Mobile Application Architecture, Implementation for a Local Gas Prices with Community Participation, and Gas Station Location

Alex Santos Ramos
Master of Engineering
in Computer Engineering
Advisor: Dr. Nelliud Torres, DBA
Electrical and Computer
Engineering and Computer
Science Department
Polytechnic University
of Puerto Rico

Abstract - The objective of this article is to establish the fundamental components of a mobile architecture model that enables the development of a production-graded solution for local (Puerto Rico) gas prices and gas station locations. It will establish this model by identifying each component based on the responsibilities of the system. Components will be identified by looking at the basic needs and desired features of the proposed system, from a high-level point of view. The approach is to match the system expected behavior with technologies that enable those functionalities. Once the components are identified, it will describe the interaction between them and take the basic considerations regarding mobile infrastructure solutions like communication technologies and security system. It will also provide a prototype mechanism to establish a custom/proprietary advertisement system through the mobile application in which advertisement space will be allocated at the mobile graphic interface. Finally, it will consider how the technology developed here can be expanded into other areas/solutions.

INTRODUCTION

This article is part of an initiative from the engineering services company *Empresas O'Neill, LLC* to enter the mobile development and application business [1]. This company is currently looking into a local advertisement business model using mobile application. One of the areas of services been considerate by the company is the facilitation of local gas prices with user participation, and near gas station location with minimal list sorting criteria as a value added for the potential mobile app users. As an example, users can decide to save money on gasoline by knowing beforehand what are the price options that they have close them. As today (Sept 2018), Puerto Rico's gas prices can variate from 3 to 8 cents per liter [2].

More importantly, the article focuses in encapsulating all the concepts, technology and further information that enables a mobile application architecture, that serve as a building block or/and starting point to develop more complex mobile solutions into the future.

PROBLEM STATEMENT

As per every application development, this article starts by gathering all the needed capabilities and expected behaviors of the mobile application. This is done by establishing a list of use cases and desired features.

Is expected for that application to:

- Create a basic model for a mobile application infrastructure.
- Be a mobile application that can be used from a cellphone.
- To obtain near gas stations with single touch.
- Get/Submit gas prices for a gas stations.
- Sorting gas station per criteria.
- Use Google Map integration for GPS.
- Provide a space for advertisement.
- User identification.

Most of the features described above translate to low-level requirements, but they also allow us to start identifying what are the mayor needs for the system. We can archive this by grouping the above features/capabilities within mayor components that can provide each service. This methodology helps in identifying components and requirements in a general high-level definition for the overall system.

All the previously describe capabilities can be assigned to the following components:

- **Mobile Application** - Be a mobile application that can be used from a cellphone, provide a space for advertisement, sorting gas station per criteria, obtain near gas stations with single touch.
- **Web Server** - Get/Submit gas prices for a gas stations.
- **Third Party Services** - Use Google Map integration for GPS, User identification.

With this exercise, the basic model of a mobile infrastructure starts to take shape.

DESIGN

Translating desired/needed features to system components creates a starting point for the design. The design is developed by expanding and completing the existing identified components by taking into consideration all technical aspects that are required to build the mobile architecture.
There are three more functionalities that from a technical aspect are required to complete the mobile infrastructure, and those are:

- **Inter Component Communication** - individual components needs to have a common way to communicate with each other.
- **Data Storage** – need to store all prices submitted by the users and provide them back to the users.
- **Security** – the infrastructure needs to provide the basic principle of security: confidentially, integrity and availability (CIA) [3].

From now on all the components are identified from the ground up starting from what will host each component of the mobile infrastructure, up to which technologies are used to implement them.

To run a **web server**, we need to select an Infrastructure-as-a-Service (IaaS) provider [4] [5]. The IaaS provides the computing and cloud resources to host the mobile infrastructure server side. This project will be build using the Amazon Web Services (AWS) Free Tier in particular the Amazon Elastic Compute Cloud (EC2) and the Elastic Load Balancer services [6] [7]. Over this IaaS a Web Stack can be installed to support all the **Web Server** functionalities, **Data Storage**, **Security** and part of the **Inter Component Communication**. Nowadays there are a lots of options as Web Stack, but for this project a Linux-Apache-MySQL-PHP (LAMP) Web Stack will be implemented as it covers all the necessities of the mobile architecture in a relative simple way [8] [9]. The AWS EC2 instance will provide the cloud/web computer to host a Linux OS. Linux will be hosting the server Apache to receive the HTTP requests to the server [10]. Other two components that the OS will be hosting are the database system MySQL and the scripting language preprocessor PHP. MySQL will contain the mechanism to storage data in the OS file system and the language to request and submit data on it [11]. PHP is a hypertext processor that allows us to define and handle the server request/response and the local interactions with the database [12]. See below illustration for an overall setup for the web server configuration.

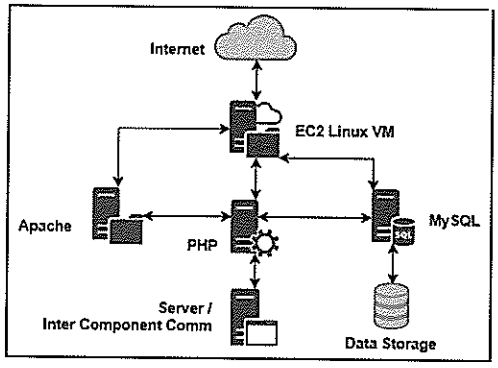

Figure 1: Web Server

The remainder of the responsibility for a mobile architecture falls under the **Mobile Client**. The **Mobile Client** will provide the users the mobility meaning that the service available from anywhere where there is a mobile carrier data signal. For this project it will be used the well-established Android Operative System (OS) environment. The user will be able to choose a cell phone manufacture of his choice. The manufacture will provide its Linux Kernel which is from where the Android framework operates from and the communication between their hardware and the Android OS called Hardware Abstraction Layer (HAL). The OS will provide a complete environment from where to run an Android application [13]. This means that the implementation of the mobile client will only need to focus on the Android mobile application.

Finally, the inter component communication will be implemented using a **RESTful API**. The RESTful API make use of the HTTP technology to establish communication between web and mobile components [14].

IMPLEMENTATION–WEB SERVER

The **web server** is divided and implemented into the following groups:

- database management
- security and access
- third party services
- support

The database management control the access to the database management system and the stored data for internal (mobile app authentication) and external (app data) requests. It is executed by three PHP classes which are:

- db_config.php
- db_class.php
- db_manage.php

The db_config holds the information needed to connect to the **database management** system. The db_class defines the object with which the web server connects to the database system in order to provide and request

information to the database in a generic aspect. These two file codes contribute into creating the technology for the mobile architecture. Last db_manage is the application specific implementation for the database which defines the queries that will be performed by both the web server and the mobile client application.

Creating a service that is accessible through the internet brings the challenge of how access to that service will be managed. The **security and access** from the web server point of view are related to getting access to the web server services, and which http protocols are used to communicate securely to the server. The following code files manage these aspects:

- fuletealo_api_authorizer.php
- http_post_only.php

The fuletealo_api_authorizer will ensure that the mobile application developed for this mobile architecture is the only with access to the web server services. It does that be validating a unique information provided by the mobile application against what is store on the database for the same mobile application.

Of all the http methods available the http POST is the only that don't expose the user/client submitted data on the request url. Therefore, the http_post_only code file will ensure that all requests made to the web server are POST methods or they will be rejected. These two code files will also provide towards creating the mobile architecture technology.

Between the application specific functionalities there is the capability gas stations near to the user. To accomplish this, the system make use of a **third party service** from Google that enables this functionality at a server level [15] called Places API. To manage the Places API endpoints and credentials the following code file was created:

- =google_api_config

Another third party service needed for the system is the definition of valid gas prices. This is taken for the local Puerto Rico consumer department DACO [2] . The following code file manage to grab the price ranges from DACO:

- =Get_daco_gas_price_ranges.php

Finally, there is a **support** function to calculate distance between the coordinates of the user location and the gas stations near him.

IMPLEMENTATION - RESTAPI

The REST API is used in other to enable a standardize way from which two different components can communicate with each other. This communication is done with the use of the http verb POST. The two components that will make used of it in the mobile architecture are the Wed Server and the Mobile Application. The mobile application will always be the requester and the web server will be the responder.

At the web server side, the RESTAPI make available the following endpoints (urls):

- (domain)/get_gas_stations.php
- (domain)/set_gas_prices.php
- (domain)/get_price_ranges.php
- (domain)/get_ads.php

The main two functionalities of the mobile application are getting near gas stations and providing prices for them. The get_gas_stations.php allows the mobile application to ask the server to request all near gas stations from the Google services Places API. The set_gas_prices allows the mobile application to ask the web server to store gas prices for a gas station near the user that was presented as a part of gas station list. To support the ability to provide gas prices by users, the mobile application can get a valid range for gas prices from the web server by using the endpoint get_price_ranges with which the application can determine if the prices provided by a user are valid or not. To support the business models of providing ads the REST API have the endpoint get_ads which return all the ads entered by the database administrator in the database. This endpoint also provides a possible component for future different mobile architecture.

To complete the RESTAPI all the http calls were defined within and Android library [16] . Within this library various java classes where defined to enable communication through the RESTAPI.

The classes can be group under the following categories:

- HTTP Connector: RESTUtil.java
- RESTAPI client: Server.java
- Data modeling: UserData.java, GasPrices.java
- Support: AppFingerprint.java, Hash.java

The *RESTUtil* class provides the bare bone of the htttp protocol for the mobile application which is used by the server class to connect/use the endpoints provided by the web server. The server class also make use of the *AppFingerprint* and Hash classes to provide the web server a unique information with which the server can uniquely identify the mobile application. Finally, the data

modeling classes *UserData* and *GasPrices* are used for easiness to data allocation and transmission. This Android library represent a building block for future mobile architecture as well.

IMPLEMENTATION - MOBILE APPLICATION

The mobile application can be divided into two major areas: its code and the graphical user interface (GUI).

The application code can be divided as well into its functional blocks as follow:

- View Controllers: *SplashScreen*, MainActivity, *DetailsPopUp*, *SortPopUp*.
- Managers: *AdsManager*, *RegionValidator*, *GoogleSignInManager*.
- Data Modeling: *AdItem*, *Region*.
- Support: *ListAdapter*, *MyComparator*, *Locator*.
- (It also makes use of the Android library mentioned before).

The responsibility of the view controllers is to put together the user GUI and handle all the user's request through the GUI interactions. The *AdsManager* class is in charge of requesting the ads from the web server and presenting them switching from one to another in a timed cycle. This class can be used as well as a mobile architecture building block where and ads system is in place. Since the mobile application is developed for a local (Puerto Rico) audience, the class *RegionValidator* evaluates the current user coordinates to enable the application, if the user is outside an authorized region, the class make the application close right away. The *GoogleSignInManager* wrap the configuration for using a sign in system with the Android account used for the Play Store application [17] . The data modeling classes put together all data information into an object for easiness of use and manage. To control the components of the list view for the gas stations the *ListAdapter* was created. The list can sort the gas station list by using the class *MyComparator* which allows the list to be sorted by gas station brand, distance and prices. To obtain the user coordinates the application uses the class locator which wrap all the steps needed to request coordinates from the location provider Android system class [18] .

This second part of the mobile application implementation is allocated to illustrate the GUI. The application starts with the presentation view which shows the application (Fule73alo) and the company names.

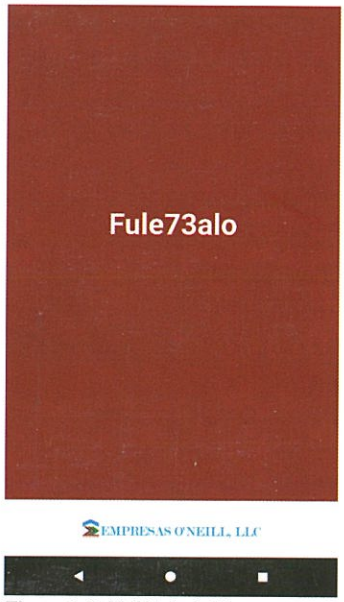

Figure 2. Initial Splash View

After the user passes the location validation where his current location is evaluated against an authorized region, the user is taken to the user principal view the main activity.

The main activity offers the following GUI interactions:

1. Ads – when this area is selected the user is taken to a link related to what is been advertised in the illustration.

Figure 3. Main Activity

2. Long Row Click – when the user holds a long click in an item of the list, the application popup a window showing more detailed information of that gas station and further options.
3. GPS Icon – when selecting the GPS icon, the application does a transition to Google Maps to illustrate the location of the gas station.
4. Drag List Down – by dragging the gas station list down, the application request again the list of gas station near, and them updates the list accordingly.
5. Three Vertical Dots Icon – when selected it present the user an option to logging (identify) itself into the application. This is required to submit gas prices.
6. Sort Icon – when selected it presents a popup window to select a criterion for sorting the presented list.

To be able to submit gas prices the user needs to login using its

Google account. Once logged in, it can perform a long click over the gas station row for which it wants to provide/update gas prices. On the text input fields, the DACO's defined prices ranges are shown.

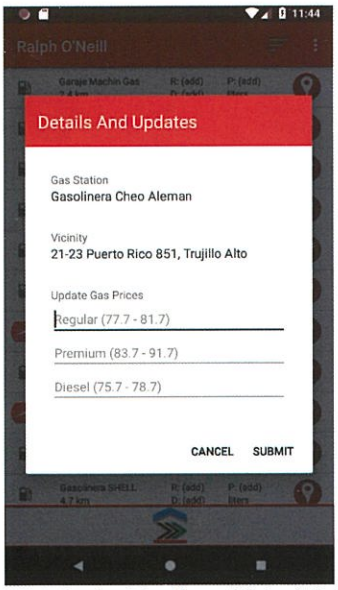

Figure 4. Station Details and Submit Prices

Once the list of gas stations near the user is obtained, the user can choose to sort the list by criteria as follows: distance, station name or prices.

Once a gas station is chosen as the place to fill up the user car gas tank, if the user is not familiar with the gas station it can then request for the application to provide direction in a GPS fashion step by step. This is enabled by the Google Maps Intents which allows a transition from our application to the Google Maps application [19] . The intent uses the coordinates of the gas station selected by the user.

SECURITY (CERTIFICATE FOR ENCRYPTION)

The final consideration will be regarding security. Since the communication is performed using a RESTAPI services, there is the need to encrypt the communication data. For this purpose, the public-key cryptography is used in particular the RSA implementation [20] . AWS services enable this functionality by using a certificate signed by them. But this signed certificate can only be assigned from the load balancer that the free tier services provides. The load balancer has a difference purpose which is outside the scope of this article, which means that is only mentioned here because of the AWS sign certificate feature. This will create an https communication protocol which is encrypted between the server and the mobile application [21] .

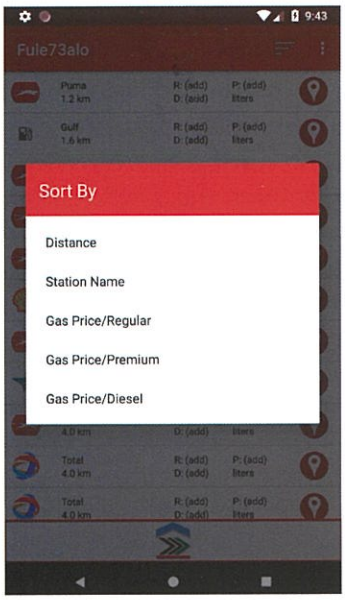

Figure 5. Sort by View

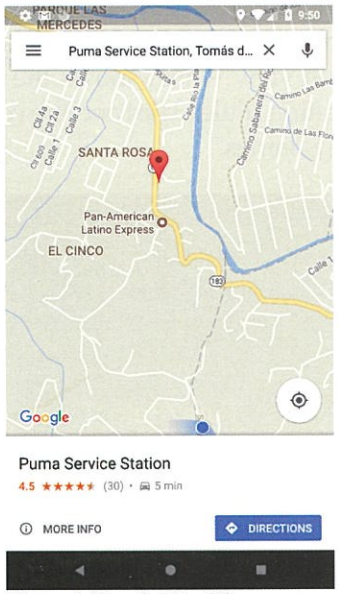

Figure 6. Google Map View

FUTURE WORK

A very important aspect of a mobile architecture that use databases is the maintenance of it which includes period backups so that the system data source can be restore completely if necessary.

Another important aspect is the ability of the system to expand/increase its computing resources. Amazon includes a system designed for this called Elastic Beanstalk which is a technology that allows the scaling of the infrastructure. Other IaaS offer similar solution [22] .

Finally, the Fule73alo application have a potential to expand its functionality into a travel or road assistance including services like finding food restaurants, hotels, car rentals and others.

CONCLUSION

From the information gathered throughout the design and implementation of the mobile architecture a generic version of a mobile architecture can be defined. Figure 7 illustrates this generic version named Web Server Infrastructure. It presents the web server setup

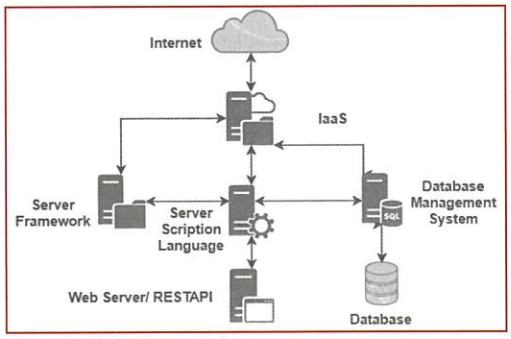

Figure 7. Web Server Infrastructure

and all technology aspects of the web server.

The RESTAPI implementation described here is an application specific development which it cannot be used directly as defined as a base code, but it will surely be a helpful reference for a future mobile architecture.

This is also the case for the mobile application which is also an application specific case, but it can be a reference for future mobile architecture. This is a mobile architecture knowledge that can be expanded by creating new solutions using this article as starting block.

References

[1] Empresas O'Neill. (2018, October 17). *Empresas O'Neill, LLC* [Online]. Available: https://www.empresasoneill.com.

[2] Departamento de Asuntos del Consumidor (DACO). (2018, October 17). *Precios de Gasolina* [Online]. Available: http://daco.pr.gov/servicios/precios_combusti bles/precios_gasolina/Pages/default.aspx.

[3] C. P. Pfleeger & S. L. Pfleeger, *Analyzing Computer Security*, Prentice Hall, 2012, ch. 1.

[4] ProfitBricks. (2018, October 17). *What is Infrastructure as a Service (IaaS)?* [Online]. Available: https://www.profit bricks.com/en-us/cloud-lexicon/iaas/.

[5] Stackyfy. (2017, October 7). *Top IaaS Providers: 42 Leading IaaS Providers to Streamline Your Operations* [Online]. Available: https://stackify.com/top-iaas-provi ders/.

[6] AWS. (2018, October 17). *AWS Free Tier* [Online]. Available: https://aws.amazon.com/free/?awsf.Free%20 Tier%20Types=categories%2312monthsfree.

[7] AWS. (2018, October 17). *Amazon EC2* [Online]. Available: https://aws.amazon.com/ec2/.

[8] StackShare io. (2018, October 17). *Tech Stacks* [Online]. Available: https://stackshare.io/stacks.

[9] M. Korsak. (2016, May 31). *What is a LAMP Stack?* [Online]. Available: https://medium.com/linode-cube/what -is-a-lamp-stack-4c36da4d2aa8.

[10] Mozilla. (2018, October 17). *An overview of HTTP* [Online]. Available: https://developer.mozilla.org/en-US/ docs/Web/HTTP/ Overview.

[11] MySQL. (2018, October 17). *What is MySQL?* [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html.

[12] PHP. (2018, October 17). *What is PHP?* [Online]. Available: http://php.net/manual/en/intro-whatis.php.

[13] Android, Developers. (2018, October 17). *Platform Architecture* [Online]. Available: http://php.net/manual/en /intro-whatis.php.

[14] REST API Tutorial. (2018, October 17). *Learn REST: A RESTful Tutorial* [Online]. Available: https://www.restapi tutorial.com/.

[15] Google. (2018, October 17). *Places API* [Online]. Available: https://developers.google.com/places/web-servi ce/intro.

[16] Android. (2018, October 17). *Create an Android library* [Online]. Available: https://developer.android.com/studio/ projects/android-library.

[17] Android. (2018, October 17). *Google Sign-In for Android* [Online]. Available: https://developers.google.com/identi ty/sign-in/android/s tart-integrating.

[18] Android. (2018, October 17). *Location Strategies* [Online]. Available: https://developer.android.com/guide/topics/loca tion/strategies.

[19] Android. (2018, October 17). *Google Maps Intents for Android* [Online]. Available: https://developers.google. com/maps/documentation/urls/android-intents.

[20] W. Stalling, *Cryptography and Network Security*, Pearson, 2012, ch. 9.

[21] AWS. (2018, October 17). *Elastic Load Balancing SSL/TTS Certificates for Classic Load Balancers* [Online]. Available: https://docs.aws.amazon.com/elasticloadbalan cing/latest/classic/ssl-server-cert.html.

[22] AWS. (2018, October 17). AWS *Elastic Beanstalk* [Online]. Available: https://aws.amazon.com/elastic beanstalk/?nc1=h_ls.