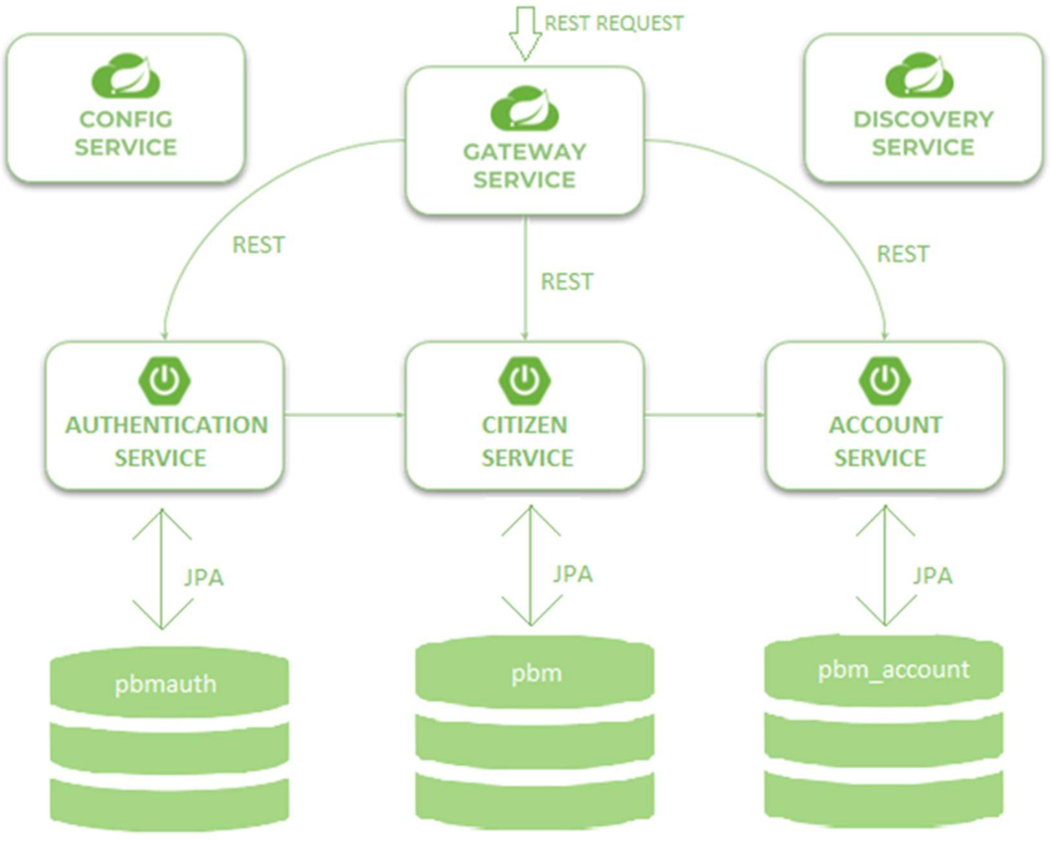


Figure 15. Architectural design of E-ticket cloud.

There are several micro services which form this cloud-based server side application. Each micro service is independent and does distinctive jobs. It also have its own database connection and own data table. JPA [26] and Hibernate [27] are the software components used to handle database connection and database transaction. Below is a short description of the implemented microservices.

- Config service: generic service storing configuration, eg: database configuration.
- Discovery service: generic service storing services and their names.
- Gateway service: generic cloud service acting as an entry point to the whole system. It asks discovery service for the correct service to handle incoming request and passes the request to it.
- Authentication service: handle user authentication.
- Account service: handle user account.
- Citizen service: main service providing all information related to a citizen: citizen info, driving license info, fine category, fine ticket, vehicle info, feedback, etc.

4.2.2 Sequence diagram

A driver having a fine ticket can make the payment for it via bank transfer or via E-ticket website. The information for paying the fine ticket (amount, bank account, reference number) is already present in the paper ticket itself. However, in case the paper ticket is lost, driver can check their tickets based on their social security number on E-ticket website. Moreover, the website provides more information about the fine, such as incident time, evidence photo of the fault, lawful information about the fault, nearest debt collection station. In addition, it allows user to make payment on the website and give feedback about performance of the police officer.

The following figure 16 shows the sequence when a driver searches and makes payment via E-ticket website.

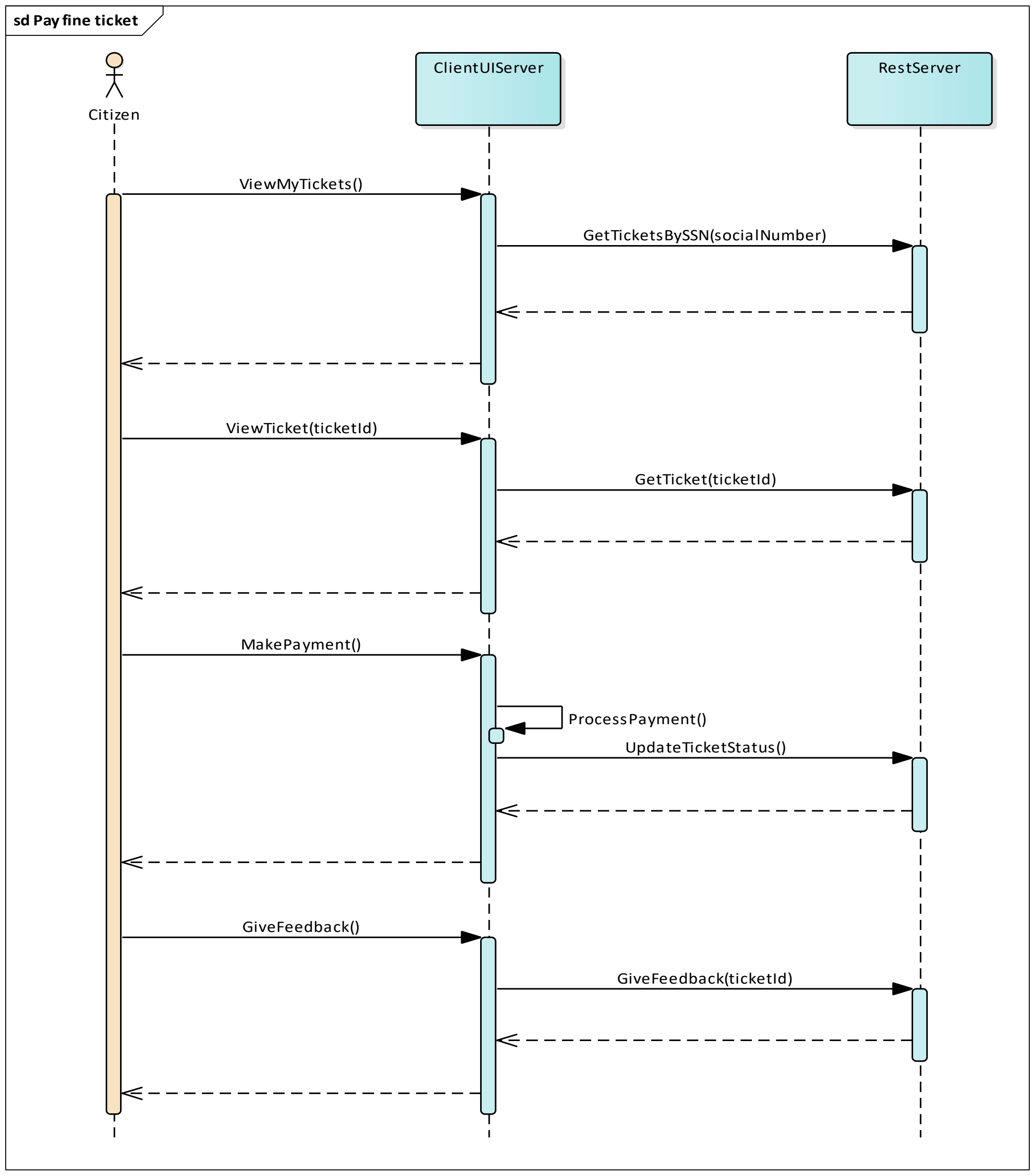


Figure 16. Search and make payment sequence diagram.

Normal users don't need to login to search for their ticket information. Instead, the system uses user information, such as social security number or fine ticket number, to search for existing fine tickets. The search result gives a list of paid and unpaid fine tickets. User can click on a ticket to view its detail: incident time, evidence photo of the fault, lawful information about the fault, nearest debt collection station. After that, user can choose to pay for the fine ticket, the system will navigate user to payment gateway page to make the payment. Finally, user can give feedback about police performance once the payment process is done.

4.2.3 Implementation

The solution consists of multiple microservices. Each has each own role described in previous chapter. The general structure all microservices is described in the following.

- Config service

Config service is a generic Spring Cloud service. There is no customization to it. The code base of this service is also very simple through Spring Cloud framework. The key in code implementation is @EnableConfigService annotation which indicates that this is a config service.

```java
package com.latekco.config;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.config.server.EnableConfigServer;

@SpringBootApplication
@EnableConfigServer
public class PbmConfigApplication
{
    public static void main(String[] args) {
        SpringApplication.run(PbmConfigApplication.class, args);
    }
}
```

Config service manages configuration files for other services. The service is a central place to configure and manage all other microservices. This simplifies the maintenance work. Moreover, it also allows editing the configuration of all applications without the need of restarting them. Examples of configuration include database connection URL, security settings, etc. Below is example of a configuration file.

```yaml
security:
  oauth2:
    client:
      clientId: pbm-account-service
      accessTokenUri: http://localhost:5000/uaa/oauth/token
    resource:
      tokenInfoUri: http://localhost:5000/uaa/oauth/token
      clientId: pbm-account-service
spring:
  jackson:
        serialization.indent_output: true
  datasource:
      driver-class-name: com.mysql.jdbc.Driver
      url: jdbc:mysql://localhost:3306/pbm_account
      username: root
      password: root

  jpa:
      database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
      database: MYSQL
      show-sql: true
```

❖ Gateway service

Gateway service is the interface to the application. It is responsible for routing requests to a correct service and balance the load among services of the same type. When receiving a request from outside, it will recognize needed service name, and ask discovery service to provide corresponding available service based on that. This is also a generic Sping service with minor modification.

❖ Discovery service

Discovery service stores the information about all available micro-service applications. Every micro-service registers itself to discovery service. As a result, this service knows every micro-service application running on each port and IP address. This is a generic Spring service with minor modification.

❖ Account service

Account service is responsible for creating and managing user accounts. The account is granted for police officers and administration staff. Account service has own database. Below is example code to create a new account.

```
@Override
public Account create(UserDc user) {

        Account existing = repository.findByName(user.getUsername());
        Assert.isNull(existing, "account already exists: " + us-
er.getUsername());

        authClient.createUser(user);
        Account account = new Account();
        account.setName(user.getUsername());
        account.setLastSeen(new Date());

        repository.save(account);

        return account;
}
```

❖ Authentication service

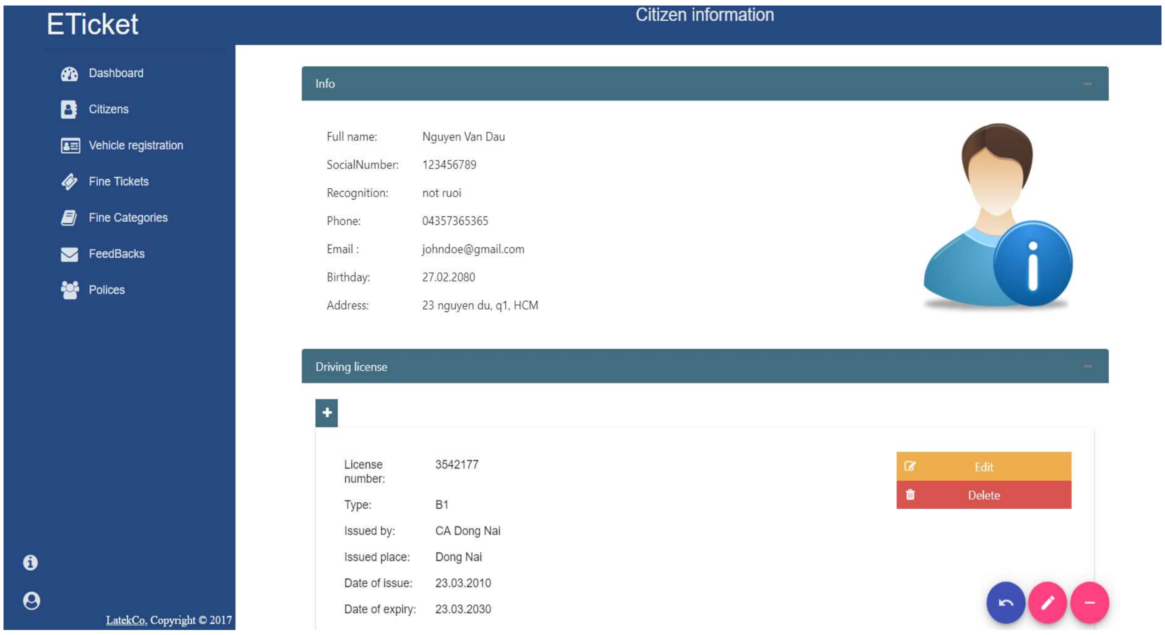
Authentication service is responsible for implementing login and user authentication service. It is also responsible for security layer of the application.

❖ Citizen service

Citizen service is the main service providing data for both E-ticket device and E-ticket web application. It provides all the functionalities that are available in E-ticket device and E-ticket web application. It contains the following controllers: citizen controller, driving license controller, vehicle controller, policeman controller, fine category controller, fine ticket controller, billing machine controller, and feedback controller.

5 Result

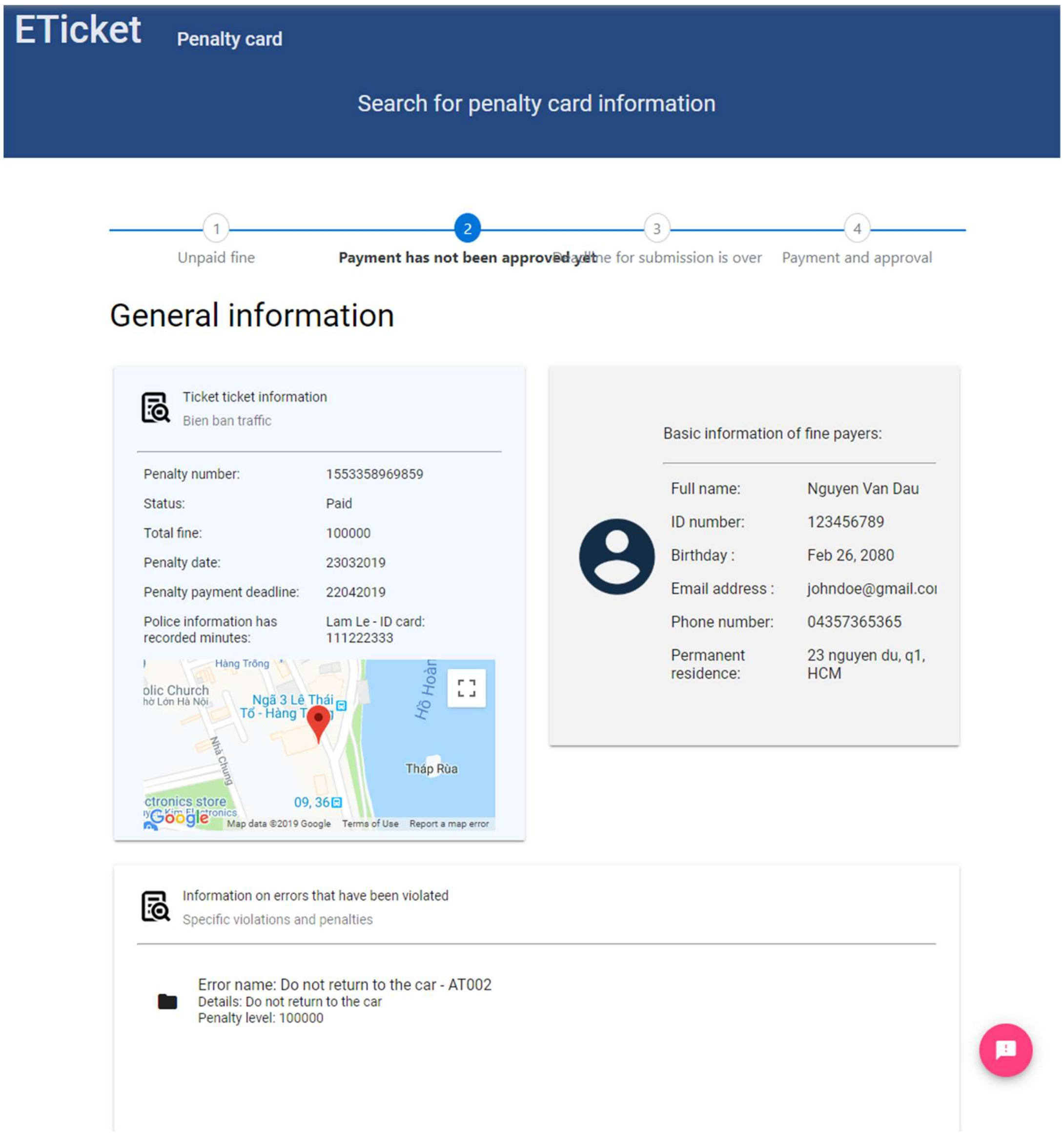
The user interface of this E-ticket application was made using latest web technologies such as Angular JS, Bootstrap, Awesome font, jquery, typescript, HTML 5, CSS 3. Moreover, sockjs-client was used to create websocket connection between web browser and the IoT server. It allows the client side to detect new changes from the server side and update accordingly. Below figures 17 and 18 demonstrate the interfaces in both admin page and user page. More detail user interfaces can be found in the appendices.



Full name:	Nguyen Van Dau
SocialNumber:	123456789
Recognition:	not ruoi
Phone:	04357365365
Email :	johndoe@gmail.com
Birthday:	27.02.2080
Address:	23 nguyen du, q1, HCM

License number:	3542177
Type:	B1
Issued by:	CA Dong Nai
Issued place:	Dong Nai
Date of issue:	23.03.2010
Date of expiry:	23.03.2030

Figure 17. Citizen detail page.



ETicket Penalty card

Search for penalty card information

1 Unpaid fine
2 Payment has not been approved yet
3 Deadline for submission is over
4 Payment and approval

General information

Ticket ticket information

Bien ban traffic

Penalty number:	1553358969859
Status:	Paid
Total fine:	100000
Penalty date:	23032019
Penalty payment deadline:	22042019
Police information has recorded minutes:	Lam Le - ID card: 111222333

Hàng Trống
olic Church
hờ Lớn Hà Nội
Ngã 3 Lê Thái Tổ - Hàng T
Nhà Chung
Hồ Hoàn
Tháp Rùa
ctronics store
09, 36
Google
Map data ©2019 Google Terms of Use Report a map error

Basic information of fine payers:

Full name:	Nguyen Van Dau
ID number:	123456789
Birthday :	Feb 26, 2080
Email address :	johndoe@gmail.coı
Phone number:	04357365365
Permanent residence:	23 nguyen du, q1, HCM

Information on errors that have been violated

Specific violations and penalties

Error name: Do not return to the car - AT002
Details: Do not return to the car
Penalty level: 100000

Figure 18. Fine ticket detail.

6 Conclusion

IoT envisions the future Internet where every physical device is connected to the Internet and exchanges information about themselves and their surroundings. This thesis discussed the general structure of IoT application, existing IoT applications and future development. The result of this research suggests that IoT has very big potential on developing smart applications in almost every field in our daily life.

The demonstration of E-ticket system in this thesis has given a concrete example of IoT application development and IoT use cases solving real life problems. Although the application is not yet taken into production, it presents a demonstration of ways to improve existing public service in developing countries using IoT. The application shall be developed further and sold to government to take it into use.

References

[1] Debasis Bandyopadhyay and Jaydip Sen. Internet of Things: Applications and Challenges in Technology and Standardization. Wireless Pers Commun (2011) 58:49–69.

[2] Eleonora Borgia. The Internet of Things vision: Key features, applications and open issues. Institute of Informatics and Telematics (IIT), Italian National Research Council (CNR), via G. Moruzzi 1, 56124 Pisa, Italy.

[3] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, Volume 29, Issue 7, 2013.

[4] Khan, R., Khan, S. U., Zaheer, R., & Khan, S. (2012). Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In 2012 10th International Conference on Frontiers of Information Technology (FIT): Proceedings (pp. 257-260). Institute of Electrical and Electronics Engineers Inc. DOI: 10.1109/FIT.2012.53.

[5] Gardašević G, Veletić M, Maletić N, Vasiljević D, Radusinović I, Tomović S, et al. The IoT Architectural Framework, Design Issues and Application Domains. Wireless Personal Communications. 2017;92(1):127-48.

[6] Smart Santander, EU FP7 project. Future internet research and experimentation. *http://www.smartsantander.eu/*.

[7] Bharath Mannaperumal. Microservice using Sping Cloud and Netflix OSS. *https://jsoftgroup.wordpress.com/2017/05/09/micro-service-using-spring-cloud-and-netflix-oss/*

[8] Amazon web services (AWS). Retrieved 11.11.2019, from *http://www. https://aws.amazon.com/*.

[9] Microservices with Spring. Retrieved 11.11.2019, from *https://spring.io/blog/2015/07/14/microservices-with-spring*.

[10] What is REST. Retrieved 11.11.2019, from *https://restfulapi.net/*.

[11] What is 3G. Retrieved 11.11.2019, from *https://www.megapath.com/blog/blog-archive/what-is-3g-network/*.

[12] Bernard Marr. What is Industry 4.0? Retrieved 11.11.2019, from *https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/#3a27b3d39788 /*.

[13] What is Cloud computing? Retrieved 11.11.2019, from *https://aws.amazon.com/what-is-cloud-computing/*.

[14] Supercomputer wikipedia Retrieved 11.11.2019, from *https://en.wikipedia.org/wiki/Supercomputer/*.

[15] What is Zigbee? Retrieved 11.11.2019, from *https://zigbee.org/what-is-zigbee//*.

[16] Bluetooth Low Energy (BLE). Retrieved 11.11.2019, from *https://www.novelbits.io/what-is-ble-bluetooth-low-energy-iot//*.

[17] Diamond A. K. Asare. Body Area Network Standardization, Analysis and Application.

[18] What are microservices? Retrieved 11.11.2019, from *https://microservices.io//*.

[19] Quick Guide to Microservices With Spring Boot 2.0, Eureka, and Spring Cloud. Retrieved 11.11.2019, from *https://dzone.com/articles/quick-guide-to-microservices-with-spring-boot-20-e*.

[20] Spring Boot. Retrieved 11.11.2019, from *https://spring.io/projects/spring-boot/*.

[21] Spring Cloud. Retrieved 11.11.2019, from *https://spring.io/projects/spring-cloud*.

[22] OAuth 2.0. Retrieved 11.11.2019, from *https://oauth.net/2/*.

[23] Building an Application with Spring Boot. Retrieved 11.11.2019, from *https://spring.io/guides/gs/spring-boot/*.

[24] Amazon VPC. Retrieved 11.11.2019, from *https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html*.

[25] Amazon EC2. Retrieved 11.11.2019, from *https://aws.amazon.com/ec2/*.

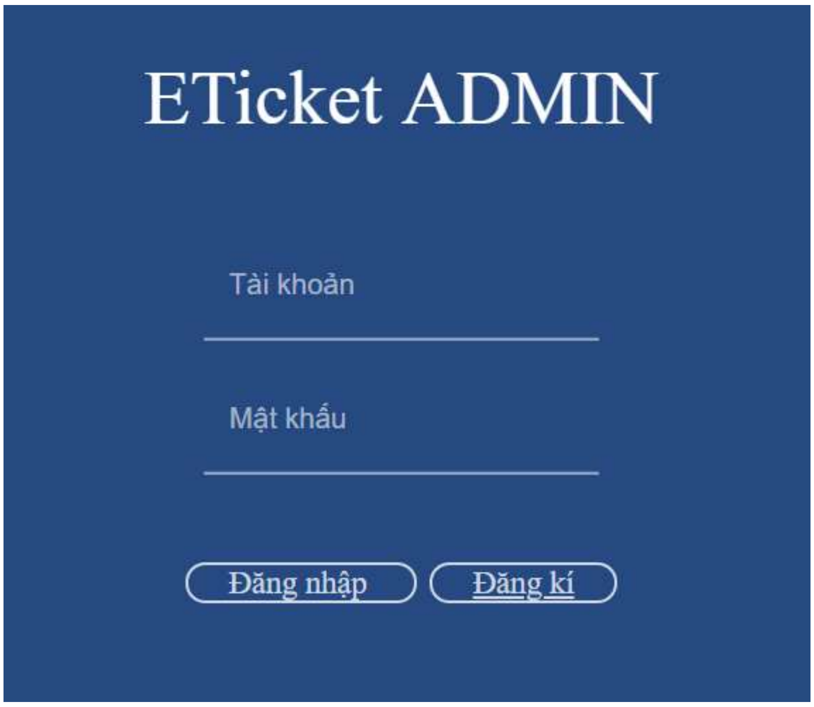
[26] What is JPA? Introduction to the Java Persistence API. Retrieved 11.11.2019, from *https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html*.

[27] Hibernate. Retrieved 11.11.2019, from *https://hibernate.org/*.

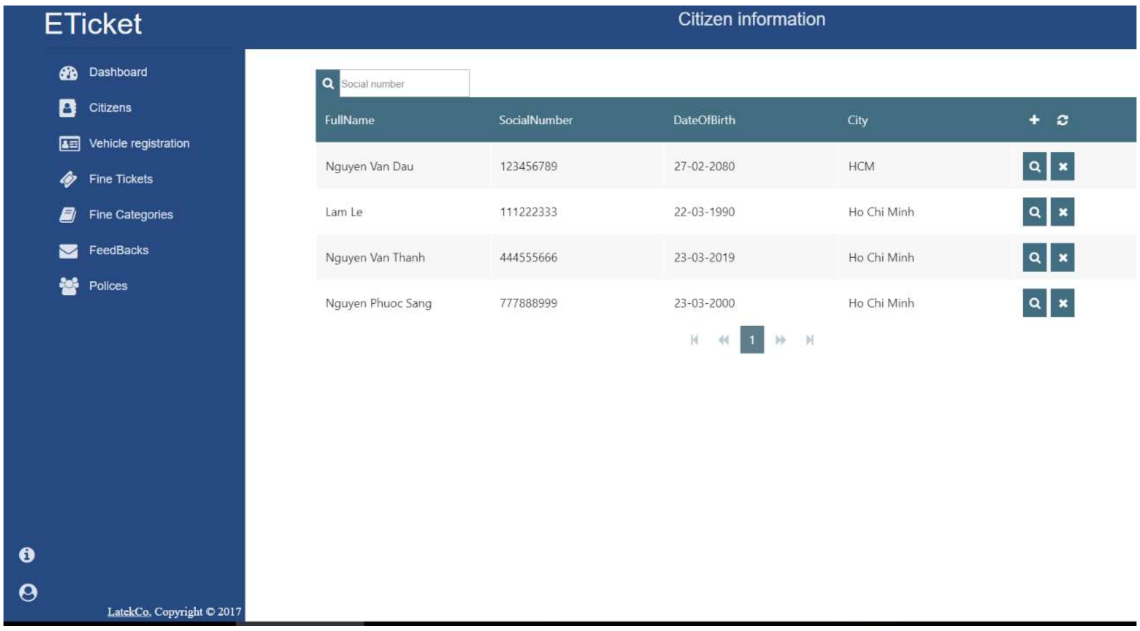
Appendices

Appendix 1: User interface of E-ticket Admin.

Login page:

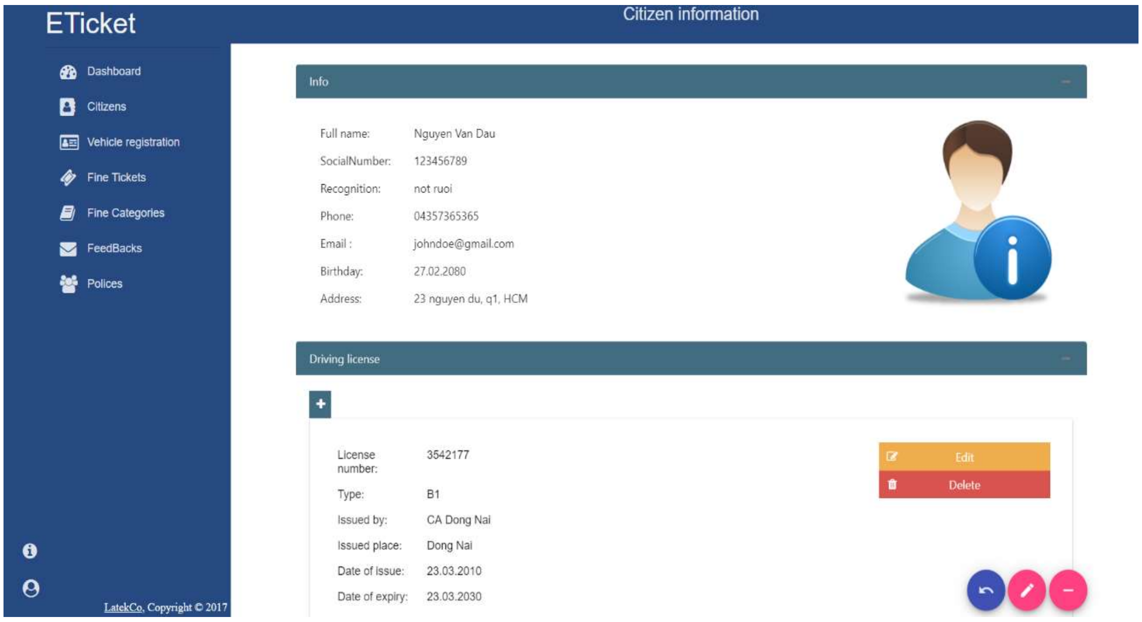


Citizen page: view all citizen and edit citizen.



FullName	SocialNumber	DateOfBirth	City
Nguyen Van Dau	123456789	27-02-2080	HCM
Lam Le	111222333	22-03-1990	Ho Chi Minh
Nguyen Van Thanh	444555666	23-03-2019	Ho Chi Minh
Nguyen Phuoc Sang	777888999	23-03-2000	Ho Chi Minh

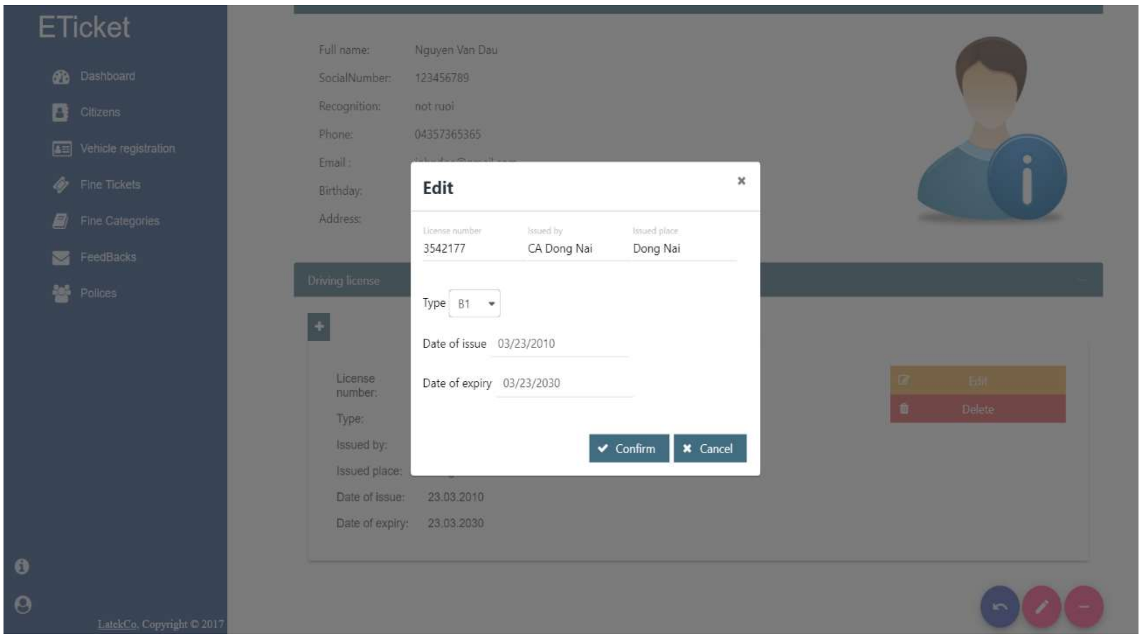
View citizen information.



Full name: Nguyen Van Dau
SocialNumber: 123456789
Recognition: not ruoi
Phone: 04357365365
Email : johndoe@gmail.com
Birthday: 27.02.2080
Address: 23 nguyen du, q1, HCM

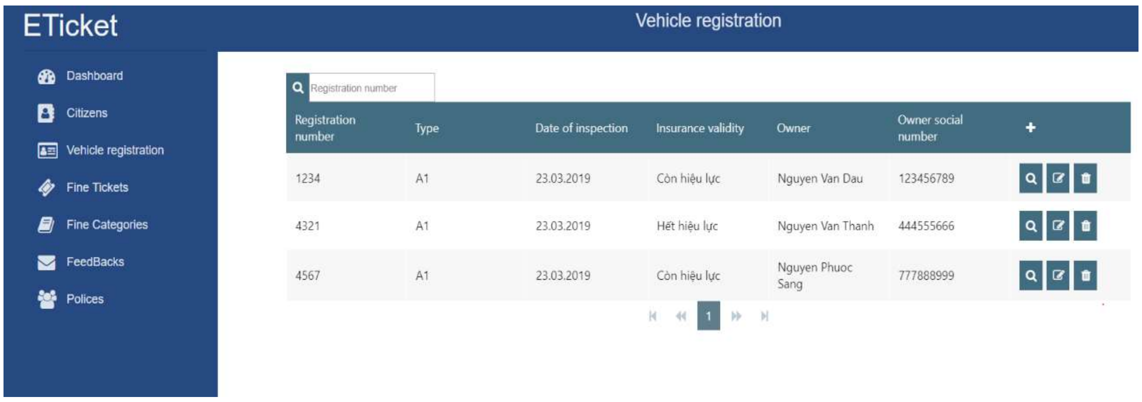
License number: 3542177
Type: B1
Issued by: CA Dong Nai
Issued place: Dong Nai
Date of issue: 23.03.2010
Date of expiry: 23.03.2030

Edit or add driving license.



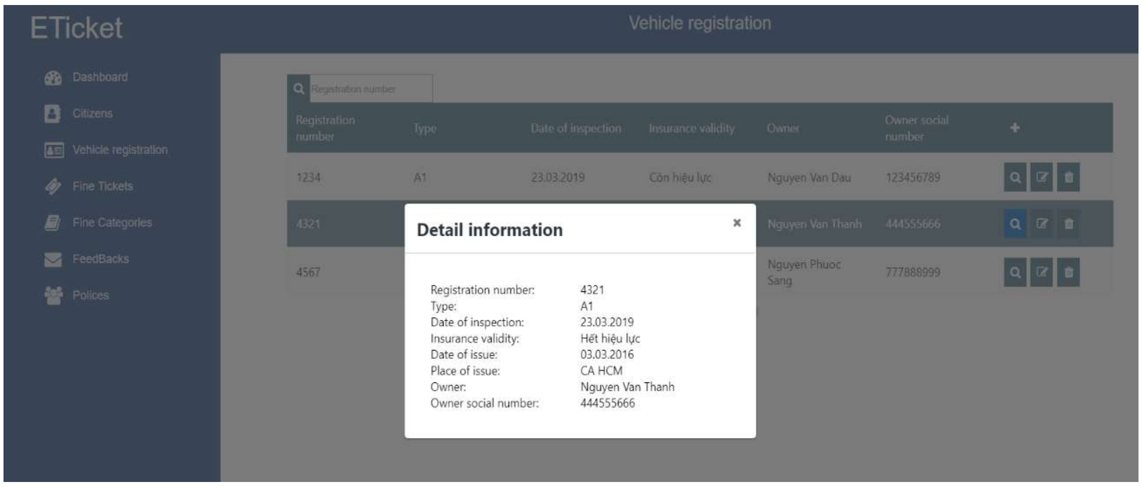
License number: 3542177
Issued by: CA Dong Nai
Issued place: Dong Nai
Type: B1
Date of issue: 03/23/2010
Date of expiry: 03/23/2030

Vehicle page: view all vehicle and edit vehicle information.



Registration number	Type	Date of inspection	Insurance validity	Owner	Owner social number	+
1234	A1	23.03.2019	Còn hiệu lực	Nguyen Van Dau	123456789	
4321	A1	23.03.2019	Hết hiệu lực	Nguyen Van Thanh	444555666	
4567	A1	23.03.2019	Còn hiệu lực	Nguyen Phuoc Sang	777888999	

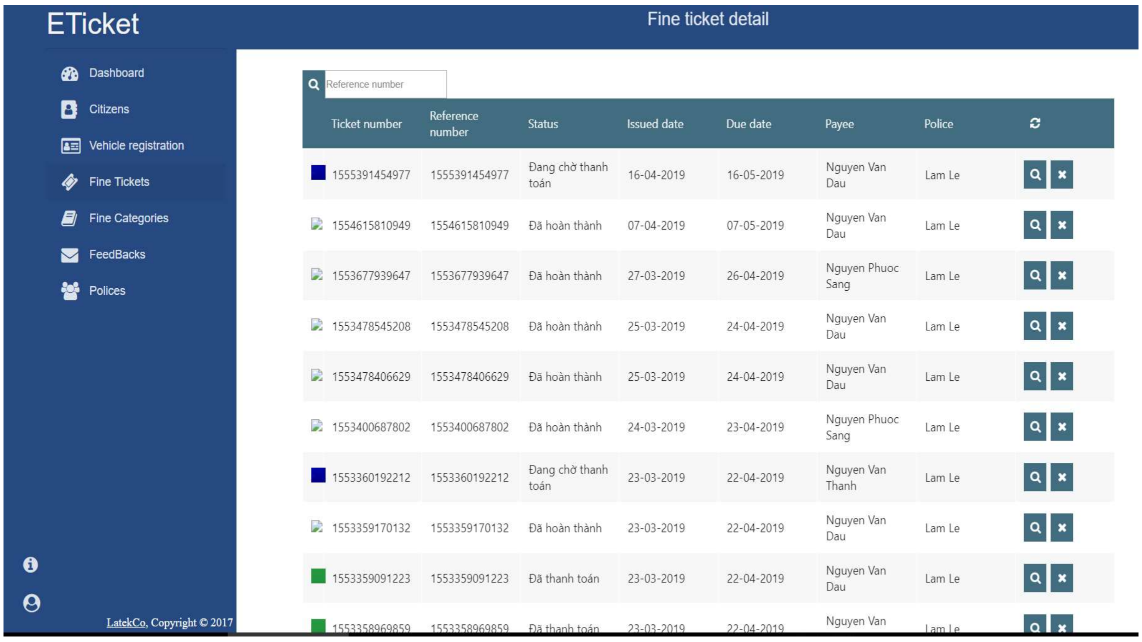
Edit or add vehicle information:



Detail information

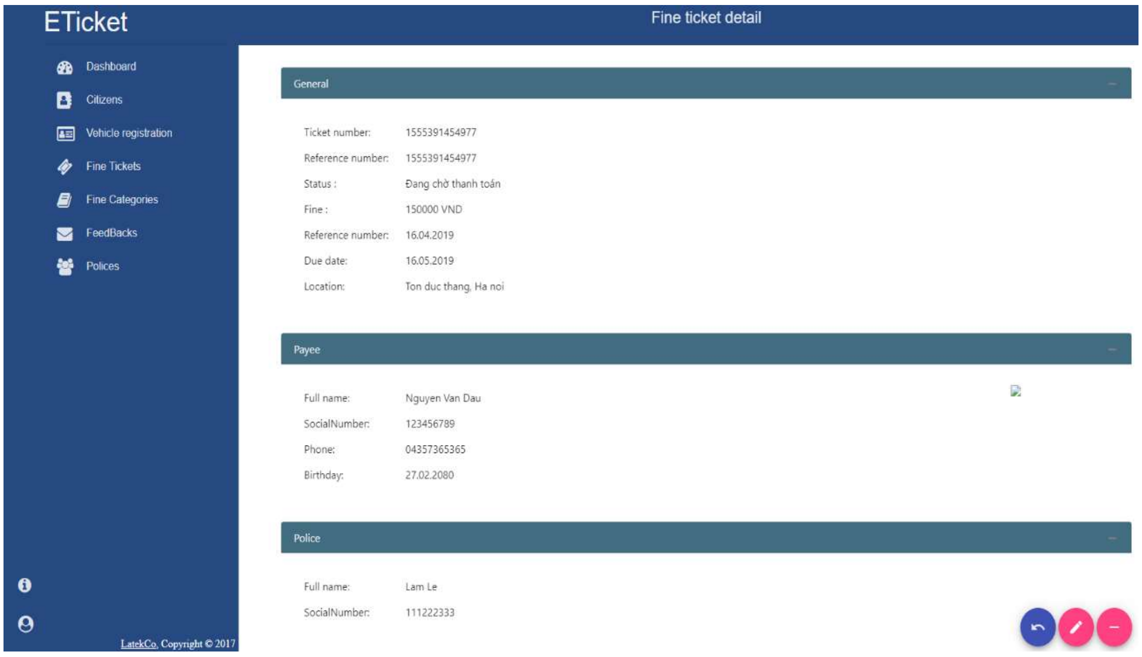
Registration number:	4321
Type:	A1
Date of inspection:	23.03.2019
Insurance validity:	Hết hiệu lực
Date of issue:	03.03.2016
Place of issue:	CA HCM
Owner:	Nguyen Van Thanh
Owner social number:	444555666

Fine ticket page: view all fine tickets and their statuses.



Ticket number	Reference number	Status	Issued date	Due date	Payee	Police
🟦 1555391454977	1555391454977	Đang chờ thanh toán	16-04-2019	16-05-2019	Nguyen Van Dau	Lam Le
1554615810949	1554615810949	Đã hoàn thành	07-04-2019	07-05-2019	Nguyen Van Dau	Lam Le
1553677939647	1553677939647	Đã hoàn thành	27-03-2019	26-04-2019	Nguyen Phuoc Sang	Lam Le
1553478545208	1553478545208	Đã hoàn thành	25-03-2019	24-04-2019	Nguyen Van Dau	Lam Le
1553478406629	1553478406629	Đã hoàn thành	25-03-2019	24-04-2019	Nguyen Van Dau	Lam Le
1553400687802	1553400687802	Đã hoàn thành	24-03-2019	23-04-2019	Nguyen Phuoc Sang	Lam Le
🟦 1553360192212	1553360192212	Đang chờ thanh toán	23-03-2019	22-04-2019	Nguyen Van Thanh	Lam Le
1553359170132	1553359170132	Đã hoàn thành	23-03-2019	22-04-2019	Nguyen Van Dau	Lam Le
🟩 1553359091223	1553359091223	Đã thanh toán	23-03-2019	22-04-2019	Nguyen Van Dau	Lam Le
🟩 1553358969859	1553358969859	Đã thanh toán	23-03-2019	22-04-2019	Nguyen Van	Lam Le

Fine ticket detail page



General

Ticket number:	1555391454977
Reference number:	1555391454977
Status :	Đang chờ thanh toán
Fine :	150000 VND
Reference number:	16.04.2019
Due date:	16.05.2019
Location:	Ton duc thang, Ha noi

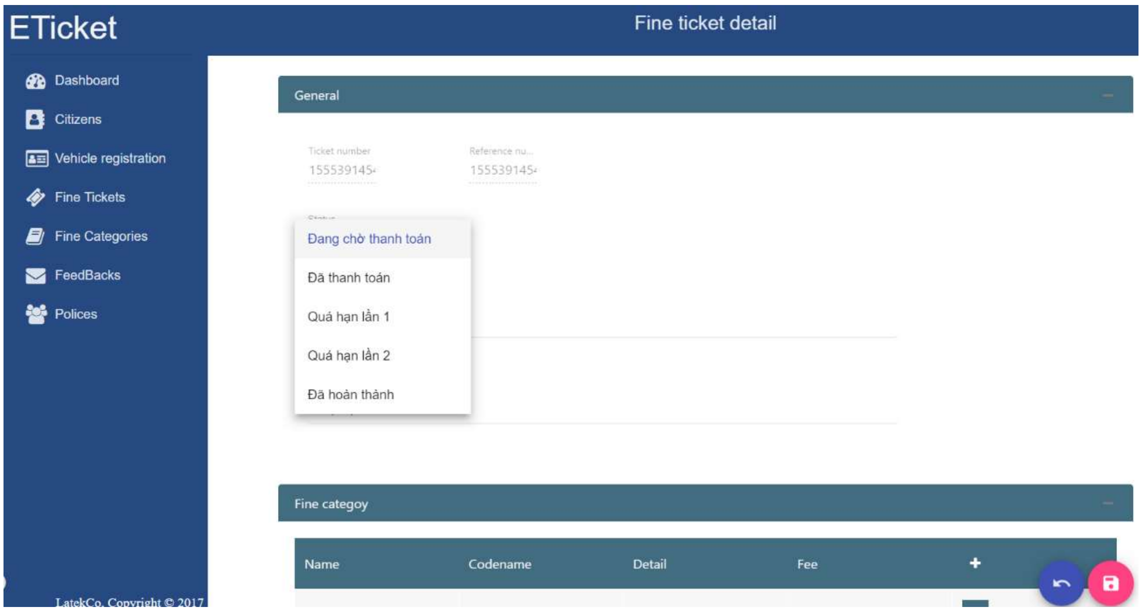
Payee

Full name:	Nguyen Van Dau
SocialNumber:	123456789
Phone:	04357365365
Birthday:	27.02.2080

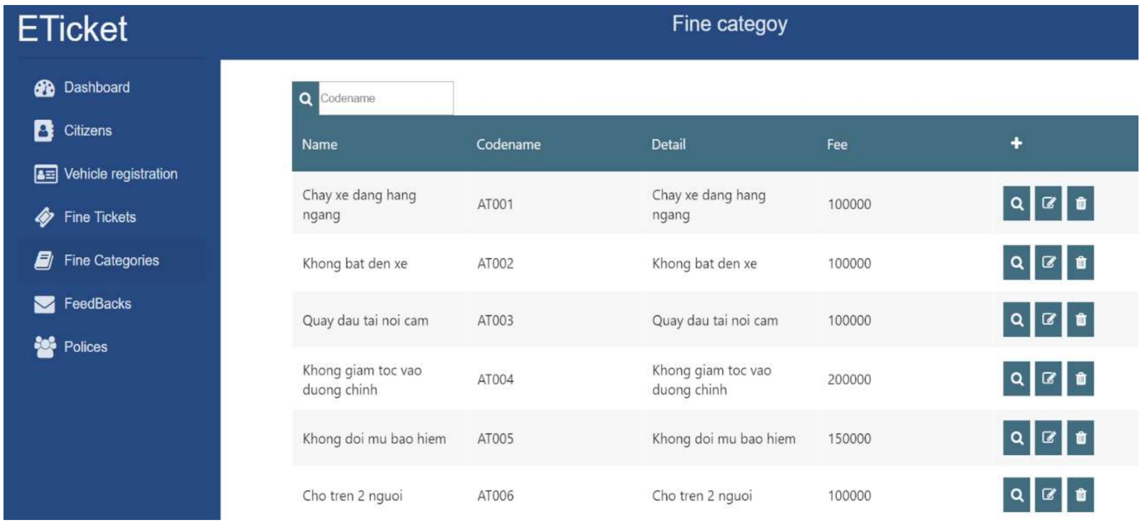
Police

Full name:	Lam Le
SocialNumber:	111222333

Update fine ticket status after payment has been received.

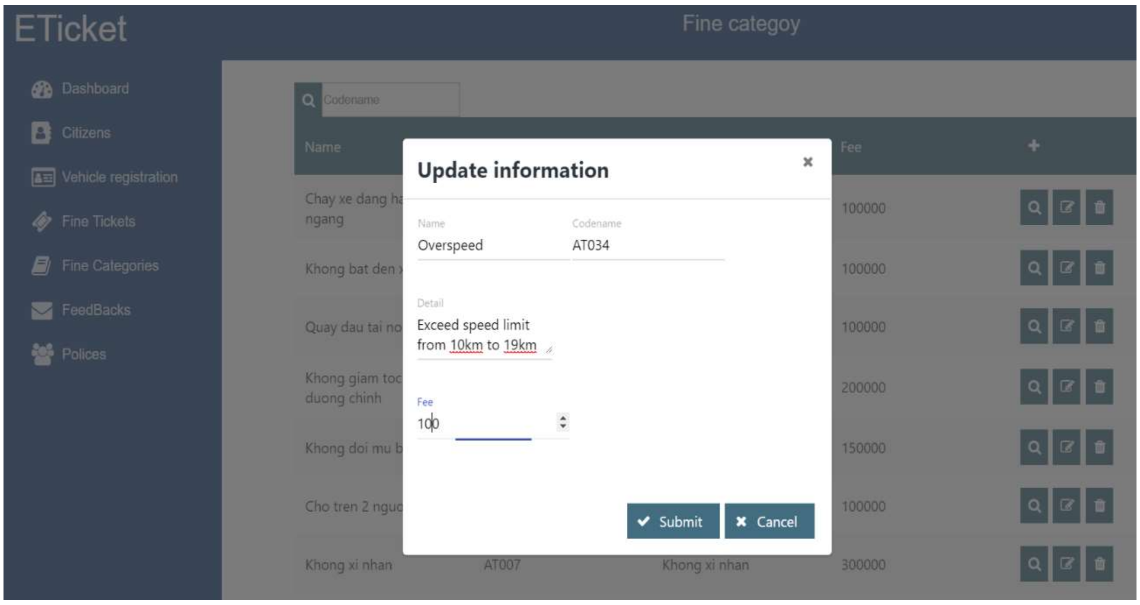


Manage fine categories: add/edit/delete fine category, including fine type, name, fine amount, and lawful information about the fine.



Name	Codename	Detail	Fee	+
Chay xe dang hang ngang	AT001	Chay xe dang hang ngang	100000	
Khong bat den xe	AT002	Khong bat den xe	100000	
Quay dau tai noi cam	AT003	Quay dau tai noi cam	100000	
Khong giam toc vao duong chinh	AT004	Khong giam toc vao duong chinh	200000	
Khong doi mu bao hiem	AT005	Khong doi mu bao hiem	150000	
Cho tren 2 nguoi	AT006	Cho tren 2 nguoi	100000	

Add new fine category

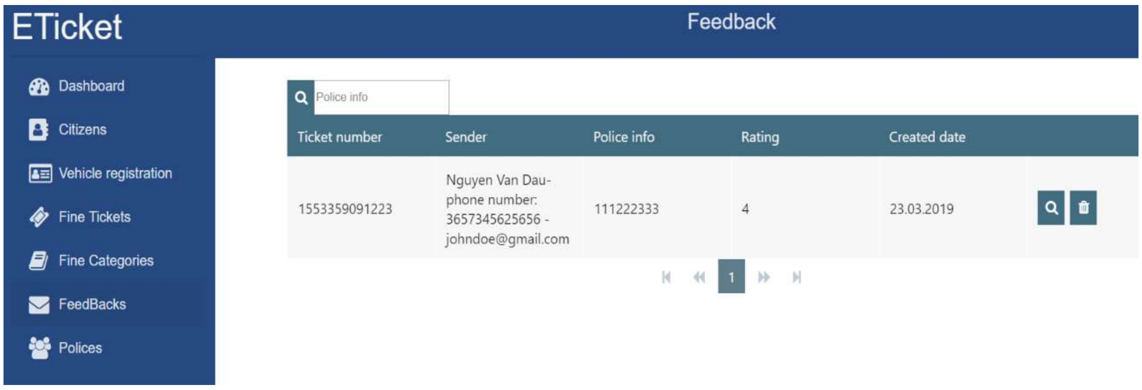


Name			Fee
Chay xe dang ha ngang			100000
Khong bat den x			100000
Quay dau tai no			100000
Khong giam toc duong chinh			200000
Khong doi mu b			150000
Cho tren 2 nguo			100000
Khong xi nhan	AT007	Khong xi nhan	300000

Update information

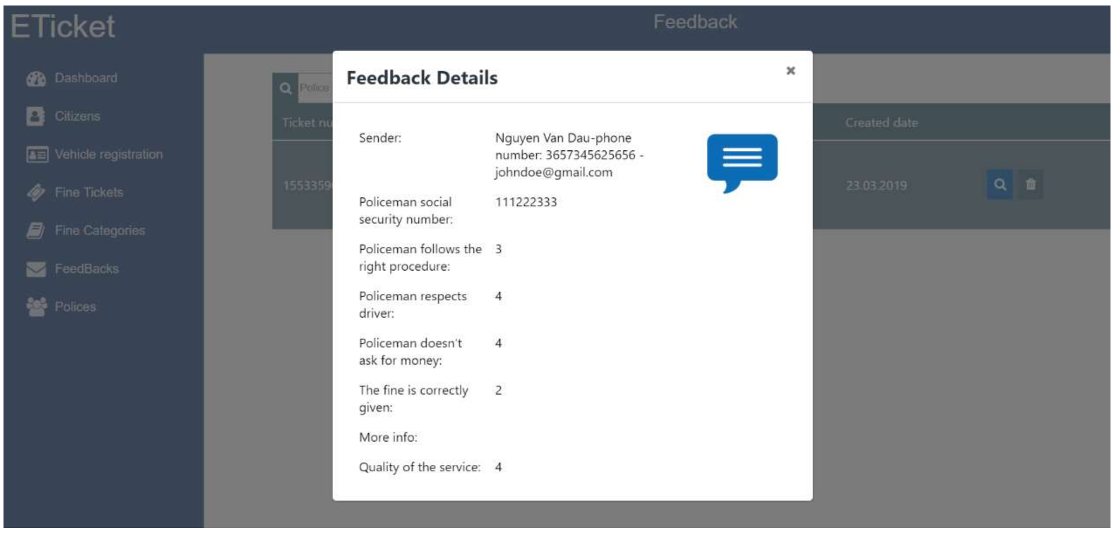
Name: Overspeed
Codename: AT034
Detail: Exceed speed limit from 10km to 19km
Fee: 100

Police feedback



Ticket number	Sender	Police info	Rating	Created date
1553359091223	Nguyen Van Dau- phone number: 3657345625656 - johndoe@gmail.com	111222333	4	23.03.2019

View feedback detail

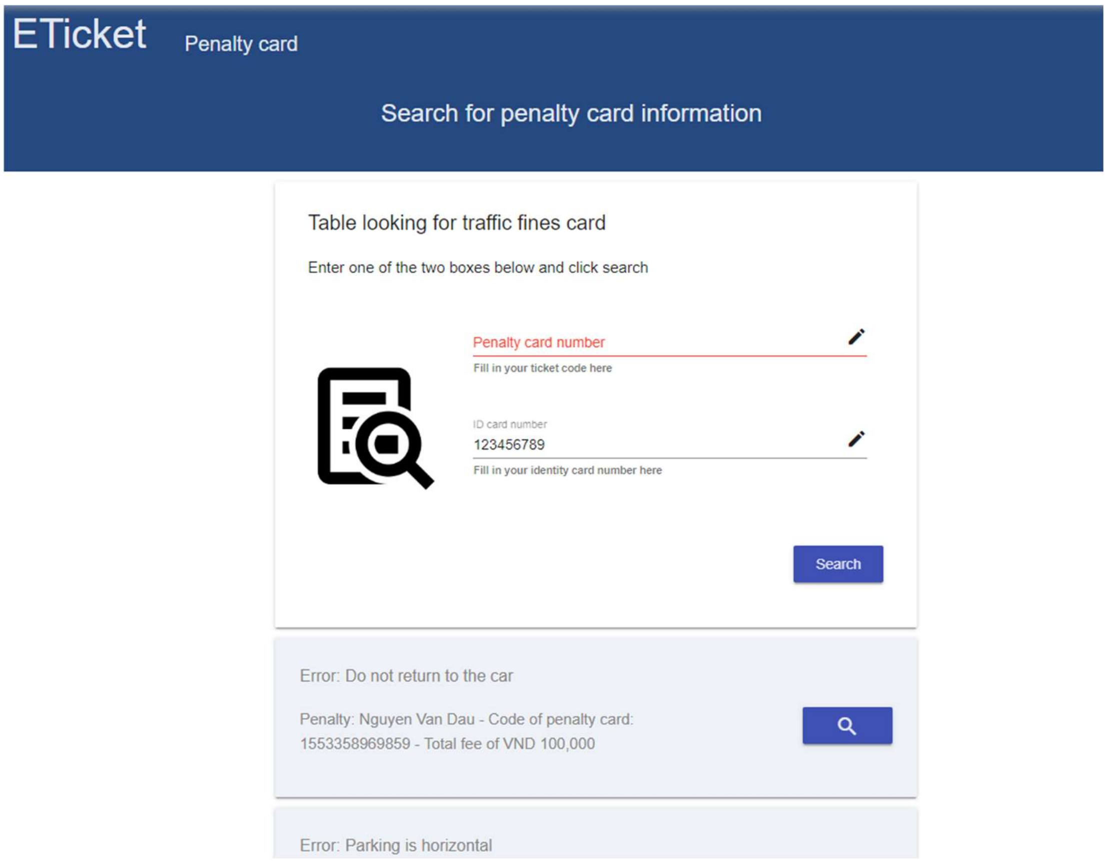


Feedback Details

Sender:	Nguyen Van Dau-phone number: 3657345625656 - johndoe@gmail.com
Policeman social security number:	111222333
Policeman follows the right procedure:	3
Policeman respects driver:	4
Policeman doesn't ask for money:	4
The fine is correctly given:	2
More info:	
Quality of the service:	4

Appendix 2: User interface of E-ticket Client.

Search for fine tickets based on ticket number or citizen’s social security number.



ETicket Penalty card

Search for penalty card information

Table looking for traffic fines card

Enter one of the two boxes below and click search

Penalty card number

Fill in your ticket code here

ID card number

123456789

Fill in your identity card number here

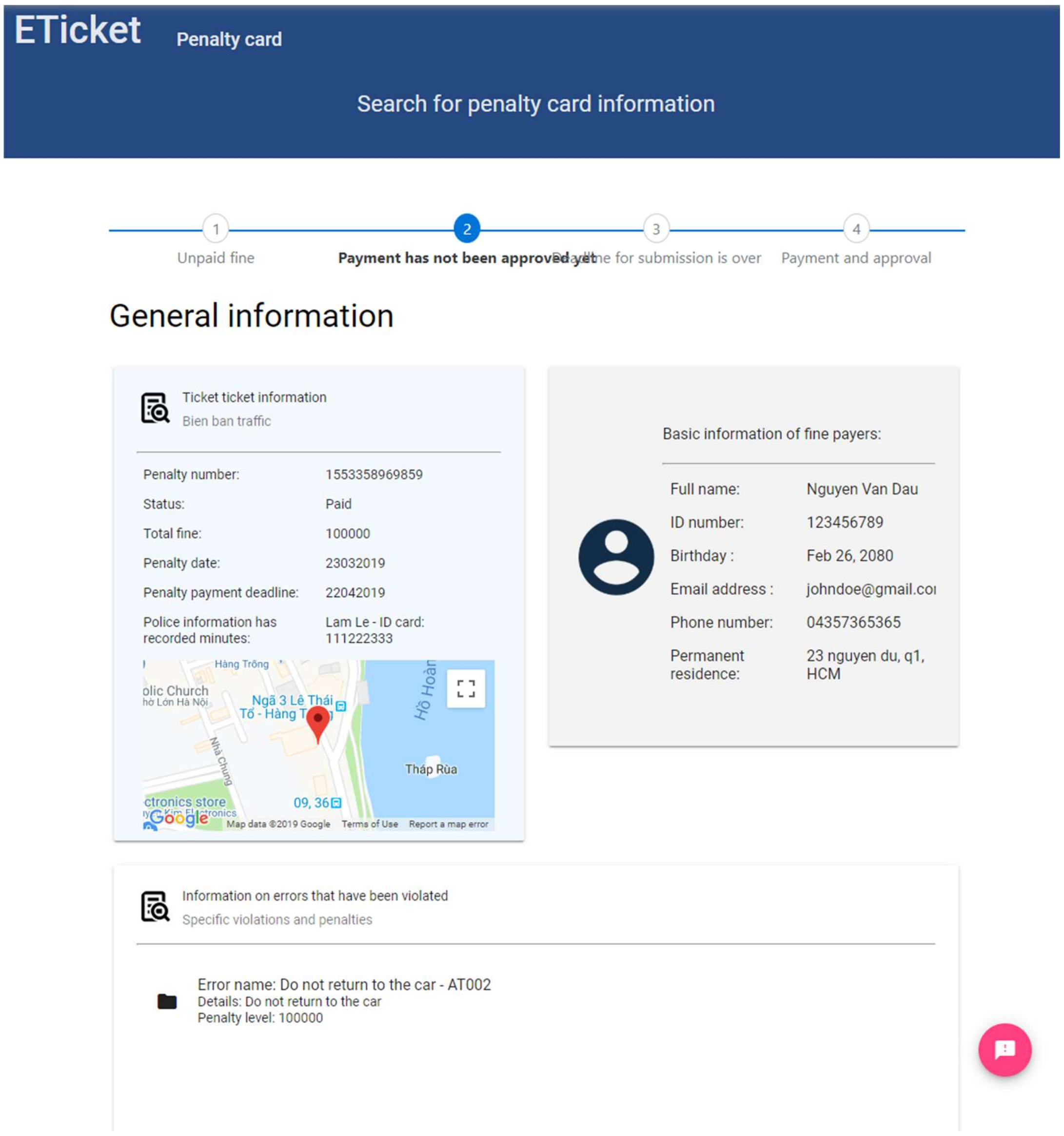
Search

Error: Do not return to the car

Penalty: Nguyen Van Dau - Code of penalty card: 1553358969859 - Total fee of VND 100,000

Error: Parking is horizontal

Fine ticket detail



ETicket Penalty card

Search for penalty card information

1 Unpaid fine — 2 **Payment has not been approved yet** — 3 Deadline for submission is over — 4 Payment and approval

General information

Ticket ticket information
Bien ban traffic

Penalty number:	1553358969859
Status:	Paid
Total fine:	100000
Penalty date:	23032019
Penalty payment deadline:	22042019
Police information has recorded minutes:	Lam Le - ID card: 111222333

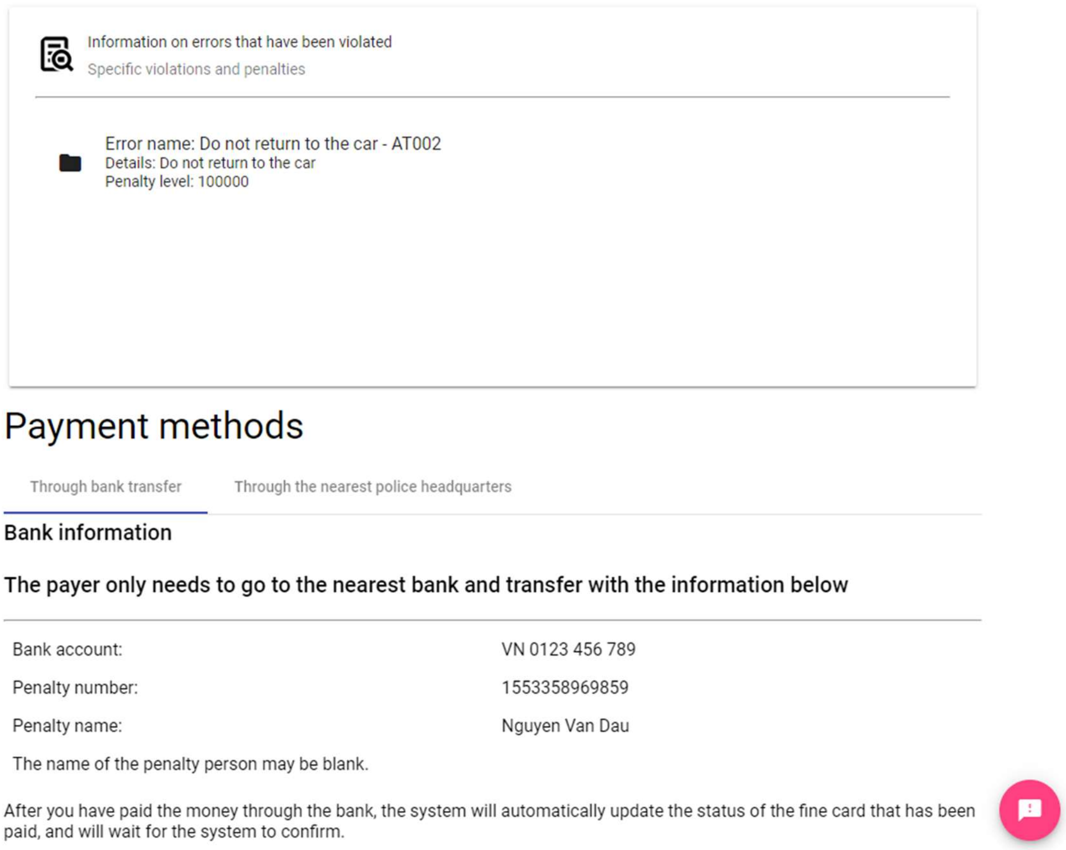
Basic information of fine payers:

Full name:	Nguyen Van Dau
ID number:	123456789
Birthday :	Feb 26, 2080
Email address :	johndoe@gmail.co
Phone number:	04357365365
Permanent residence:	23 nguyen du, q1, HCM

Information on errors that have been violated
Specific violations and penalties

Error name: Do not return to the car - AT002
Details: Do not return to the car
Penalty level: 100000

Payment method


Information on errors that have been violated

Specific violations and penalties

Error name: Do not return to the car - AT002
Details: Do not return to the car
Penalty level: 100000

Payment methods

Bank information

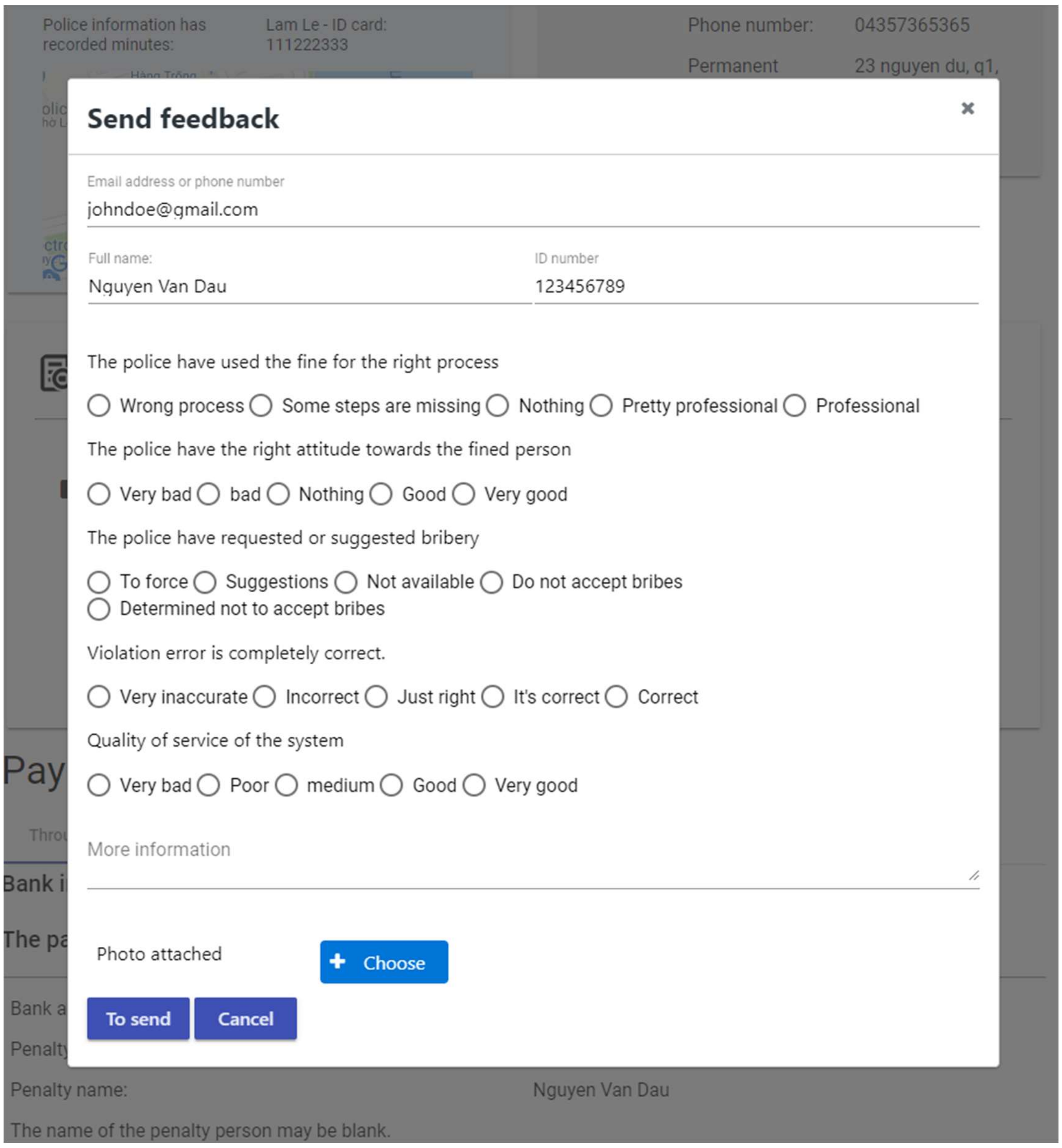
The payer only needs to go to the nearest bank and transfer with the information below

Bank account:	VN 0123 456 789
Penalty number:	1553358969859
Penalty name:	Nguyen Van Dau

The name of the penalty person may be blank.

After you have paid the money through the bank, the system will automatically update the status of the fine card that has been paid, and will wait for the system to confirm.

Send feedback



Police information has recorded minutes: Lam Le - ID card: 111222333

Phone number: 04357365365

Permanent 23 nguyen du, q1,

Send feedback

Email address or phone number: johndoe@gmail.com

Full name: Nguyen Van Dau

ID number: 123456789

The police have used the fine for the right process

○ Wrong process ○ Some steps are missing ○ Nothing ○ Pretty professional ○ Professional

The police have the right attitude towards the fined person

○ Very bad ○ bad ○ Nothing ○ Good ○ Very good

The police have requested or suggested bribery

○ To force ○ Suggestions ○ Not available ○ Do not accept bribes
○ Determined not to accept bribes

Violation error is completely correct.

○ Very inaccurate ○ Incorrect ○ Just right ○ It's correct ○ Correct

Quality of service of the system

○ Very bad ○ Poor ○ medium ○ Good ○ Very good

More information

Photo attached

Penalty name: Nguyen Van Dau

The name of the penalty person may be blank.