EPiC Series in Computing

Volume 58, 2019, Pages 325–335

Proceedings of 34th International Conference on Computers and Their Applications

Emergency Communications Database with Ticketing Archive for Training New Employees

Dustin T. Ginn, Ilhyun Lee, and Haesun Lee
Mathematics and Computer Science Department
The University of Texas of the Permian Basin
Odessa, Texas 79762
lee_i@utpb.edu

Abstract

The Emergency 9-1-1 Resiliency Platform (E9RP) is a web tool designed as a ticketing system and an efficiency database tool. The ticketing system will allow users to initiate, update, track, close, and view notes of an emergency 9-1-1 system. Furthermore, the ticketing system will archive ticket, and their pertinent details, for users to retrieve at a later date. Primarily, these users will be less experienced technicians that find themselves stumped by a particular problem that a customer has reported. These users will be able to access and search the historical reference to compare past comparable issues to get notions as to how to repair their current issue. Secondly, the Emergency 9-1-1 Resiliency Platform will host a comprehensive database of PSAP information that is accessible through a web interface for technicians to quickly and efficiently access pertinent information about the network and its resources. This will allow technicians to correct any network failures with greater efficiency, as much of this information is stored on hard copies located in file cabinets, and can be accessed only by “sneaker net”. The target audience for the Emergency 9-1-1 Resiliency Platform is the system administrators and technicians who maintain these systems. This unrelenting 24/7/365 task can be made simpler by centralizing the data into an easy to access Web Interface. Auxiliary users and vertical personnel will also find great use, however, the primary target for this project is the heavy users, or system administrators.

1 Introduction

When emergency communications systems fail, and they do, time is of a crucial nature. It is the goal of the Emergency 911 Resiliency Platform to streamline this timeline by connecting people with the information they need in one, simple relational database. By consolidating information from numerous databases into one, with a clean web interface which sits on existing, common infrastructure shared by all E-911 systems.

Through consolidating data from various sources and organizing it into an easily searchable, clean web interface E9RP intends to streamline the process of handling outages into an efficient and reliable solution. The ticketing system will allow junior employees to draw on the experiences of more seasoned technicians. E9RP will employ a multi-level user system that will discern user rights by login and display level-appropriate rights menu to users, ensuring data integrity and also security measures required by the Council on State Emergency Communications (CSEC). The E9RP application will be isolated to existing, closed emergency communications networks. The single outside access for these networks is a VPN access that is primarily set aside for maintenance and system health monitoring (referred to as mission control). E9RP will be an additional server to the stack at both ends of a geographically diverse system. The system consists of two hosts, on one end there is an additional device for the database server. At the opposite host will reside an additional device for the web server processing both website and server-side scripting and processing. Weaving the E9RP into an existing, established IP scheme allows the service to be accessed anywhere on the network, as well as by the VPN. Access to different elements will, of course, be rights permitting.

2 System Environment

2.1 Conceptual Design

The system environment in which the Emergency 911 Resiliency Platform exists is a geologically diverse, multi-leveled service ring. There are two hosts which are connected by an optical network of metro-Ethernet circuits. Each remote PSAP has a ds-1 connection to both hosts on diversified UPSR utility routes. All infrastructure will be installed on top of the customer's existing network. The customer will provide clean, useable IP addresses that the new database and web servers can utilize. The database server is located at the Midland, Texas host and the web server is located at the Rankin, Texas host. In between the hosts, there has been 1.5 megabits per second dedicated for the database server and the web server to communicate. Both servers are running Microsoft Server 2012R2 with 16 gigabytes of RAM on Intel i5 processors. Each server also requires its own licensed copy of

remotely anywhere, which is software that allows remote access/administration of the servers. This is required simply because the hardware is resting inside a closed network with only VPN access for maintenance and troubleshooting. Therefore, the router and adaptive security appliances will question and could possibly block messages from an RDP connection. Whereas, Remotely Anywhere is software designed specifically for this purpose and essentially creates its own VPN to the desired equipment without having to handle any messaging through the cisco security hardware.

2.2 Software Environment

- HTML5
- PHP 5.6
- phpMyAdmin
- JavaScript
- MySQL
- Apache 2.2
- Laravel
- Bootstrap 3.0.3

The preceding software was utilized to realize the E9RP. MySQL, Apache, and phpMyAdmin were utilized for the server-side scripting and handling of all database and web-server accounting. The bulk of the website design and action-items are accomplished in HTML5 and PHP.

2.3 Hardware

- HP -ProLiant DL360 1U
- Intel XEON Dual-Core 5130/2.0GHZ
- 1GB RAM
- Dual Gigabit Ethernet

The E9RP rests on an HP ProLiant DL360 1U rack mounted at each of the host sites. Each server is running a RAID 5 mirror for redundancy. Each server is also runs teamed, dual Ethernet cards, also for redundancy. And of course, for redundancy this model of ProLiant has dual power supplies.

2.4 Network

The E9RP utilizes the existing network common to Emergency 9-1-1 systems. Each site has an umbilical connection to each host. Both hosts have redundant paths to each other. The connection rate of these depends on the particular system, but the standard rate of connections between the hosts are 10+ Mb/s, with the connections to

the remotes at a minimum of 6 Mb/s. The network architecture is summarized in figure 1. Each host, located on the left and right side of the diagram, have VPN connections that filter-through Cisco 5100 ASA firewalls before connecting directly to network switches. Once connected to the network switches, a user will have permissions contingent access to the entire network. Each host and each remote PSAP have dual umbilical network connections to the system for redundancy and security. Each remote PSAP will continue normal operation while either connection has been disrupted.

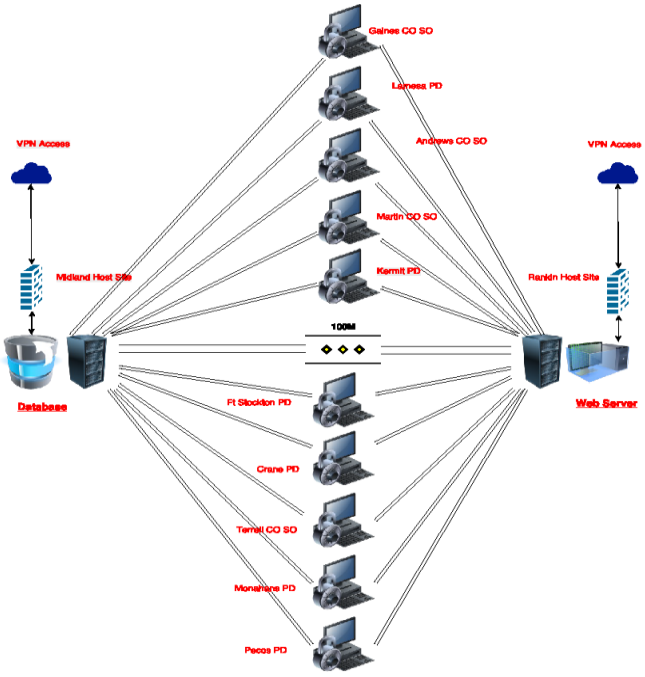

Figure 1: Network Layout

3 DESIGN

The architecture of the E9RP can be broken into two principal elements: the client-side and the server-side. Within each of these elements, many technologies collaborate to deliver the many pieces such as providing the front-end GUIs and menus to the users and system administrators, the communication between the client and server sides, and the persistence, integrity, and continuity of the data. In figure 2, the fundamental layout of the site is explained by the graphic representation of the layers and how they interact.

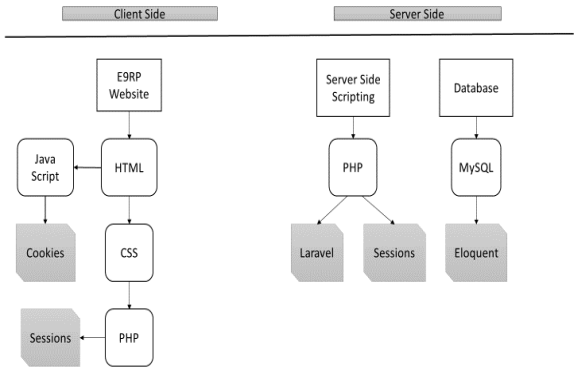

Figure 2: Client-side and Server-Side Design Diagram

3.1 Client-Side

The client-side describes the technologies used to build the graphical components for the user's web interface of the E9RP and the communication of the front-end and back-ends of the service. The client-side of figure 2 describes the web-based processes in graphical perspective. The HTML processes utilize JavaScript and cookies in order to track the user history, and PHP sessions are used to ensure that a user login has appropriate user rights to access the pages they browse.

3.2 Graphical User Interface (GUI)

The graphical user interface is built using HTML, CSS, and JavaScript. PHP is used a great detail, but PHP is only used in the GUI when the user is reading, writing, or updating a process involving the database server...as well as when changing pages within the website. Bootstrap is utilized in lieu of CSS because of its greater flexibility [5]. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code [6]. Bootstrap provides a set of stylesheets that provide basic style definitions for all key HTML components. These provide a uniform, contemporary look for formatting text, tables and forms. A graphical representation of the user interface site map is described in figure 3.

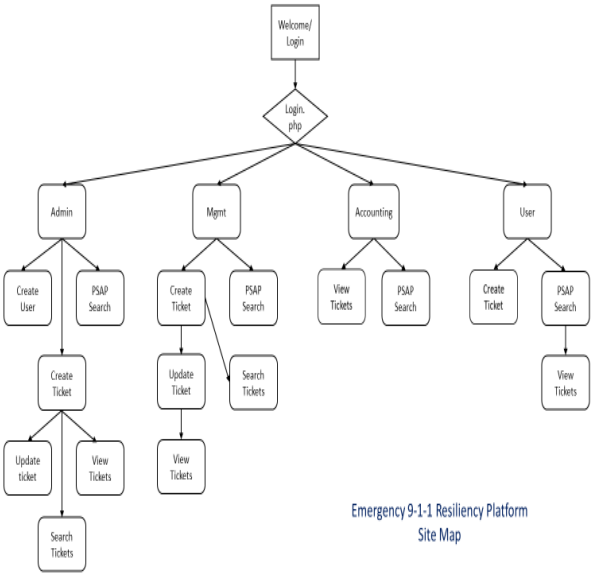

Figure 3: E9RP Site Map

3.3 HTTP GET and POST Events

HTTP GET requests are used in order to retrieve data from the database and POST method is used in order to write information to the database. These are the two principal methods by which a browser interacts with a web server [4]. The GET method, in which the browser simply requests the contents of a particular URL. When the browser uses this method, the web server ends up calling the servlet's doGet() method. The second, POST method, in which the browser "sends" data to a particular URL. For example, it may send some data the user entered in a form on a web page. In this case, the servlet's doPost() method is called.

3.4 Server-Side Design

The server-side design describes the technologies implemented in order to make the E9RP web application function, including the scripting language and the database. The Server-side processes all the PHP calls and checks to ensure the user's session is still active before allowing navigation to a new page. If the user is not authorized to view the page that is being selected, first the PHP session calls embedded in the HTML of the pages will attempt to redirect the user to the page they are attempting to view, only it will be at their assigned user rights level [3].

Second, and finally, if the redirect at the user's assigned rights level fails, then the user's session will be terminated and the user will be redirected to the initial login page. The general layout of the server-side design and the way the layers, software, and hardware interact is conveyed in the diagram in figure 2.

3.5 Authentication

Laravel loads with several pre-built authentication controllers. The RegisterController handles new user registration and the LoginController handles authentication. E9RP utilizes a user level, or user rights, policy to validate the user's permissions on navigation from the authentication web-page. The Laravel framework was invaluable tool to accomplish the initial user setup.

3.6 Database Seeding

Database seeding is the initial seeding of a database with data. Seeding a database is a process in which an initial set of data is provided to a database when it is being installed. In this case, database seeding was used for testing purposes, as well as setting up the initial administrator accounts. The database seeding saved a great deal of time because of the large amounts of data that would, otherwise, have had to been manually input before testing could occur.

3.7 Database Communication

Database communication is handled through PHP and Eloquent ORM, or Object Relational Mapper. Eloquent ORM checks against errors and ensures that the database is in third normal form and handles the schema and persistence of user and application data. PHP sessions are also used, as are cookies as an implementation of security protocols [7]. The PHP sessions preserve the database connection for the duration of the user's visit, or until the user clicks the logout button.

3.8 Database Design

The database design describes the schema of the database that is used for the E9RP web application data persistence. In this case data persistence means that the data survives after the process with which it was created has ended. In other words, for a data store to be considered persistent, it must write to non-volatile storage.

3.9 Implementation

Client-Side

The layout and the styling were created using a combination of Bootstrap and CSS. Using Bootstrap allowed an almost automated process to the repetitive processes of creating a layout and keeping it consistent throughout the website and from page to page. While CSS was necessary both inside Bootstrap and standalone. Any parts that were not universally applied to all pages were augmented with their own CSS in the stylesheet.

There are two separate instances of the E9RP, the ticketing system and the PSAP database. For the ticketing system, the user logs in with their credentials. Upon login, the webpage executes a PHP script to the server requesting user level. The user level is assigned to the user when the account is created. Only system administrator level users are able to create accounts.

Upon successful login, the user's account information is referenced in order to redirect them to the account appropriate page. Each user account is assigned a user level by the system administrator upon account creation and the user will be restricted to web pages that are assigned to that permissions group [8]. This policy is similar in nature to the active directory in Microsoft products. Once logged in, the user's permission level will be displayed at the top of the page. By displaying the user's permission level, this will remove all doubt as to the pages that are and are not accessible.

The permissions for administrator have select features that are restricted in other permission profiles. One such instance is creating new users. Creating new users is restricted solely to system administrators and this page cannot be accessed by any other user profile. If a user were to attempt to browse manually, the PHP session protocols would create an error. The error would close the PHP session and return the user to the welcome page where they would have to login to create a new session. Creating a repair request in the ticketing system, is straight-forward and another example of the application of form formatting discussed earlier. Entering all the pertinent information results in a ticket being generated, and created in the ticketing database. After successful submission, the user will be encounter a web redirect to the view ticket page. Here, all ticket in the database will be displayed. The ticketing system will only show the tickets for the last thirty days.

However, the primary motivation of the ticketing system is to allow a searchable interface for technicians. This functionality would be to serve as a reference for lesser experienced technicians. The technician could simply search by keyword (i.e. hardware or CAD). The search result would include all tickets in the database that have been tagged, closed, or otherwise modified with the keyword. This feature

creates a virtual mentor/trainer that would be available all the time. This feature would give technicians "hints" as to the nature of a solution to their issue. It is also quite rare to experience a problem that has never happened before. Therefore, in as few as twelve to eighteen months, there would exist a veritable powerhouse of solutions to potential problems.

Figure 4: Searchable Ticketing Database Example

Server-Side

The server-side implementation heavily utilized PHP for its processing forms, web traffic, and requests. PHP was selected due to its efficient and simple implementations. Simple meaning PHP integrates well with other web utilities, such as HTML. By utilizing PHP sessions, the security implementation that is being used to filter content by user permissions can temporarily store the user's access level for the duration of the website visit. The PHP sessions also keep the database connected, making the INSERT, UPDATE, and SELECT functions much more efficient [9]. Whenever a user logs in or changes pages, the PHP script will reference the users privilege (or rights) level to ensure the appropriate level is possessed by the user.

```php
<?php
session_start();
//Checking User Logged or Not
if(empty($_SESSION['user'])){
 header('location:index.php');
}
//Restrict User or administrator to Access Management.php page
if($_SESSION['user']['role']=='user'){
 header('location:user.php');
}
if($_SESSION['user']['role']=='administrator'){
 header('location:administrator.php');
}
?>
```

Figure 5: PHP Site Security Example

As for the database implementation, PHP was also the preferred tool to complete much of the heavy lifting in the processing of table information back and forth from the front-end to the back-end. PHP's flexibility offered to make short work of the tasks asked, and performed them quickly and efficiently. In utilizing PHP for the server processing, it was therefore available to fetch, process, and display queries all in one convenient language.

4 Future Enhancements

Looking to the future, there are some features that would prove useful and provide additional functionalities that would better serve the desired customer base. By adding a repository for GIS information, users would have a place to warehouse map data and Pictometry flyover data. The servers that have been deployed have many terabytes of additional space that will, otherwise, be underutilized. Another improvement that would be useful in an import utility for the billing functions. A customer could then have their accounts payable/receivable personnel export their monthly bills into the database in order to keep track of pricing and service changes. With this information, combined with the existing historical ticketing information would provide the customer with extraordinary negotiating power when it is time to renegotiate equipment purchase and service contracts.

A phone application would also be a welcomed future enhancement but would all but be required for all major mobile phone platforms. This would be time and cost prohibitive, especially for the scope of this project. Email features - Tying in with the social media integration, an email system would be an effective feature to add allowing community members to communicate through email with employers and vice versa.

5 References

[1] Bootstrap Documentation http://getbootstrap.com/getting-started/
[2] CodeIgniter User Guide https://codeigniter.com/user_guide/
[3] Community Auth Documentation https://community-auth.com/documentation
[4] jPList Documentation http://jplist.com/documentation
[5] jQuery API Documentation http://api.jquery.com/
[6] jQuery UI API Documentation http://api.jqueryui.com/
[7] MySQL 5.5 Reference Manual http://dev.mysql.com/doc/refman/5.5/en/index.html
[8] PHP Manual http://www.php.net/manual/en/index.php
[9] PHP OpenSSL Manual http://www.php.net/manual/en/book.openssl.php
[10] World Wide Web Consortium https://www.w3.org/WAI/intro/wcag

6 Images

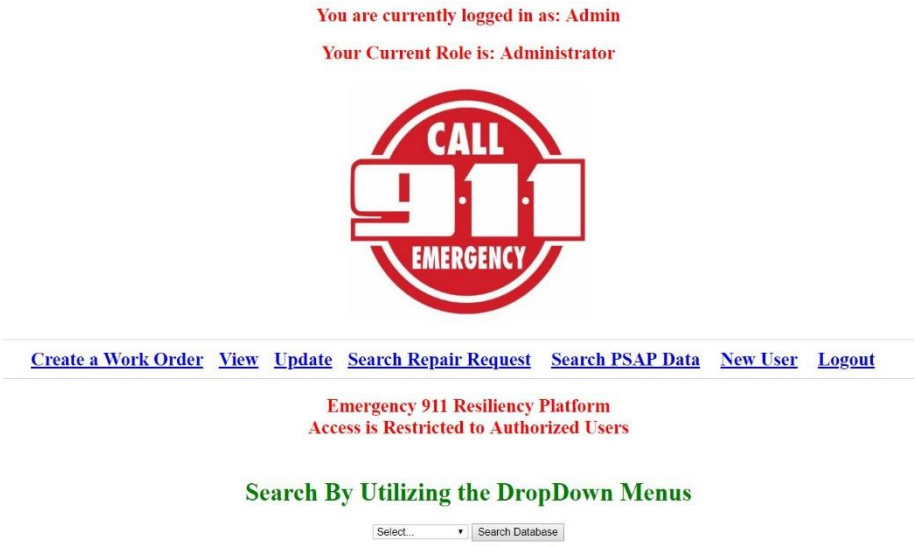

Figure 6: Successful Login Redirect Page - Visual Reference