ECAI 2017 - International Conference – 9th Edition
Electronics, Computers and Artificial Intelligence
29 June -01 July, 2017, Targoviste, ROMÂNIA

Traffic management system

Alexandru Nițu
University of Pitesti
Pitești, România
alexandru.nitzu@gmail.com

Cosmin Știrbu
University of Pitesti
Pitești, România
cosmin.stirbu@upit.ro

Alexandru Ene
University of Pitesti
Pitești, România
alexandru.ene@upit.ro

Florentina Magda Enescu
University of Pitesti
Pitești, România
enescu_flor@yahoo.com

Abstract

This paper's purpose is describing and highlighting of how our system is intended to work and how it can help us in everyday life, as well as making the Earth a safer and cleaner place, by not even noticing it's presence.

Our system is always monitoring the traffic of every major city, analyzing it and making decisions on how people should drive, in a manner that the roads are safer, the fuel consumption is reduced, as well as the pollution. The system also takes in consideration the traffic jams and gives hints on how we can avoid the jammed traffic so we can be on time to our destination.

On the other hand, we have to consider the pollution that appears when everybody drives his own car instead of using the public transportation systems. Our system will receive information from strategic placed sensors that measure the pollution level, and after interpreting the information, takes decisions on how we can reduce the level of pollution.

Keywords: Traffic management system, SQL, Java, JSON, PHP3

I. INTRODUCTION

The system and application will be used by anyone who wants to drive more conscious, but at the same time avoid traffic jams. We all are tired of waiting in lines and being late at work because an accident happened or there is some work done on the roads. With this in mind, we created a system that solves this problem and even more, while giving the consumer a user friendly way to interact with this data, by creating a mobile application, more specific, an android application.

Our system consists of three main components: the sensors, which will be installed on the streets and connected to the internet so they can send data to the second component, which is the server, and a mobile application. The server receives the information from the sensors, analyze it and store it in a database. The processed information is sent to the mobile devices so the users can take advantage of it. Evidently, the client-server model is used in this system, both by the sensors transmitting data to the server, and the mobile application receiving the information to the server, but also transmitting information to the server.

On the server side, there is a web service that waits to receive requests, then verifies the request and takes action according to the request type. The web service in implemented using PHP, Java Script and HTML. For the database we use My SQL. The android application was created in Java. We also used other technologies, as Google Cloud Messaging for sending notifications on the mobile devices, Google Maps API, JSON for transforming the information sent from the server, to usable information on the mobile client.

II. SENSORS

A. Comunication

For the connection between the Sensors and the server, we are using TCP / IP protocols and HTTP requests. Using the HTTP requests we can send information to the server so it can be stored in the database and after that processed and sent to the user. We are using this kind of communication, because we are making the system reconfigurable for anyone and anyone can add sensors that send data to the server by using the API we provide.

B. Security

One of the most important aspects in the online world we live in, is the security of the data we are putting online and transmitting over the internet.

To cover this issue we are using an encryption algorithm that uses 2048 both public and private keys to ensure that the date cannot be decrypted even if it is intercepted by anyone. More details about this encryption method can be found in the server section below.

To ensure the security even more, we assured that the sensors can only send data to the server, not the other way around, this way minimizing the possibility

978-1-5090-6458-8/17/$31.00 ©2017 IEEE

of too much data being requested from the server and possibility of system overloading. The sensors send data to the server via an encrypted string, the server receives it, decrypts and analyzes it. If the information doesn't respect a specified pattern and doesn't fin in our specified boundaries, the information is dismissed and not stored in the database.

C. Sensor architecture

The sensor type doesn't matter. It can be any kind of sensor and controller connected to the internet that can send a HTTP request. This way the sensor can be just a carbon dioxide sensor which measures the pollution around or a station that consists of a video camera which sends traffic information, a pollution sensors and pressure sensors in the road which count the number of cars passing by.

III. THE SERVER

As mentioned above, the server part of this system is created as a web service, using PHP, and a MySQL database. We chose to create it as a web service mainly for security purposes, and the fact that we can create an API that everybody can use in their own applications.

The server receives requests both from the sensors and the mobile clients. The request could be for the server to save data in the database or to send a specified type of information to the client or to the sensors if requested.

A. Client – Server communication

The Client – Server communication is assured using the TCP/IP protocol and HTTP requests. The system architecture can be seen in the Figure 1. The request comes from the client or the sensors in an encrypted format (string) which is encrypted by the client and then decrypted by the server. The request is then analyzed by the server and if the format is good, the server takes action to save the specified information in the database if requested by the sensors / clients.

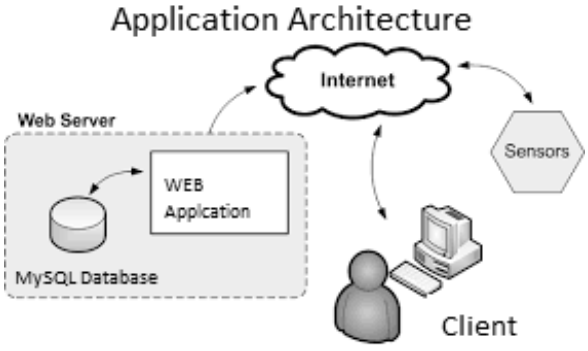

Figure 1. Application Architecture

In case the clients are making requests for the server to send them information, after the string is decrypted and analyzed by the server, this one is searching for the required information in the database or calculating information. The server then takes the information from the database and uses it to calculate and send the information back to the client.

The server transforms all the information that is about to be sent back, in a JSON format, this way, the information is received by the client in an object or string of objects form the information is simply written using HTML on a blank page that the client can request to receive and read character by character and then used in the mobile application as desired.

B. The database

As mentioned above, the database type we are using in this system is called My SQL. In the following image (Fig. 2) can be seen four tables from our database: Users, User Location, Sensors and SensorData.

In the Users table is stored all the information we need about the client user. His email, password, name, date that he created his account and date he last used the application, as well as the mobile device's ID, used by Google Cloud Messaging to send push notifications and alerts. The User Location table is used to store every user's current password.

In the Sensors tabled is stored the information about every sensor which is connected to the server and sends information to it. We have an ID which is used to identify every sensor, as well as a title and details field to better identify the sensors. The type field helps us know what the sensor is used for (pollution sensor, traffic sensor, etc.) In the Sensor data table, we sore every value the sensor sends as well as the date each data was recorded.

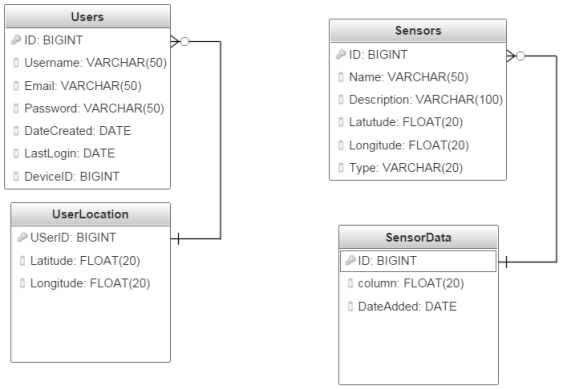

Figure 2. Database tables

C. Security

The messages sent from the server, as well as from the client application or the sensors, are encrypted, this way ensuring the security of our system even if the messages are intercepted on the way from each external component to the server.

As encryption algorithm we chose to use a practical public-key cryptosystem which is used in a wide variety of applications, called RSA [1]. One of the advantages of an encryption system like this is that the encryption key (which is public) is different from the decryption key which is kept secret

In our system, there are two ways in which the encryption-decryption system can work. The first one states that both the server and the clients have stored in their code, the public and the private keys. This way

we can use 2048 bit keys which decrease the possibility that the keys can be broken in any way. The second method consists in generating both the public and private keys for each client individually, as described in Fig 3. [2] The first step is that the server generates his keys, public and private, and sends the public key to the client. The client then generates his keys, encrypts them using the server's public key, and sends them to the server. The next step for the server is decrypting the client's keys using his own private key. From now on the communication can be assured safely and encrypted using the client's public key.

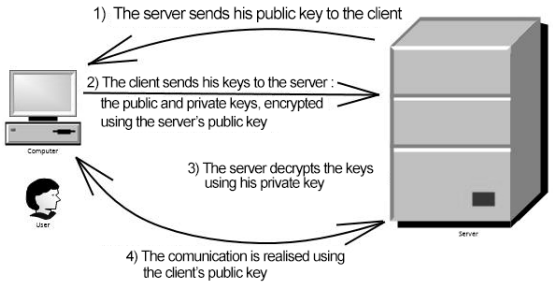

Figure 3. Encryption system diagram

D. Sending notifications and alerts

Sometimes, we need to send the users alerts and notifications on their mobile devices, either about the traffic ahead, the pollution level ahead or alerts about any other modifications. We can do that using the GCM (Google Cloud Messaging) platform, for its ease of use, security and multi device notification sending features [3]. This platform works even if there is no connection to the server, meaning that we don't stress the server too much by sending notifications to every user and occupying to many resources. This way, the notifications can be sent either from a separate web application, an android application or automatically from our web service if the server considers that it should.

For Google Cloud Messaging to send information, we must register on the same platform, any new client device that is going to be used. Google then gives each device a unique ID which we store in our database as "DeviceID". We can then send notifications to a specific user, a group of users or all the users of the application. In the image below (Fig 4) we can see the way GCM works [4].

The mobile application sends a request to the Google cloud messaging platform, the GCM server responds by sending back an ID based on the phone's IMEI address. Now our web service comes in help, and after the mobile client sends the new generated ID, it saves it in the database. When the server has to send an alert or notification, it calls a function from the GCM API, sending the ID's of the devices the notification will be sent to. The Google Cloud messaging servers send the notification to all the specified ID's using the resources of their own servers which helps us a lot. After all the notifications are sent, the GCM Server responds to our web service with information regarding how many devices received the notification, and if some of them don't receive it, this will be shown in the response.

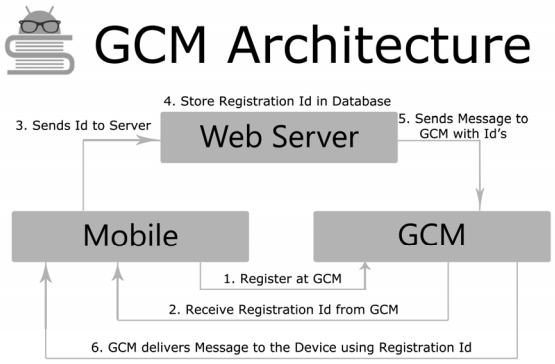

Figure 4. Google Cloud Messaging Platform Architecture

IV. THE MOBILE APPLICATION

The mobile application is written in Java and the design is created using XML.

From the user interface the user can create a new account or sign in using his previous created one. For the application to be more user friendly, we also added the possibility of the user to login / register using his / hers Google account. We use the user account feature so that all the user information can be stored in the database, and also register the user's location at a specified time.

For the design part, we used a design library called Material Design, An universal design language for any user interface on any type of device [5].

After logging in, the user has the possibility to see the map, his previous routes, the most traffic intense areas of a specified location, or the most polluted areas. You can also calculate the fuel consumption by selecting two points between which to navigate. You can also input manually the number of liters left in your fuel tank and select the medium consumption, and the application calculates at which fuel station you should fill up the tank at.

All this information about the maps and navigation is calculated and shown using the Google maps API, [6] by using public known functions from google maps. The distances are calculated directly from the maps, so they are accurate, but the altitude is also taken in consideration both when calculating distances and fuel consumption.

For the notification and alerts part, we use the client part of the Google Cloud Messaging platform which stores on the device the given ID. This ID is also connected to the phone's IMEI Address, which makes the ID unique across every mobile operating system. For the notification part to work, we have to implement a service that always runs in the background, but also not use too much battery life and not interfere with another application's services. This service must also start when the phone boots up, so whatever happens to your phone, if it's on and online you can receive the notifications. We also implemented another service for the alerts, so if there are news of public interest of in case of natural

disasters, we can send alerts to every phone so people get to safe places.

There is an option for everyone who uses the app to search for specific location, for example all the café shops in one specific area, all the places to visit in one town.

Directly on top of the map, while the navigation is opened, you will see three buttons which can be used to alert other people of a different situation on the road. For example, if you are driving and you pass by an accident, you can simply tap the accident button, and everyone in that specific area can see that at your location there is an accident. You can do the same thing with a traffic jam, if you find yourself in one and nobody alerts it, you can help people around you avoid it and find a better route. The third button is used to signalize if there is some work done to a portion of the road. Any of this alerts have a time limit depending on their type, but if someone passes by and sees that the alert is no more valid, can cancel it by two clicks. We minimized the number of clicks, because we know driving is important and always looking at the road is important.

V. SIMILAR IMPLEMENTATIONS

At this moment, there is a mobile application in Europe which is used by a lot of users, system which lets users log on the map information that could not else ware be known, like where the road is damaged, or location where there is work done to the road, even where there are police radars. Our system's advantage is that using sensors, we rely on more sources of information which helps us in better determining traffic information and detecting traffic jams. Our system is also an upgrade to the existent system, called Waze, by the fact that our system is also pollution friendly, by redirecting traffic to less polluted areas, or areas containing more vegetation.

VI. CONCLUSIONS

The system will help every user in driving smartly, by providing useful information about the traffic in the corresponding area the user wants to navigate through. The system needs sensors to work accordingly to the specifications, sensors which are not installed yet in neither of the cities in Romania. The good part is that it also relies on users to register data to the application, which can be useful in the beginning, before the sensors are installed.

REFERENCES

[1] RSA Cryptosystem https://en.wikipedia.org/wiki/RSA_(cryptosystem)

[2] S. Burnett, and S. Paine, "RSA Security's official guide to cryptography" Hill Companies, 1st edition, p12, 2001.

[3] Google Cloud Messaging https://developers.google.com/cloud-messaging/

[4] J. U. Gonzales, and S. P. T. Krishnan, "Building your next big thing with Google Cloud Platform" Apress, 1st edition, p27, 2015.

[5] Material design specs , colors and layout https://material.google.com/

[6] Add google maps to android application https://developer.android.com/training/maps/index.html