UC San Diego

Cruz Roja

Continuity Report

ENG 100L Fall 2016

Professor de Oliveira

TA: Gabri DiFiore

Authors:

Ashley Zhao, Oliver Reyes, Khalid Almandeel Al-Otoom, Shaheryar Ajmal,

Kelvin Sarabia, Lynwhel Claricia, Alexander Adams, Oscar Ortega,

Hans Yuan, Kevin Dang, Manuel Flores, Maria Lomidze

Table of Contents

Overview

(Cited: Cruz Roja Spring 2016 Continuity Report)

Our project addresses several needs of Cruz Roja (Red Cross), Tijuana. Because the Mexican government does not run the emergency service system in Tijuana, the nonprofit Cruz Roja takes on the responsibilities of providing emergency services without charge. One issue at hand is that Tijuana has only 8 ambulances in use at a time, leading to unreasonably long response times for ambulances. In contrast, San Diego has an average of 28 ambulances (sandiego.gov). With the help of the new prevalence of mobile devices, we hope to create new technologies that will take some work out of ambulance drivers and optimize ambulance usage. Our project also addresses enhancing public knowledge of emergency devices, specifically AEDs. We aim to develop a version of crowdsourcing, allowing credible sources to add known AED locations to a database.

To help decrease Cruz Roja's ambulance response times, our project reduces the amount of information that has to be repeatedly entered into forms by the call dispatch, the emergency responder, and the hospital personnel at admission. Our ambulance display and mobile app (currently in Android) work together to streamline the information that dispatchers and drivers need to convey, thereby decreasing response times. We also plan to have data visualization for Cruz Roja so they can see where in Tijuana most ambulances are going to and which areas have the highest concentration of emergency services in order to better come up with strategies to improve response times in the future.

To get started, this quarter we are focusing on tracking the ambulances and AEDs in Tijuana. This will make it easier for dispatchers to quickly find the locations of the nearest ambulance and AED so they can be directed with ease. This first step lets us use the locations to calculate the shortest path to each emergency and create an optimal plan for the dispatchers responses. With this completed, future quarters can use this software to get closer to the main goal of the team.

Team Structure

Subteam 1: AED Web Application Team

Team Members: Kelvin Sarabia, Lynwhel Claricia, Alexander Adams, Oscar Ortega

Mission: To create a web application that allows users to know the location of the nearest AED. To do this, we are going to display the location of the AEDs in Tijuana and allow users to add, remove, and edit new and existing AEDs.

Quarterly goals:

1. Create users and give them permissions to edit, remove and edit new and existing AEDs
2. Give users the ability to add photos of the AED to the AED description
3. Translate the AED information to Spanish
4. Display the radius around each user and the region around each AED

Inventory: Django Book, Tango with Django 1.5 Book

Subteam 2: Ambulance Web Application Team

Team Members: Ashley Zhao, Oliver Reyes, Khalid Almandeel Al-Otoom, Shaheryar Ajmal

Mission: To create a web application that displays the changing status and location of ambulances in real time, and compiling this in a user-friendly web interface for medical services to use.

Quarterly goals:

1. Update database with data from mobile devices containing location and status of ambulances
2. Create a user interface that displays this data in the most effective way

Inventory: Python-Django server side framework, Leaflet map, JQuery

Subteam 3: Ambulance Location Mobile Team

Team Members: Hans Yuan, Maria Lomidze, Kevin Dang, Manny Flores

Deliverable: Android app that communicates with dedicated server to transmit information regarding ambulances' locations and statuses.

Quarterly goals:

1. Create a mobile Android app to communicate its location to database.
2. Provide ambulance drivers with a way to update their current status.
3. Make the application compatible with stable Android releases/versions

Inventory: Android Studio, repository on Cruz Roja git, Samsung Galaxy Tab A Android tablet

Quarterly Goals

Previous Quarter's Goals

Last Quarter's goals dealt with data collected by the Cruz Roja to visualize where emergency calls occurred. This data helps to predict where emergencies are more likely to occur, so the Cruz Roja can prioritize certain locations more than others. Furthermore, the team worked on a system to collect more data that would help with the planning the dispatcher responses. Lastly, the team worked on a web application to map the AED locations. This application was further developed this quarter and will likely be worked on for next quarter.

This Quarter's goals

The goal of Ambulance Location Web and Mobile Applications is to facilitate and ensure the adequate response time of ambulances in Tijuana, Mexico. La Cruz Roja is limited by the number of ambulances which in turn affects the emergency response time for many local civilians. It is our duty to develop an application that would help the well being of all Mexicans in Tijuana.

- AED Web Application
    - Complete user login and permission system
    - Display AED information based on location
    - Translate to Spanish
    - Include photos with AED
    - Include "radius" around each user
    - Include "region" around each AED
- Ambulance Location Web Application
    - Enhanced version of the AED Web app
    - Display real time location of ambulances

    - Display real time status of ambulances
    - Display hospitals and real time availability
- Ambulance Location Mobile App
    - Android version relays GPS location and call status
    - Updating status of ambulance en route
    - Hospital version relays availability
    - Utilizes Android devices

Content

Technical Details:

AED Web App Team:

- http://www.tangowithdjango.com/book17/
  Helped with User Authentication for the project, although outdated
- User Authentication
  - Created a template .html file for the page at the project directory:
    /aed/craed/templates/registration
  - Combined this template with the view and mapped it to the url
  - In the forms.py file, we created a form, that was handled by the createAccount view in views.py file

- AED Image Support -
  - Added ability to upload images into static/images folder
    - https://docs.djangoproject.com/en/1.10/topics/http/file-uploads/
    - https://docs.djangoproject.com/en/1.10/ref/forms/fields/#django.forms.ImageField
  - Did not finish deciding best places to display images
  - Started researching potential frameworks for resizing images into thumbnails
    - https://github.com/vinyll/django-imagefit looks promising for on the fly resizing and leaving original photo unchanged

- Spanish Language Support -
  - Failed to implement this internationalization through the tutorial below.
  - http://www.marinamele.com/taskbuster-django-tutorial/internationalization-localization-languages-time-zones#inter-settings
  - Don't know if it was because we didn't add the correct file to the local path.
  - Was never able to get a running example going.

Ambulance Web Application Team:

- Forms:
    - We created an Ambulance form and a Reporter form. These forms allow users to input an ambulance or a reporter into the database.
    - **1)** We first created a model in the models.py file of our object. The model would contain variables that define the Tutorials/ Documentation:
        - https://tutorial.djangogirls.org/en/
            - This tutorial improves understanding of Django and relationships amongst different files such as models.py, views.py, forms.py, urls.py, and others in order to create a fully functioning program. It also teaches basics about HTML and the connection between front-end and back-end of a Django program.
            - This tutorial is a bit low-level compared to this project, so adjustment is going to be needed
    - http://leafletjs.com/reference-1.0.2.html
    - https://docs.djangoproject.com/en/1.10/
    - object, such as license plate for an ambulance, or a badge number for a reporter
    - **2)** We then created a form class in the forms.py file. This class would have fields listed underneath it. The fields are what the user must enter when he wishes to create the object on the website. For a Reporter, the user needs to enter a badge number on the form.
    - **3)** Next, we created a class in the views.py file. This class just contains variables that link it to the object model and contains a “context_object_name” that will be used in the html file

- **4)** We then created a url in the urls.py file. This is the url that will be displayed for the form. The end of the url should say something like “create_Reporter” if the user is on the create reporter form. It also contains a parameter that is the class in the views file that you created for the form.
- **5)** We had to create a html template to display the information to the user on the website. The html template can be styled in many different ways. However, the form id has to match the “context_object_name” that was created in the views.py file
- NOTE: To actually input an object into the database, a class is necesarry in the views.py file for the form. Having just a method to render the form will display the form to the user, but any object created using the form will NOT be inputted in the database.

- Handling Requests: The real-time functionality of the ambulance map is supported via http requests, communicating an ambulance’s location from device to table, and from table to map.
  - Requesting Ambulance Data: (‘craed/ambulance_info/<LICENSE_PLATE>’)
    - An external source can receive information on a certain ambulance by forming a request containing the appropriate license plate. If the request is valid (if an ambulance with the license plate exists) the webpage will respond with a json object containing the information that pertains to the ambulance whose license plate you specified. This functionality is handled entirely in the views.py file, using the tables existing within models.py.
  - Updating Ambulance Data: (‘/craed/update_ambulance/<LICENSE_PLATE>?<COLUMN_NAME_1>=<COLUMN_VALUE_1>&<COLUMN_NAME_2>=<COLUMN_VALUE_2>’)
    - The function responsible for bringing information from the internet and into the project’s Ambulance table. There are many places where a request can go wrong

so be careful when you make one! For starters, a good request will contain a license plate that belongs to an ambulance in the Ambulance table.

        - Like the ambulance_info function, this function is fully implemented in the views.py file, updating the table specified in models.py.
        - For now, the update view will look for up to 3 values; status, longitude, and latitude. A record is retrieved from the Ambulances table and assigned new attributes based on the information contained in the request. For the location, longitude and latitude have to be extracted, converted into the long data type, and then used to create a Point object that will be passed to the record.
        - If more attributes need to be added, simply retrieve them from the request with a GET.get() method and check if what you got back was non-null. If it's non-null you're good, and you can assign the attribute of the record to it.
        - It is good practice to provide ajax and non-ajax versions of this function and any other request-style services. An ajax version can be used when you don't need user-friendly responses (like just responding with json for some back-end work), versus the non-ajax version (where you can respond with a viewer-friendly webpage).

- Webpage/ Leaflet Map:
    - ambulance_base.html: Template base html file for the Ambulance Web Application page ('/craed/ambulance_view').
        - <head> contains scripts to JQuery, Leaflet, and our Leaflet Javascript file and links to stylesheets for Leaflet and our 'ambulance_base.css' file.
        - Displays the Leaflet map and base overlay.

        - <script> section calls the function to create the map from our Javascript file. Additional Javascript can be added through the jscontent block.
        - This is the base template, meaning that we are able to add additional functionality to the basic webpage by extending it with other html files. Content from other template files will be added through the {% block %} sections.
    - ambulances_list.html: Template file adding list of ambulances to the overlay and ambulance markers to the map by extending base template.
        - Div class ambulancelist creates the list based on each ambulance entry in the database. Attach an onclick method to each list item, assigning the coordinates of the ambulance for the map to pan to when the list item is clicked.
        - Javascript section initially places the markers by calling the addMarker method for each entry in the ambulance table.
        - ajaxCall method pulls data from the database at a set interval asynchronously allowing us to update the locations and statuses of the ambulances without refreshing the page.
        - intervalUpdate uses data pulled from ajaxCall to call updateMarker from our Javascript file. This updates markers on the map to their new locations.
- leaflet_map.js: Javascript file containing webpage functionality
    - MyMap function creates the map
    - ambulance_icon implements custom ambulance icon
    - MyMarkers function calls the map and creates markers layers
    - addMarker function adds initial markers
    - updateMarker function updates location and status of current markers
    - panToAmbulance function centers map view on specified coordinates

Ambulance Location Mobile App:

- The entire repository can be found at the git repo.
- For screenshots of the GPS Activity, refer to the Appendix.
- Java Implementations("Back-End"):
    - MainActivity: Contains the single button to load the GPS screen. May add more later.
    - GPS: The screen to activate GPS, implemented in GPSTracker. Can also transmit statuses by choosing them in the dropdown menu. The function and the menu are linked.
    - GPSTracker: The functionality to invoke Android's GPS functions to return coordinates.
    - LocationPoint: The implementation to "package" the coordinate information for delivery.
    - CommunicationDatabase: May be used in the future to implement more complex functions to communicate with the database/website.
- XML Implementations ("Front-End")
    - MainActivity: a single button exists, Java implementation takes you to GPS screen
    - GPS: Contains dropdown menu to choose status, is linked to the function to send the coordinates to the website/database.
    - GPSTracker, LocationPoint, and CommunicationDatabase do not need XML implementations since they are not Android Activities.
- Budget: Tablet
    - Most development could take place on the virtual emulator, though all development can be done on a real Android device. The most important thing about the Android device is to test the real world use of the app.
    - We have a Samsung Galaxy Tab A with Android 5.1.1 to test the app and check for bugs

    - Therefore, we may need to use our own devices with Android 6 or above to test newer versions.
- Maintenance
    - Your Java code must be readable by other programmers! Refer to introductory computer science guidelines, Stack Overflow, and Google Code Style Guidelines for more information.
    - Android Studio makes having good style extremely easy, so there is really no excuse not to document the code especially when Android programming is largely plug and chug.

Reflection

Overall Performance

The team made good progress this quarter since we had clear goals in place. We were able to add user authentication and also support tracking locations of devices. We were also able to get a finished

prototype for the Web ambulance application, and get started off with the Ambulance Mobile Application. A majority of the quarter's goals were met which is great news for future teams.

One thing that we can certainly improve on is the onboarding process. Many team members had trouble installing the existing web application. We didn't have proper documentation for users working on Windows or Linux machines, and this delayed each team's proposed schedule. Furthermore, with many new members coming in each quarter, getting started takes a significantly long time. By week 3 most members were finally able to get the apps running, but still had to get familiar with the current code which took more time. If the team can get this problem fixed by week one, the progress of the project throughout each quarter will improve dramatically.

AED Web Application Team Reflection

The AED team was successful in completing some of its goals this quarter. We prioritized goals that we thought would be easier to complete. Our team divided and conquered and had each member focus on a goal and compare progress and blockers during subteam meetings. We had an original goal of building features biweekly, but once we took a look at the code we had trouble understanding how most operations were even being run by Django. We expected to see certain lines of code in our codebase, but learned that Django abstracted this from the user by use of views and forms. We recommend skimming through the Django book early on in the quarter.

The user authentication was our top priority because we only want registered users to have the ability to add and edit AED locations. Adding the ability to upload photos was another goal we achieved, allowing users to upload photos helping other user discern where the AED are located with buildings, etc. We attempted to add support for the spanish language, but were not successful in implementing this. Reason why we were unable to implement this was because we thought it would be an easy task and decided to put a hold on it. After going through the tutorial and trying to tweak certain implementations,

we were unsuccessful. If we would have read through the Django book before trying to implement the tutorial, it would be clearer and easier to follow through. That was the mistake that we made and got tangled up.

Ambulance Web Application Team Reflection

The Ambulance Web Application Team set out specific goals in the beginning of the quarter and were able to follow through with most of them and deliver a working, finished product that we intended for initially. Starting from the beginning of this quarter, we split the work into primarily a front and backend team. The front end team was in charge of everything Leaflet and map related, in terms of HTML and Javascript. This was all handled by code that would display maps, markers, ambulances and other data and display them in an organized and easy to view format on the web page. The backend team was primarily in charge of maintaining the database and making sure our server was picking up on the changes made in the status and location of the ambulances we are tracking. Dispersed here and there, we created different forms necessary to create a new ambulance or reporter. In the end we were able to attach the front and backend side as we made sure to discuss our personal progress during our weekly meetings so we could connect both ends to create and end-to-end dispatch system.

Ambulance Location Mobile Team Reflection

We have met all our goals set in the beginning of the quarter, albeit for a couple goals we took a different approach. We built a native Android app from scratch that was able to display ambulance location and time. We were also able to let ambulance drivers' select their status from a dropdown list of options. We relied on several third party APIs to allow us to successfully build our Android application.

We did run into an issue with conflicting versions of Android, where Google updated the newer versions with spontaneous permissions settings. We later resolved this issue later in the quarter.

There was nothing that could not be solved by Stack Overflow and Android Developers. The toughest part about programming in a new environment is the same as doing things in any new environment: the early learning curve. Once surpassed, the app practically puts itself together. The best recommendation an indicator of success is to pass that early learning curve of using Android Studio and Java within the first few weeks, since any time afterwards is extremely tough to fit into one's schedule. Therefore, shove the learning curve into the first few weeks so that the time after can be used overcoming mainly function-specific issues. On this point, work should be divided by function-specific tasks--there is little reason to divide by front-end/back-end since one aspect cannot be implemented without the other.

Future Goals

AED Web Application Team

The next steps for the next AED Web Application Team are:

- Effective "radius" around each user
- Effective "region" around each AED
- Where to display uploaded images
- Resizing of images for thumbnails
- Spanish Translation

We want the app to display all recorded AED locations, but place a special emphasis on AEDs within the user's reach for actual emergency use. This would require to use the user's current location and produce backend work to determine the best AED to direct a user to. Trying to get the localization to be able to get the Users' to select their preference in language.

Ambulance Web Application Team

The next steps for the next Ambulance Web Application subteam are:

- Implement mapview centering on ambulance marker when corresponding list item is clicked after the location of the initial marker has changed. (Currently only centers on the initial location of marker when page is refreshed).
- Figure out a way to offset the mapview centering when an ambulance on the list is selected so that it centers the ambulance marker on the center of the visible map. Currently, map panning centers to the absolute center of the page since the map takes up the whole page. This causes an aesthetic problem because the overlay covers the left side of the map, causing the ambulance marker to appear on the left hand side of the visible map.
- Implement updating the status of an ambulance on the list asynchronously so updated statuses show in real time. The necessary data is already pulled asynchronously; the list just needs to be updated with that data.
- Effectively link AED's to the location of the ambulances. This will involve updating the location values of the AED's to match that of the ambulances and whenever the location of an ambulance changes, the location of the AED should update as well.
- Display route Ambulance should take to reach its location. Route should be highlighted on the map and time to destination should be visible.
- Display locations of all hospitals in Tijuana and show their availability.

- Collect and monitor data such as routes ambulances take and the time it takes to get each patient to a hospital, where most emergency calls are located from, and which hospitals are most used so that the data can be analyzed and used to improve the efficiency of the ambulance program. The data can be used to tell drivers which are the fastest routes, which areas on-call ambulances should patrol, and which ambulance vehicles have accumulated the most miles.
- Expand on the information shown in the list and popups. Possible additions are displaying the destination of the ambulance, the patient's name and cause of emergency, and the names of the operators in each vehicle.

Ambulance Location Mobile Team

The next steps for the next Ambulance Location Mobile Team are:

- Implement user login. Authenticate drivers based on user login information. There was talk about combining specific aspects of each subteam project this quarter to achieve this.
- Enhance the LocationPoint class as needed.
- Implement a clock that does automatic transmission, and implement an enable switch for it.
    - Note: Never use a while loop that doesn't terminate in Android. Your device/simulator will freeze.
    - This feature will require some kind of customized list data structure since we want to both send only the newest information as well as store all pings.
- Establish a concrete way to transmit the information in the URL, whether it is "hard-coded" or with a grammar in which the database must parse.
- Implement the display so that the information that one device transmits can be seen by all drivers. In other words, one driver can see all the information transmitted by other drivers.

- Work on an easy way to port the Android code to iPhone. Note that there are software out there, even by Google, to do this translation. (This is why we focused on Android, rather than develop in parallel.)
- Add Google Maps usages to the app.

Inventory

Team Website: http://cruzroja.ucsd.edu/wiki/index.php/Cruz_Roja_Tijuana

AED Web Application Team

- All of work done this quarter is found in the "aed/craed" repository
- HTML template for user creation is in /aed/craed/templates/registration
- Go to the admin page ("localhost:xxxx/admin") to look at the users created

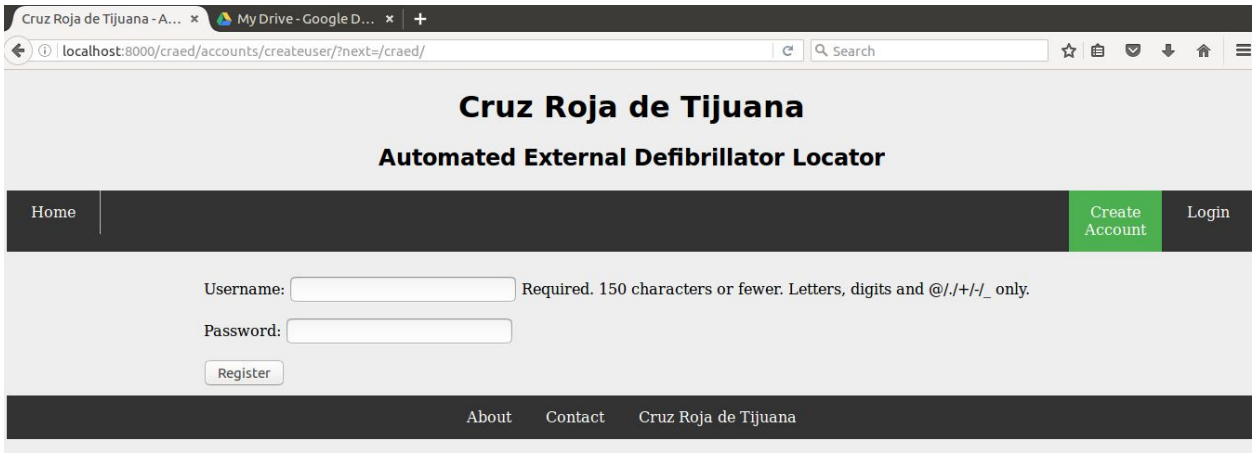
Ambulance Web Application Team

- All work done this quarter if sound in the "aed/craed" repository
- All necessary program installations will be found in the install.md document in the documents folder in the "aed" directory
- NOTE: Installation of program packages and running of project works best using a MAC OS or Linux operating system/emulator. There was great difficulty in installing the necessary packages and running the server on a Windows operating system. Recommended Linux system would be the latest Ubuntu version.

Ambulance Location Mobile Team

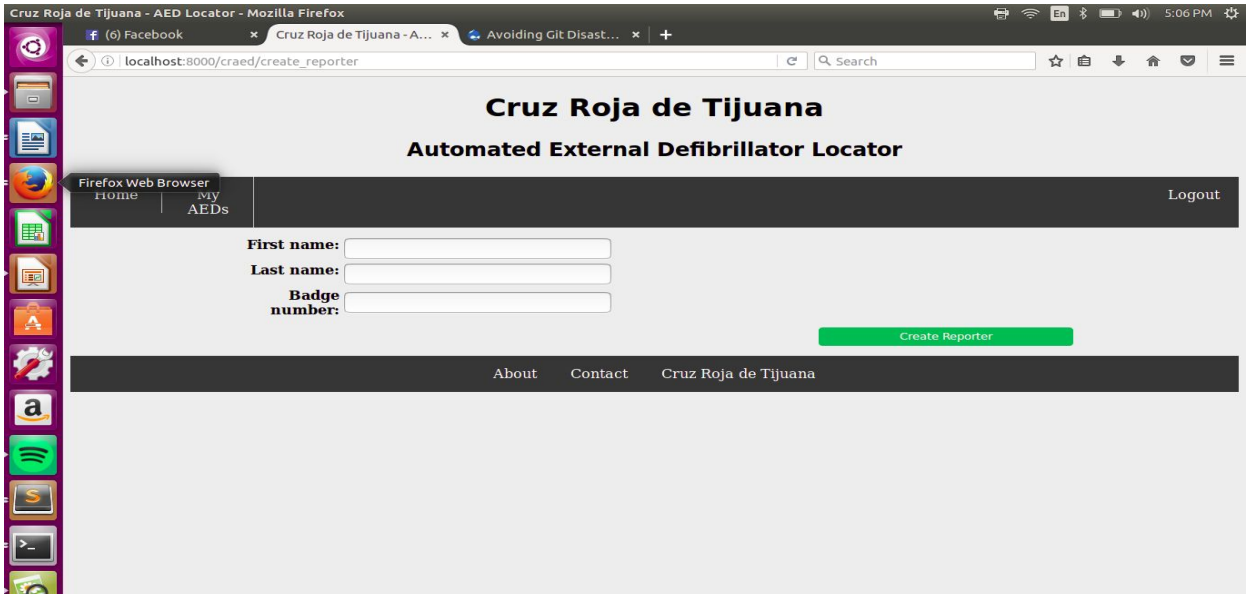
- Required:
    - Android Studio (free!)
    - Any operating system (Mac, Windows, Linux) works equally as well as each other since Google supports Android Studio on all three.
- Recommended:
    - Computer with at least 8GB of RAM for Chrome, Android Studio, and virtual emulator.
    - Android device to avoid running the virtual emulator.
    - (Windows only) OpenSSH in order to generate and use public keys for the repo.

Appendix

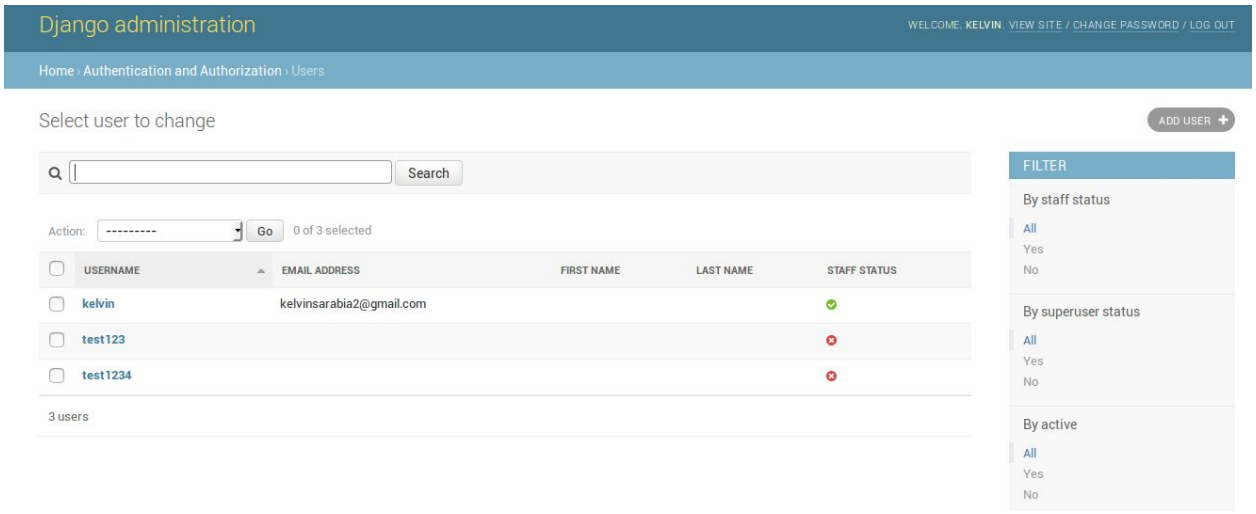


Registration page

Once Login we are redirected to AED locations page

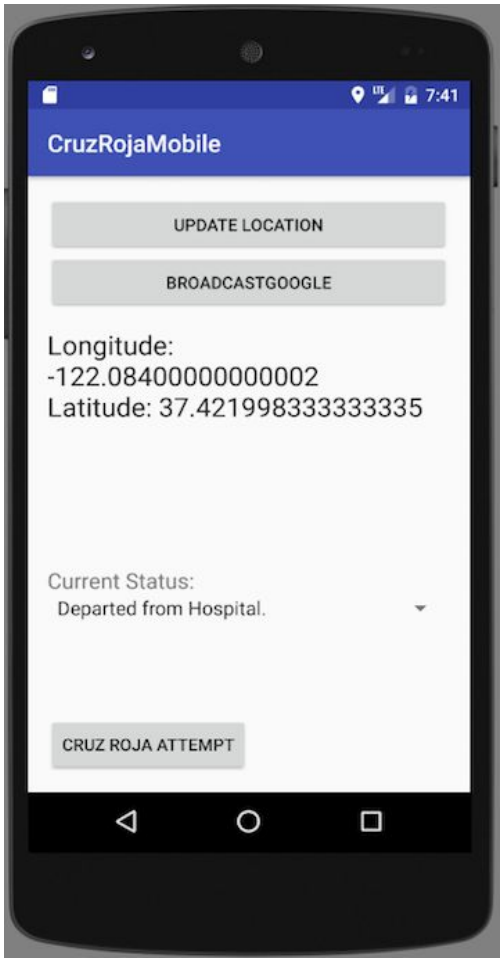


This form is used to create a reporter

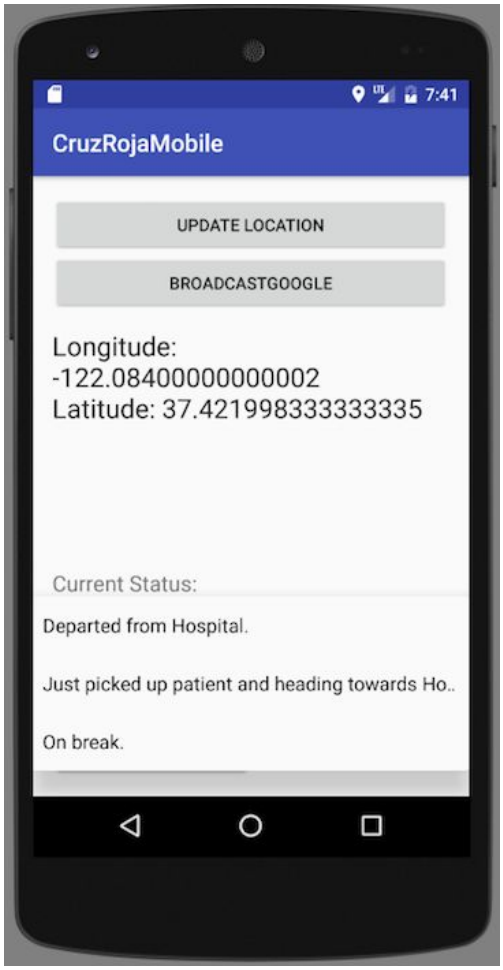


Administration page

Ambulance Location Mobile App Functionality

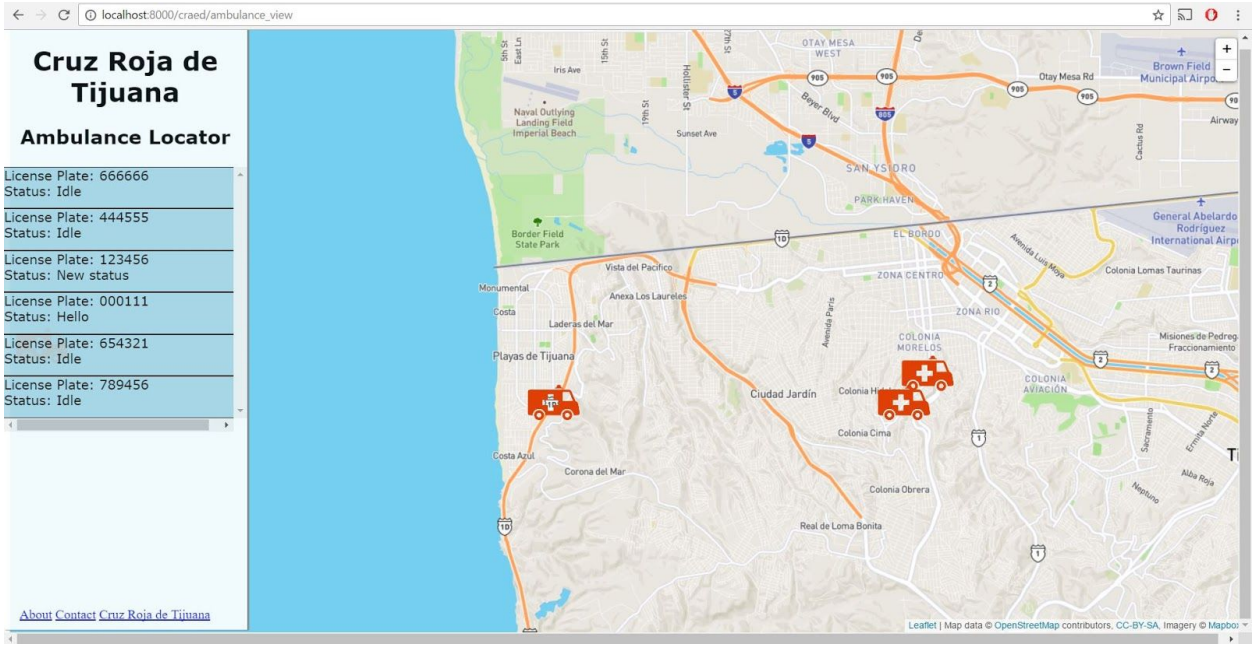


The app allows ambulance drivers to broadcast your location and send the data to Ambulance Web Application Team

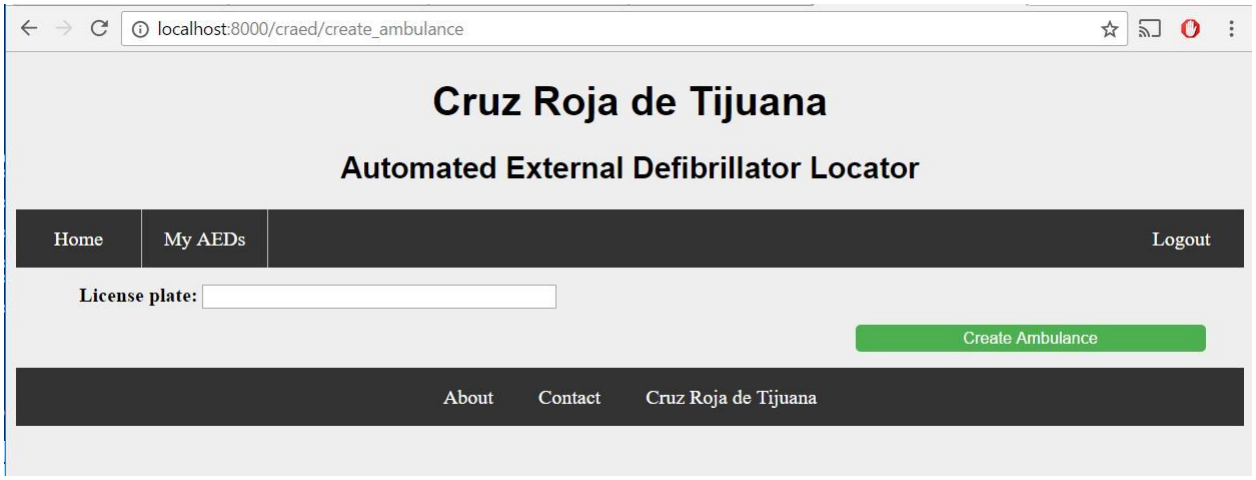


The app allows ambulance drivers to change their status and select one of the predefined status shown in a dropdown menu

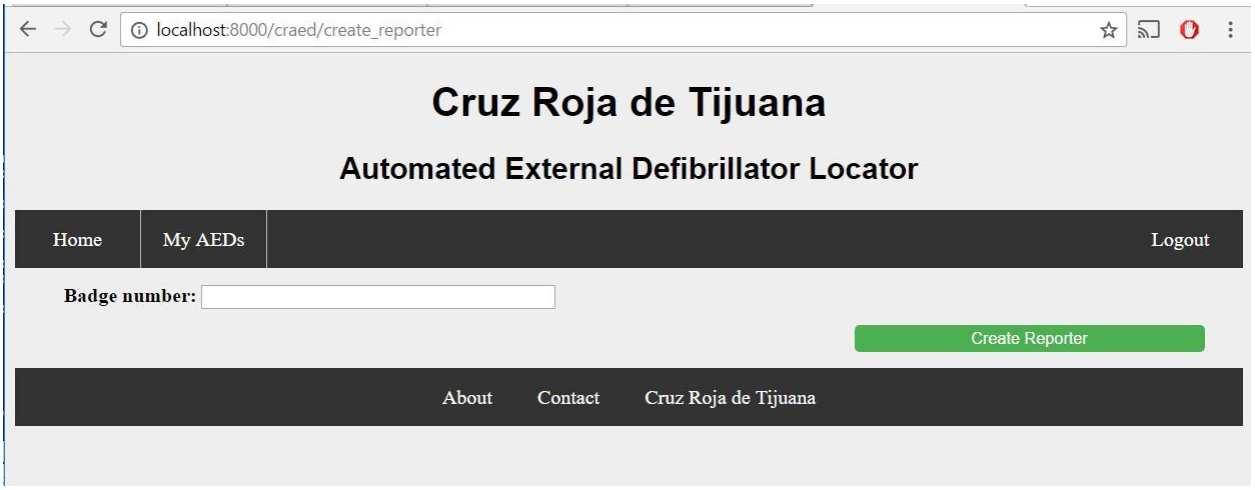
Ambulance Web App ListView



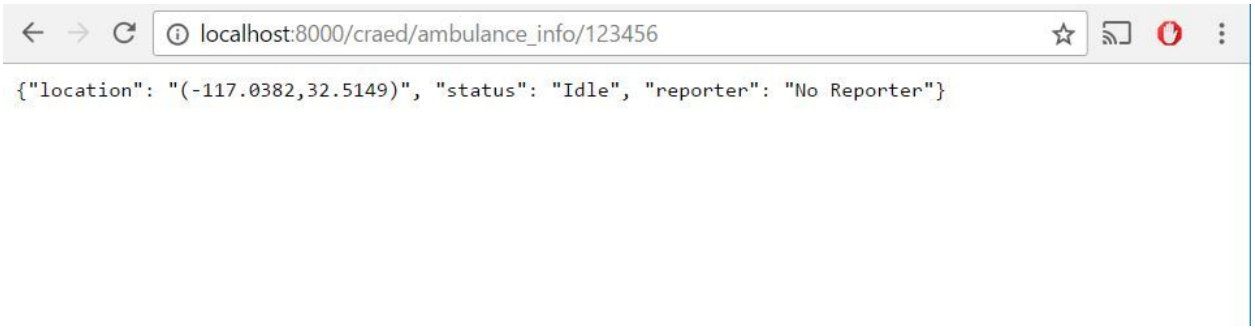
This view will contain the current location and status of all ambulances in the database.



A view of the form used to add ambulances to the database



A view of the form used to create a reporter. A reporter is associated with an ambulance for the duration of its time when it is active.



You can formulate a request for the information about a specific ambulance. If structured correctly, you will receive a json object with the ambulance's current status, location, and who is reporting on behalf of it.

To update an ambulance's information the magic is in the URL, not in what the page responds with. If structured correctly, you will simply receive a "Got it!" response. Pay attention to the structure of the URL, as the information in the URL is now the information in the table too.