Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

APLICACIÓN DE LOCALIZACIÓN PARA LOS CUERPOS DE SOCORRO

Braulio Juan Carlos Padilla Rosales
Luis Eduardo Peralta Martínez

Asesorado por el Ing. Herman Igor Véliz Linares

Guatemala, agosto de 2022

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

APLICACIÓN DE LOCALIZACIÓN PARA LOS CUERPOS DE SOCORRO

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

BRAULIO JUAN CARLOS PADILLA ROSALES
LUIS EDUARDO PERALTA MARTÍNEZ
ASESORADO POR EL ING. HERMAN IGOR VÉLIZ LINARES

AL CONFERÍRSELES EL TÍTULO DE

INGENIEROS EN CIENCIAS Y SISTEMAS

GUATEMALA, AGOSTO DE 2022

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martinez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Herman Igor Véliz Linares
EXAMINADOR	Ing. César Augusto Fernández Cáceres
EXAMINADOR	Ing. Pedro Pablo Hernández Ramírez
SECRETARIA	Inga. Lesbia Magalí Herrera López

Braulio Juan Carlos Padilla Rosales

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martinez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. César Augusto Fernández Cáceres
EXAMINADORA	Inga. Mirna Ivonne Aldana Larrazábal
EXAMINADOR	Ing. Pedro Pablo Hernández Ramírez
SECRETARIA	Inga. Lesbia Magalí Herrera López

Luis Eduardo Peralta Martínez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

APLICACIÓN DE LOCALIZACIÓN PARA LOS CUERPOS DE SOCORRO

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha septiembre de 2021.

Braulio Juan Carlos Padilla Rosales

Luis Eduardo Peralta Martínez

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala 14 de Febrero del 2022

Ing. Carlos Azurdia
Carrera de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala
Guatemala, Ciudad

Estimado Ing. Azurdia:

El motivo de la presente es para informarle que en mis labores como asesor de los estudiantes Braulio Juan Carlos Padilla Rosales que se identifica con el numero de DPI 2733171800101 y Luis Eduardo Peralta Martinez que se identifica con el numero de DPI 2520029460101 he procedido a revisar el trabajo de graduación titulado como "Aplicación De Localización Para Los Cuerpos De Socorro" siendo así a mi criterio este se encuentra completado.

Habiendo tenido reuniones periódicas con el estudiante y luego de haber revisado el trabajo mencionado, considero que cumple con los requisitos de calidad y profesionalismo que deben caracterizar a un futuro profesional de la carrera de ingeniería en ciencias y sistemas.

Sin otro particular me suscribo a usted.

Atentamente,

Ing. Herman Igor Véliz Linares
COLEGIADO No. 4836

Ing. Herman Véliz

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 1 de abril del 2022

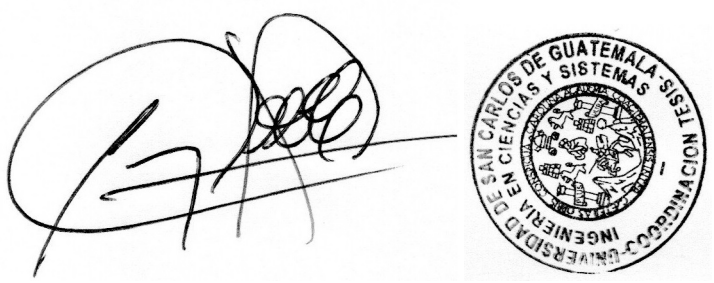
Ingeniero
Carlos Gustavo Alonzo
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación de los estudiantes **BRAULIO JUAN CARLOS PADILLA ROSALES** con carné **201213026 y** CUI **2733 17180 0101** y **LUIS EDUARDO PERALTA MARTÍNEZ** con carné **201212682 y** CUI **2520 02946 0101,** titulado**: "APLICACIÓN DE LOCALIZACIÓN PARA LOS CUERPOS DE SOCORRO",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,



Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LNG.DIRECTOR.148.EICCSS.2022

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del Asesor, el visto bueno del Coordinador de área y la aprobación del área de lingüística del trabajo de graduación titulado: **APLICACIÓN DE LOCALIZACIÓN PARA LOS CUERPOS DE SOCORRO**, presentado por: **Braulio Juan Carlos Padilla Rosales y Luis Eduardo Peralta Martínez**, procedo con el Aval del mismo, ya que cumple conlos requisitos normados porla Facultad de Ingeniería.

"ID Y ENSEÑAD A TODOS"

Msc. Ing. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, agosto de 2022

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.508.2022

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **APLICACIÓN DE LOCALIZACIÓN PARA LOS CUERPOS DE SOCORRO,** presentado por: **Braulio Juan Carlos Padilla Rosales y Luis Eduardo Peralta Martínez,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANA
FACULTAD DE INGENIERÍA

Inga. Aurelia Anabela Cordova Estrada
Decana

Guatemala, agosto de 2022

AACE/gaoc

ACTO QUE DEDICO A:

Mi madre	Gloria Rosales, por ser apoyo incondicional en mi jornada como estudiante.
Mi abuela	María Alvarado, por siempre ayudarme a estar listo día a día y enfocarme únicamente en mis estudios.
Mi hermano	German Padilla, por tolerar las molestias ocasionadas a lo largo de la carrera.
Mis compañeros	Por su valiosa ayuda para comprender y aprender a trabajar en equipo para lograr grandes resultados.

Braulio Juan Carlos Padilla Rosales

ACTO QUE DEDICO A:

Mis padres	Francis Martínez y Eduardo Peralta, por apoyarme y ser mi principal fuente de motivación y soporte durante la carrera.
Mis abuelos	Amparo Quiñónez y Fernando Martínez, quienes me brindaron su hogar y atención a lo largo de mi vida.
Mis hermanos	Roberto Peralta y Josué Peralta, quienes siempre me brindaron su ayuda cuando más los he necesitado.
Mis amigos	Kevin Barrientos, Argel Figueroa, Braulio Padilla, Javier Carpio, Daniel Chavarría, José Estrada, Samantha Barrera y Christian Escobar, quienes fueron un punto de apoyo para mí en este proceso.

Luis Eduardo Peralta Martínez

AGRADECIMIENTOS A:

Ingeniero Edgar Santos	Por su valiosa colaboración en la asesoría y revisión del presente trabajo de tesis.
Ingeniero Herman Véliz	Por su valioso apoyo en la asesoría y revisión del presente trabajo de tesis.
Facultad de Ingeniería	Por brindarme la oportunidad de obtener conocimientos para desempeñarme como profesional.
Universidad de San Carlos de Guatemala	Por ser una institución generosa que me brindó muchas experiencias enriquecedoras.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
%	Porcentaje

GLOSARIO

API	*Application programming interface* (Interfaz de programación de aplicaciones).
APK	*Android Package* (paquete de aplicación Android).
Base de datos	Conjunto de datos que entrelazados a base de relaciones forman información importante para una entidad.
Benchmarking	Comparación de las mejores prácticas llevadas por empresas que están en el mismo segmento de la empresa que lo está realizando.
Enlace	Son textos, imágenes o sitios web, que el usuario puede revisar buscándolos en un navegador web.
Framework	Conjunto de librerías diseñadas para simplificar el proceso de realización de *software* a los programadores.
GPS	*Global Positioning System* (sistema de posicionamiento global).

NO SQL	Término con el que se identifican todas las bases de datos que no utilizan el paradigma relacional para almacenar la información.
Plugin	Aplicación que se relaciona con otra, para aportarle alguna funcionalidad nueva y específica.
SDK	*Software Development Kit* (kit de desarrollo de *software*).

RESUMEN

En general, el problema que se resuelve con la aplicación descrita se debe a que, actualmente los bomberos solo cuentan con el proceso de atención tradicional, el cual consiste en una llamada con la que se reporta un accidente y una dirección o una ubicación aproximada donde ocurren los hechos, los agentes envían una ambulancia para verificar el suceso y realizar las acciones pertinentes sobre el accidente reportado, en muchos casos, les es difícil a los agentes encontrar dicha ubicación que describe la persona afectada y, por lo tanto, les es difícil brindar sus servicios.

A continuación, se explica cómo se realizó el sistema que ayuda a los cuerpos de socorro a localizar a las personas que se han accidentado, este sistema está compuesto por un sistema web programado con productos de AWS y *React* (SOSapp web) que funciona como un *dashboard* para las personas que administran las ambulancias y dos aplicaciones móviles desarrolladas en *React Native* como lenguaje de programación, una aplicación que utilizan los conductores de las ambulancias (SOSapp movil) y otra aplicación que descargarían las personas que requieran atención por parte de los Bomberos Voluntarios (EmergenciasApp), el sistema funciona a partir de cuando una persona utiliza la opción de llamar dentro de EmergenciasApp, automáticamente, se envían las coordenadas del accidentado al *dashboard* en las oficinas de los cuerpos de socorro, la persona encargada asigna a una ambulancia a la emergencia con las coordenadas enviadas, el conductor ingresa a la aplicación el código de la emergencia y en ese momento se le indica al conductor cómo llegar donde se encuentra la emergencia.

OBJETIVOS

General

Implementar una aplicación que facilite la localización de personas en estado de riesgo, peligro o contingencia por el cuerpo de Bomberos Voluntarios en el municipio de San José Pinula del departamento de Guatemala.

Específicos

1. Determinar los impactos de la aplicación en las personas del municipio de San José Pinula del departamento de Guatemala.

2. Evidenciar cuáles son las características tecnológicas del cuerpo de Bomberos Voluntarios del municipio de San José Pinula del departamento de Guatemala.

3. Realizar un *benchmarking* del mercado para tomar puntos de diferenciación para el desarrollo de la aplicación.

4. Establecer las fortalezas y debilidades de la aplicación para su uso y utilidad en el cuerpo de Bomberos Voluntarios del municipio San José Pinula del departamento de Guatemala.

INTRODUCCIÓN

La implementación del sistema que automatiza la distribución de las ambulancias para los Bomberos Voluntarios tiene como finalidad ayudar tanto a las personas que se encuentran en un accidente como a los agentes voluntarios a localizar de una manera eficiente a los accidentados, además de ayudar a quien distribuye las ambulancias para las emergencias que se presentan cada día.

El sistema por presentar fue validado con las personas que viven dentro del municipio estudiado, realizando encuestas de usabilidad, utilidad y descarga de la aplicación. Se realizó, además, un *benchmarking* sobre tres aplicaciones distintas, con el motivo de incluir algunas de las características más significativas y valiosas dentro del sistema realizado, así también crear nuevos atributos que maximicen la eficiencia de atención a los accidentados.

Las aplicaciones que componen el sistema se realizaron bajo herramientas de *software* de código abierto con comunidades que crecen continuamente y una documentación amplia para que todo el sistema no incurra en costos de licenciamiento o de mantenimiento.

1. ESTUDIO DE LA TECNOLOGÍA A UTILIZAR

Al realizar una aplicación de *software*, se debe someter a una teoría, la cual dará un acercamiento de cómo desarrollar; de tal forma, que las personas acepten la aplicación cuando la utilicen y no la terminen abandonando.

1.1. *Technology acceptance model* (modelo de aceptación de la tecnología)

Esta teoría de información establece que la aceptación y utilización de una tecnología por parte de los usuarios finales, se basa en dos parámetros los cuales son:

1.1.1. Utilidad percibida

Este parámetro le indica al usuario qué tanto se beneficiará al utilizar la tecnología.

1.1.2. Facilidad de uso percibida

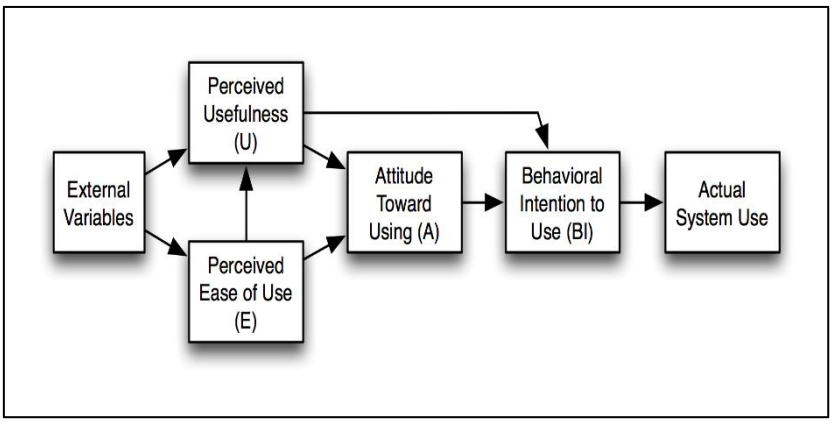
Este parámetro le indica al usuario qué tan fácil le resulta utilizar la nueva tecnología.

Cabe mencionar que, al realizar una nueva tecnología, en realidad existen más de dos parámetros que intervienen en el modelo, pero esta teoría enfatiza que lo más importantes para determinar si una tecnología será aceptada o no, son los dos anteriores.

1.2. Diagrama

El esquema de la teoría muestra cómo las variables externas influyen en percibir la facilidad de uso de una tecnología y de la utilidad de esta, estos dos parámetros motivan la actitud de las personas a utilizar la aplicación y estas a su vez determinan la intención del comportamiento, la cual determina que las personas utilicen el sistema.

Figura 1. **Esquema del modelo a utilizar**



Fuente: Bagozzi, R. P. *The Legacy of the Technology Acceptance Model and a Proposal for a Paradigm Shift.* Consultado el 10 de julio de 2021. Recuperado de https://aisel.aisnet.org/jais/vol8/iss4/12/.

Se observa que el parámetro de utilidad percibida es más fuerte que el de la facilidad de uso, ya que, aunque a alguna persona se le haga complicado aprender a utilizar la tecnología, la aprenderá si sabe que su rendimiento se incrementará al usar la tecnología.

Las variables externas del sistema son:

- Tiempo de mejora de servicios al utilizar una simple llamada
- Interfaz de aplicación fácil de utilizar y aprender
- Manuales de usuarios fáciles de usar
- Experiencia de uso

1.3. Teoría y la relación con el tema escogido

Se escogió esta teoría porque el objetivo de la aplicación es ayudar a la población guatemalteca en general, es de suma importancia que se tenga un impacto en la sociedad y no solo que sea una aplicación más, por lo que se utilizarán los principios de esta teoría para que al crear esta tecnología sea aceptada por la población.

2. ANÁLISIS DE MERCADO Y FACTOR DE DIFERENCIACIÓN

Es muy importante realizar un análisis de mercado al momento de realizar un *software*, ya que puede haber varios en el mercado que tengan funcionalidades similares, entonces, si no se posee un factor de diferenciación, probablemente la gente no va a utilizar el *software*.

2.1. Antecedentes

Actualmente, aún no se ha desarrollado ninguna tecnología innovadora en Guatemala que ayude a los cuerpos de socorro a salvar vidas, únicamente se cuenta con el proceso normal de atención, el cual consiste en realizar una llamada y dar una pequeña información del incidente a los cuerpos de socorro.

En la llamada, el agente obtiene la dirección aproximada y envía a una ambulancia a verificar el suceso, al llegar a la dirección obtenida los cuerpos de socorro realizan las operaciones pertinentes.

Este proceso no es muy eficiente y muchas veces se pierden vidas tratando de localizar el lugar exacto del incidente, debido a que la ubicación que dio la persona afectada no es exacta y dificulta la localización de lugar, es por esta razón que surge la idea de realizar una aplicación que le provea a los bomberos y a los accidentados una herramienta por medio de la cual ambas partes puedan salir beneficiadas.

Se considera que esta tecnología reducirá, considerablemente, el tiempo que tardan en llegar los cuerpos de socorro al incidente, porque estarán provistos de la ubicación exacta del mismo, haciendo así más fácil el arribo al lugar.

2.2. Observación de necesidades

Con el exponencial avance de la tecnología, es responsabilidad como ingenieros en sistemas, proveer soluciones de calidad a los problemas existentes en Guatemala.

Actualmente, los bomberos pierden demasiado tiempo buscando la ubicación exacta del incidente de donde fueron solicitados y esto repercute en que muchas veces las personas fallecen, por eso es de vital importancia un sistema confiable que ayude tanto a los bomberos como a las personas afectadas a reducir el tiempo de espera.

En muchas ocasiones se pierden vidas debido a que las personas no tienen ningún conocimiento en el área de primeros auxilios, por lo que en Guatemala es necesario que se provea de forma fácil y rápida información de primeros auxilios para que las personas puedan educarse y así actuar de la mejor manera ante algún incidente.

2.3. Mercado objetivo

Es el mercado de masas, es decir, la tecnología presentada fue realizada para el público en general, ya que, al ser una aplicación para los cuerpos de socorro, cualquier persona con un teléfono debería de ser un cliente potencial.

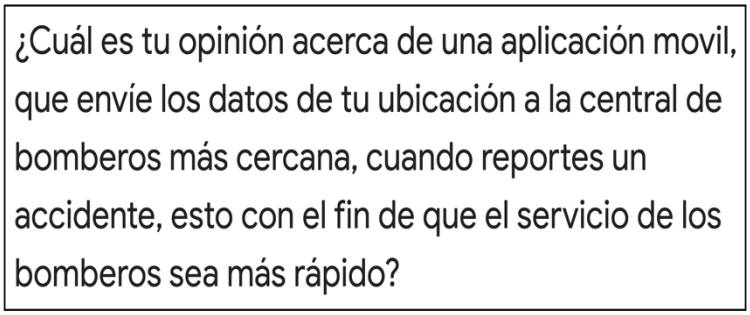
2.4. Recopilación de información

Se realizó una encuesta, esta sirvió como base para saber si realmente lo que propone EmergenciasApp satisface las necesidades de la población guatemalteca, la encuesta tiene cuatro preguntas, la información a recopilar en ella se detalla en el siguiente enlace:

http://goo.gl/forms/tU2Wt3TqkxzvMW8u1

A continuación, se muestran en la figura 2 las estadísticas de cada pregunta.

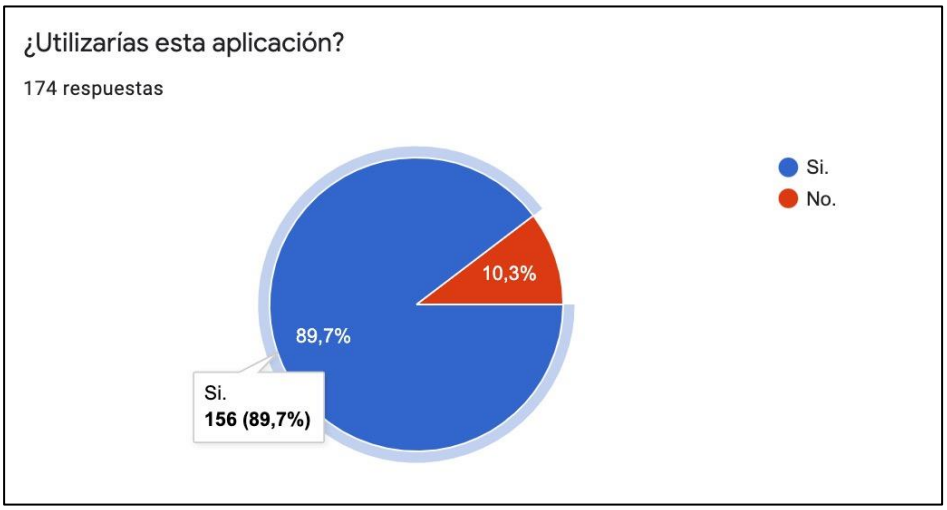
Figura 2. **Primera pregunta**



Fuente: elaboración propia, realizado con Google forms.

Aunque la primera pregunta solo fue contestada por 64 personas de las 170 encuestadas, la mayoría de los comentarios expresan estar a favor de la iniciativa.

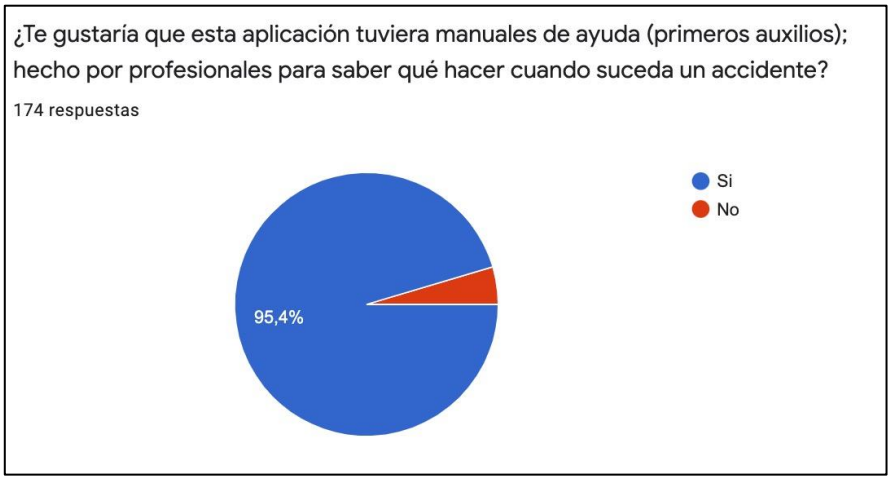
Figura 3. **Segunda pregunta**



Fuente: elaboración propia, realizado con Google Forms.

A partir de los resultados obtenidos de la imagen anterior, se concluye que las personas utilizarán la aplicación para reportar accidentes.

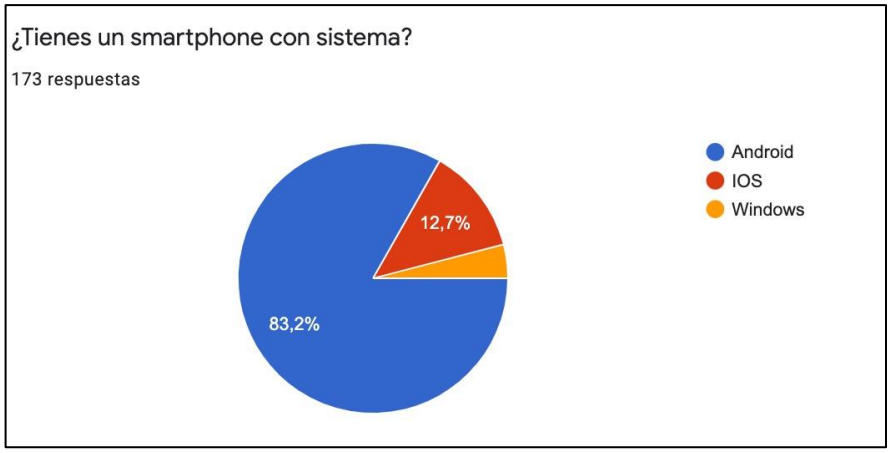
Figura 4. **Tercera pregunta**



Fuente: elaboración propia, realizado con Google forms.

El 95.3 % de la población encuestada le gustaría tener manuales de primeros auxilios, este es un dato importante, ya que los manuales son el principal factor de diferenciación con las demás aplicaciones existentes.

Figura 5. **Cuarta pregunta**



Fuente: elaboración propia, realizado con Google Forms.

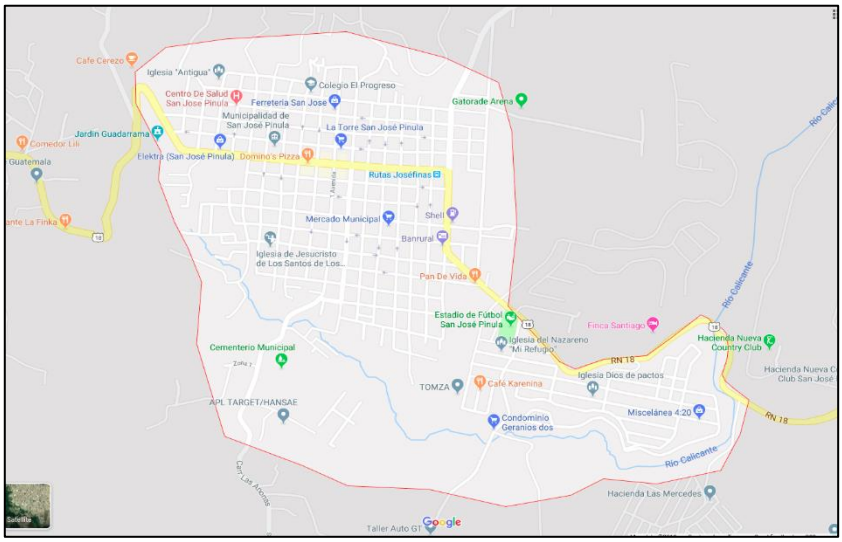
Se puede observar que el 82.7 % de la población utiliza teléfonos con sistema operativo Android, por lo que es de vital importancia que el software esté disponible de forma prioritaria en la plataforma Android.

2.5. Mercado geográfico

Esta tecnología, por el momento, se restringe únicamente al departamento de Guatemala y se espera en un futuro que esta pueda crecer para abarcar toda Guatemala, pudiendo así tener el servicio en todas las estaciones de bomberos.

En la figura 6, se muestra delimitado con color rojo el área a abarcar inicialmente por la aplicación.

Figura 6. **Mapa de San José Pinula, Guatemala**



Fuente: Google Maps. *Mapa San José Pinula.* Consultado el 21 de julio de 2021. Recuperado de maps.google.com.

2.6. Análisis de mercado de la aplicación

Al momento de realizar una aplicación es de vital importancia realizar un *benchmarking* del mercado para tomar puntos de diferenciación y además aprender de los errores que hayan cometido otras empresas en el desarrollo de su objetivo.

Para la aplicación de EmergenciasApp se decidió hacer un *benchmarking* con las aplicaciones que se describen en la tabla I.

Tabla I. ***Benchmarking***

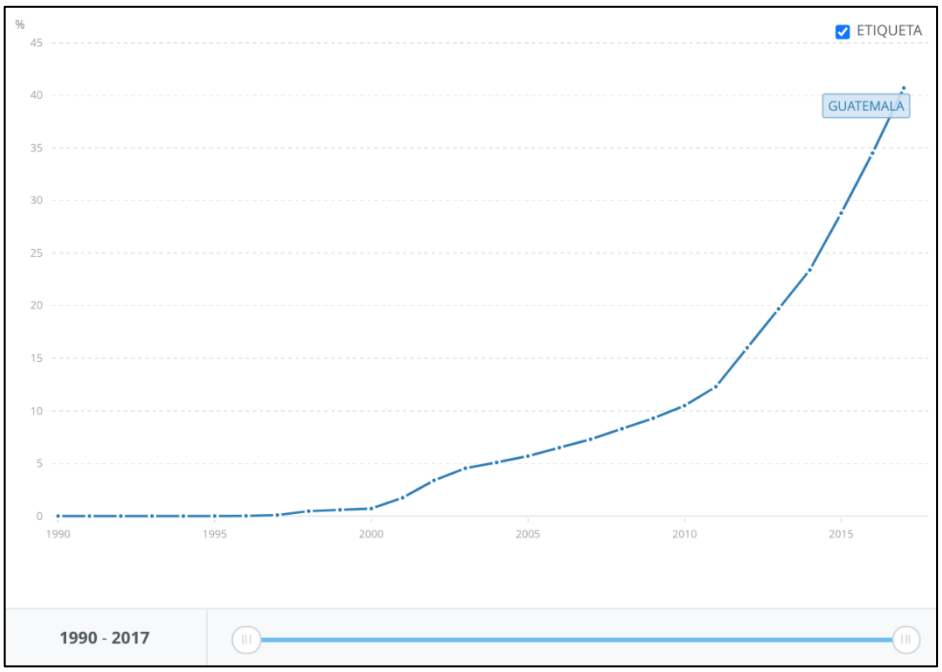
Aplicación
Alpify
911HelpSMS
RapidSOS

Fuente: elaboración propia, realizado con Word 365.

2.6.1. Marco referencial

Esta sección da un marco de referencia inicial para saber las especificaciones de la aplicación y así compararla con las demás. Un punto importante por saber es la cantidad de personas que tienen acceso a internet en Guatemala.

Figura 7. **Usuarios de internet como porcentaje de la población**



Fuente: Banco Mundial. *Personas que usan internet (% de la población).* Consultado el 21 de agosto de 2021. Recuperado de https://bit.ly/3lJG3hk.

La función de la gráfica anterior tiene un comportamiento exponencial, hasta el 2014 el 23.4 % de la población tenía acceso a internet, se estima que para el 2025 habrá más del 80 % de la población que tendrá acceso al mismo, este dato es importante debido a que los segmentos de mercado que las aplicaciones a comparar poseen, son más grandes, siendo esta la primera diferencia.

2.6.2. Fortalezas y debilidades

A continuación, se presenta una tabla de fortalezas y debilidades con aplicativos que tienen una funcionalidad parecida, esto para conocer los factores de diferenciación.

Tabla II. **Fortalezas y debilidades**

Aplicación	Fortaleza	Debilidad
EMERGENCIAS APP	• Cobertura en toda Guatemala • Simple y fácil de usar • Manuales de primeros auxilios • Mapa de ubicación actual	• No posee rastreo en directo
ALPIFY	• Página web de monitoreo • Ubicación exacta en tiempo real de la persona que denuncia • Si no posee internet envía • Mensaje de texto	• Poca cobertura • No posee retroalimentación de los usuarios • Alto consumo de energía
911HelpSMS	• Envío de mensajes de texto con ubicación. • Cobertura en más de 15 ciudades. • Localización de estaciones de servicio. • Mapa de ubicación actual.	• Sencillo • Poca calidad
RapidSOS	• Envío de imágenes del accidente. • Selección de tipo de accidente. • Localización del accidente. • Envío de mensajes a contactos de confianza.	• Funcionamiento únicamente con *wifi*.

Fuente: elaboración propia, realizado con Word 365.

2.6.3. Aplicaciones

En el presente capitulo, se describirá cada una de las aplicaciones con las cuales se hizo *benchmarking.*

2.6.3.1. Alpify

Esta aplicación posee un servidor web que da soporte a las solicitudes entrantes que realizan las personas que utilizan la aplicación móvil, es capaz de rastrear en tiempo real la ubicación del dispositivo, es gratis y sus ingresos los ganan cobrando una tarifa de servicio a las empresas que la utilizan (bomberos, rescatistas, entre otras).

Una de las particularidades de Alpify es que, si no se posee una conexión a internet, se pueden enviar mensajes de texto a los cuerpos de socorro.

Enlace de presentación del *software*:

https://www.youtube.com/watch?v=5Gl2OFfnnGs

Figura 8. **Interfaz de usuario Alpify**

Fuente: Google Play. *Locadlizador de GPS Alpify.* Consultado el 21 de agosto de 2021. Recuperado de https://bit.ly/3wMEipP.

Como se puede observar en la figura 8, su interfaz es simple, pero funcional, las verdaderas funcionalidades de la aplicación están del lado del servidor, por medio del cual van a rastrear y mostrar la ubicación en tiempo real del dispositivo.

Figura 9. **Logotipo de Alpify**

Fuente: Google Play. *Localizador de GPS Alpify- logo.* Consultado el 21 de agosto de 2021. Recuperado de https://bit.ly/3wMEipP.

2.6.3.2. 911 HelpSMS

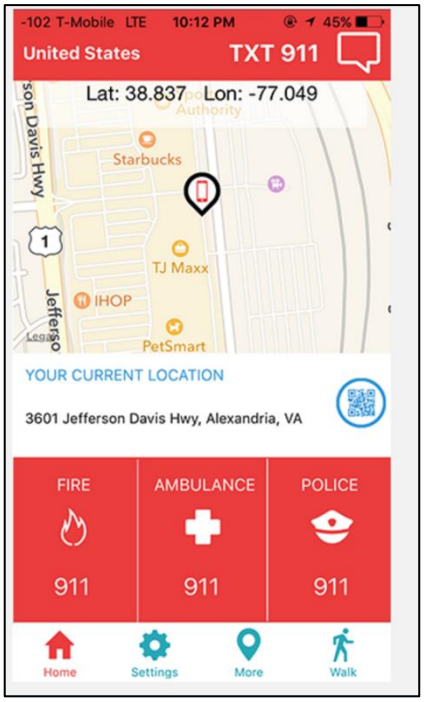
Esta aplicación permite de forma sencilla enviar información a los cuerpos de socorro, así también mandar la información a los contactos de confianza de las personas, todo esto por medio de mensajes de texto, esta aplicación, además de enviar la información, posee un mapa en el cual indica la estación de bomberos o policías más cercana, así como la ubicación actual.

Otra funcionalidad que posee la aplicación es que puede dar la ubicación de servicio más cercana, ya sean bomberos, policías, ambulancias, entre otros.

Enlace de presentación del *software:*

https://www.youtube.com/watch?v=CzyUGw0KEA4

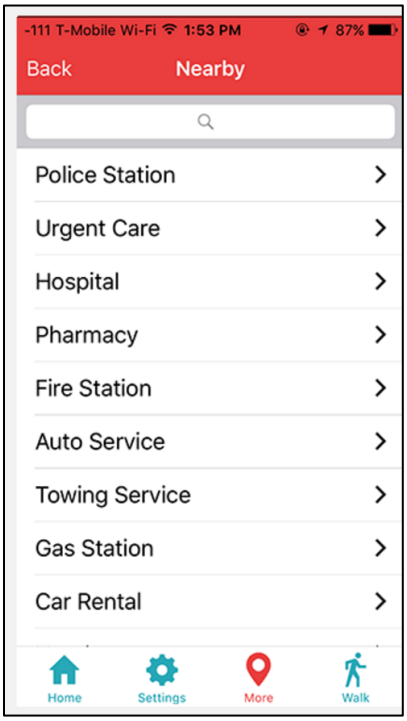
Figura 10. **Interfaz de usuario**



Fuente: 911 Help SMS. *Interfaz de usuario*. Consultado el 21 de agosto de 2021. Recuperado de https://bit.ly/3z073AO.

Como se puede observar en la figura 10, la aplicación permite indicar qué tipo de ayuda se necesita, también muestra las estaciones de servicio más cercanas.

Figura 11. **Búsqueda de estaciones de servicio más cercanas**



Fuente: 911 Help SMS. *Búsqueda de estaciones de servicio más cercanos*. Consultado el 21 de agosto de 2021. Recuperado de https://bit.ly/3z073AO.

Como se muestra en la figura 11, así es como se puede buscar la ubicación de servicio más cercana en 911HelpSMS.

Figura 12. **Logotipo de *911HelpSMS***

Fuente: Google Play. *911 Help SMS – logo.* Consultado el 20 de mayo de 2018. Recuperado de https://bit.ly/3wMlpSK.

2.6.3.3. RapidSOS

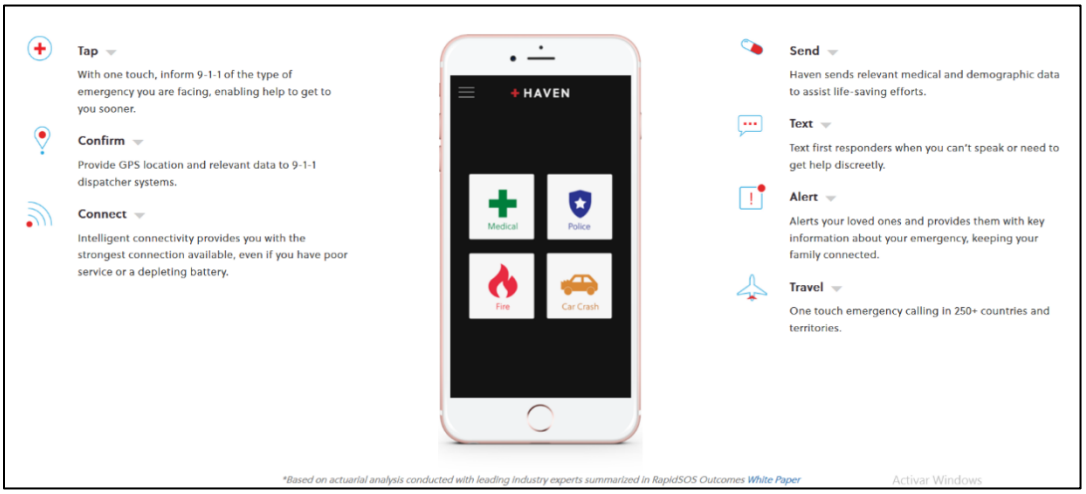
Esta aplicación, además de brindar las características que brindan las aplicaciones anteriores, proporciona las siguientes:

- Envío de imágenes del accidente.
- Envío de videos del accidente.
- Selección de tipos de accidente (medicina, policía, accidente automovilístico y fuego).
- Enviar mensajes a contactos de confianza.

Enlace de presentación del *software*:

https://www.youtube.com/watch?v=aUluO-qtQ4w

Figura 13. **Interfaz gráfica *RapidSOS***



Fuente: RapidSOS. *Interfaz gráfica.* Consultado el 20 de mayo de 2018. Recuperado de https://rapidsos.com/.

Como se observa en la figura 13, con RapidSOS se puede seleccionar el tipo de asistencia que se necesita para una emergencia, puede ser policía, bomberos, médicos y aseguradora médica.

Además, tiene otras opciones como la de alertar, la cual avisa a las personas de confianza que predefine en el momento de iniciar sesión.

3. DISEÑO DE LA APLICACIÓN

3.1. Prototipo

La aplicación de localización para los cuerpos de socorro, para teléfonos Android, consta de 3 capas que son descritas a continuación.

3.1.1. Capa de ubicación

Esta capa provee toda la abstracción necesaria para que el usuario pueda conectarse con la estación de los cuerpos de socorro más cercana.

Figura 14. **Prototipo de la capa de ubicación**

Fuente: elaboración propia, realizado con Moqups.

Esta capa tendrá tres puntos principales, el botón de SOS que automáticamente indicará la ubicación del usuario a la estación de socorro más cercana.

El segundo es un ícono que parpadea dependiendo del estado del GPS o conexión a internet del teléfono, si no se posee alguna de estas conexiones se enviará un mensaje para que el usuario active las funcionalidades.

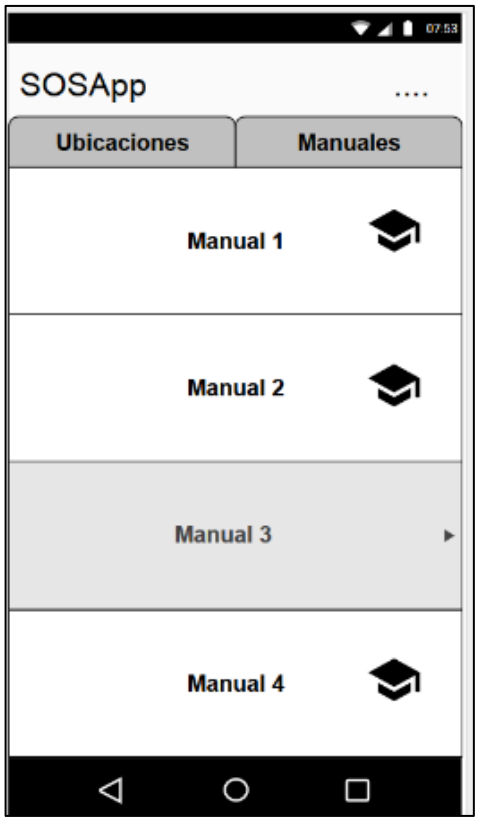
El tercero es la ubicación del usuario final, esta se marca en el mapa de Google, la cual le dará al usuario la noción de saber que su ubicación es la exacta y se ha enviado satisfactoriamente.

3.1.2. Capa de manuales

Esta capa contendrá todos los manuales que los cuerpos de socorro proporcionen para que los usuarios lean y estén preparados para actuar de forma correcta a la hora que se encuentren en un incidente.

Esta capa incorpora la principal diferenciación entre las demás aplicaciones que se compararon en el capítulo anterior, ya que EmergenciasApp, no solo se preocupa por la rápida llegada de los cuerpos de socorro al lugar del accidente, sino también, que las personas se eduquen para afrontar dichos sucesos.

Figura 15. **Prototipos manuales médicos**



Fuente: elaboración propia, realizado con Moqups.

La figura 15 describe el prototipo de cómo se mostrará el manual en la aplicación, luego de ser seleccionado de la lista de manuales.

Figura 16. **Ejemplo de visualización de un manual**

Fuente: elaboración propia, realizado con Moqups.

Para ver a los prototipos ir al siguiente vínculo:

https://app.moqups.com/brauliojuancarlos@gmail.com/MyaZpWAAG0/view/page/ad64222d5

4. DOCUMENTACIÓN BASE PARA EL DESARROLLO DE LA APLICACIÓN

Para comprender de mejor manera cómo fue realizada la aplicación móvil, el servidor y la página web, es necesario saber cuál fue la estructura de la misma, así como algunos aspectos importantes que se explican en esta sección.

4.1. Tecnologías utilizadas

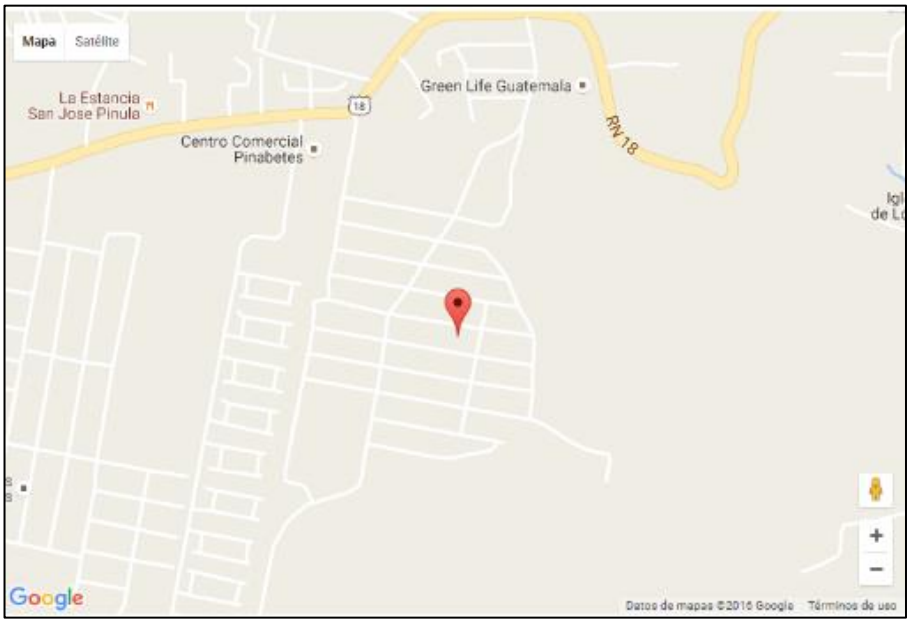
El *software* EmergenciasApp utiliza distintas tecnologías, las cuales se explican a continuación.

4.1.1. Google Maps

Esta tecnología sirve para representar en un mapa, las coordenadas GPS que se obtienen desde el teléfono que realiza la llamada a emergencias.

Además, permite realizar gráficos, trazo de rutas y marcadores en los mapas. Esto es de vital importancia para presentar al usuario información de fácil comprensión.

Figura 17. **Mapa de Google Maps**



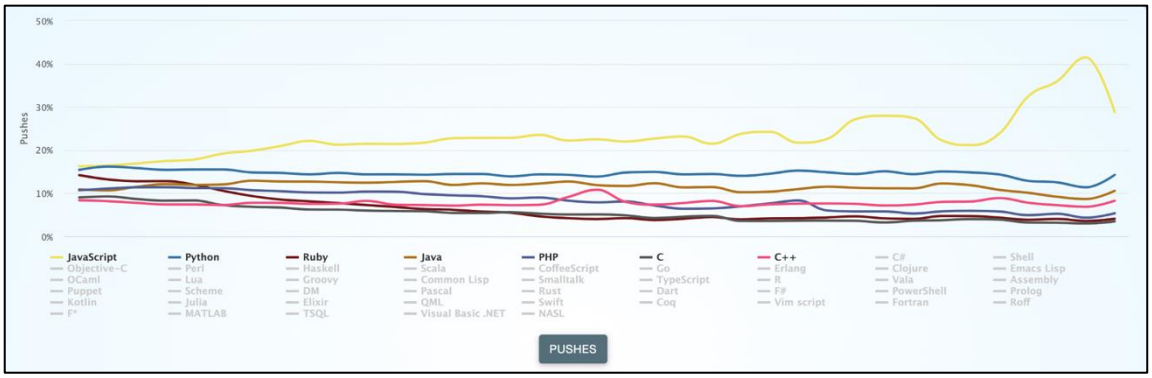
Fuente: Google Maps. *Coordenadas.* Consultado el 15 de agosto de 2021. *Recuperado de* http://www.coordenadas-gps.com/.

4.1.2. JavaScript

Es un lenguaje ligero, interpretado, puede ser orientado a objetos, funcionales, así como un lenguaje de *scripts* para páginas web, pero también para desarrollo móvil, de escritorio, de base de datos, entre otros.

Es el lenguaje más utilizado, como se observa en las figuras 18 y 19 tiene la mayor cantidad de contribuciones por la comunidad de desarrolladores en Github, que es el controlador de versiones de repositorio más grande en la actualidad.

Figura 18. **Cantidad de contribuciones de JavaScript en Github**



Fuente GitHut 2.0. *Cantidad de contribuciones en JavaScript en Github.* Consultado el 23 de agosto de 2021. Recuperado de https://bit.ly/3LSjBNJ.

Figura 19. **Cambio en la cantidad de contribuciones en el año 2021**

Year: 2021 | Quarter: 2

# Ranking	Programming Language	Percentage (YoY Change)
1	JavaScript	28.653% (+4.960%)
2	Python	14.183% (-0.033%)
3	Java	10.498% (-0.187%)
4	C++	8.222% (-0.585%)
5	PHP	5.323% (-0.387%)
6	TypeScript	4.940% (-0.151%)
7	Go	4.639% (-0.162%)
8	Shell	4.564% (+0.109%)
9	Ruby	4.051% (-0.253%)
10	C	3.463% (-0.388%)

Fuente: GitHut 2.0. *Cambio en la cantidad de contribuciones en el año 2021.* Consultado el 23 de agosto de 2021. Recuperado de https://bit.ly/3LSjBNJ.

4.1.2.1. Conceptos más importantes en el desarrollo con JavaScript

Para construir aplicativos en JavaScript es necesario entender sus características, a continuación, se presentan los conceptos más importantes

4.1.2.1.1. DOM

DOM o *Document Object Model,* por sus siglas en inglés, es una abstracción de las múltiples etiquetas que se encuentran en un documento HTML cuyas etiquetas están relacionadas entre sí y forman un árbol denominado DOM y debido a que, mayormente JavaScript es utilizado directamente con HTML para construir sitios web, se utiliza el árbol DOM para controlar los elementos y eventos dentro de una página web.

La forma de acceder a los elementos de una página web es a través del objeto documento que representa todos los componentes de una página web en un navegador.

4.1.2.1.2. Scoping

Es una parte importante en JavaScript, ya que se refiere al ámbito de la ejecución, y por ámbito me refiero al contexto en donde las expresiones y declaraciones pueden ser referenciadas o accedidas, los ámbitos pueden tener jerarquía, es decir, que pueden tener ámbitos donde los ámbitos padre acceden a los ámbitos hijo, pero no viceversa. En JavaScript las funciones crean ámbitos; en las siguientes figuras se verán los ejemplos.

Figura 20. **Error de referencia dentro de un ámbito**

```
function pruebaAmbito(){
 //Declaracion de variable
 var variableDentroAmbito = "Ambito1"
 console.log('Dentro de la Funcion')
 console.log(variableDentroAmbito);
}

//Ambito global
console.log(variableDentroAmbito); // <---- Daría un error de ámbito
         // Uncaught ReferenceError: variables Dentro Ambito is not defined
```

Fuente: elaboración propia, realizado con Visual Code.

Figura 21. **Declaración de ámbitos hijos**

```
(function ambitoPadre() {
 var variablePadre = "Padre";
 function ambitoHijo(){
   var variableHijo = "Hijo";
   console.log(variablePadre);
   console.log(variableHijo);
 }
 ambitoHijo();
})();

// <---- Imprime Padre Hijo
```

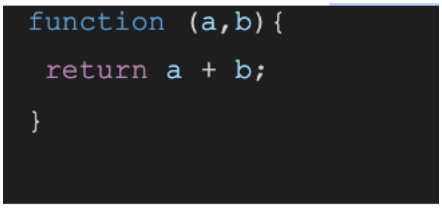
Fuente: elaboración propia, realizado con Visual Code.

Cabe mencionar que existe un ámbito global y un ámbito por bloque. El ámbito global se refiere al que todas las funciones pueden acceder y el ámbito de bloque se refiere al ámbito propio de cada función.

4.1.2.1.3. Funciones

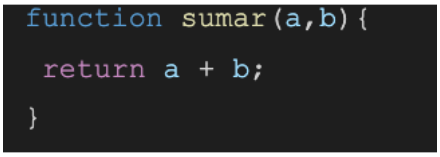
Es un conjunto de sentencias encapsuladas por un bloque que pueden ser llamadas por otra parte del código o por ella misma y existen diferentes tipos.

Figura 22. **Función anónima**



Fuente: elaboración propia, realizado con Visual Code.

Figura 23. **Función nombrada**



Fuente: elaboración propia, realizado con Visual Code.

Figura 24. **Función IIFE**

```
(function (a,b){
 return a + b;
})()
```

Fuente: elaboración propia, realizado con Visual Code.

Figura 25. **Función flecha**

```
//Funcion Normal
function (a,b){
 return a + b;
}

//Funcion Flecha
//1. Quitar la palabra function y agregar paréntesis flecha llaves () => {}
(a,b) =>  { return a + b}
//2. quitar la palabra return ya que si no se tiene más de una expresión
automáticamente retorna el resultado de la expresión
(a,b) => a + b
```

Fuente: elaboración propia, realizado con Visual Code.

4.1.2.1.4. Hoisting

Conceptualmente, por ejemplo, una estricta definición de *hoisting* sugiere que las declaraciones de variables y funciones son físicamente movidas al comienzo del código, pero esto no es lo que ocurre en realidad. Lo que sucede es que las declaraciones de variables y funciones son asignadas en memoria durante la fase de compilación, pero quedan exactamente en donde se han escrito en el código.

4.1.2.1.5. Promesas

Es un objeto que produce un valor que será ejecutado en un futuro y tiene los siguientes estados:

- Cumplida: cuando se llama al método *resolve ()*
- Rechazada: cuando se llama al método *reject ()*
- Pendiente: cuando no está ni cumplida ni rechazada

Las promesas poseen las siguientes reglas:

- Una promesa es un objeto que proporciona un método *then ().*
- Una promesa pendiente puede pasar a un estado cumplido o rechazado.
- Una promesa cumplida o rechazada se establece y no debe pasar a ningún otro estado.
- Una vez que se establece una promesa, debe tener un valor (que puede o no estar definido). Ese valor no debe cambiar.

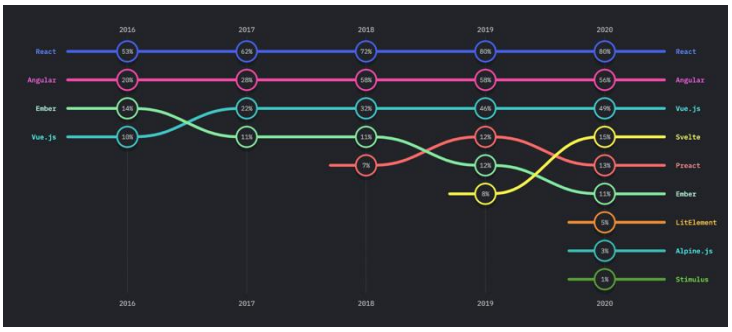
4.1.3. *React*

Las aplicaciones web actuales crecen de manera exponencial, por lo que, agregando nuevas funcionalidades, estilos y demás el código puede tornarse poco mantenible y perder el patrón de diseño y construcción del aplicativo. Para solventar esta situación, las compañías grandes de *software* han desarrollado sus propios marcos de trabajo o *frameworks*, para ayudar en la construcción de sitios web. Actualmente, existen más de 30 *frameworks* en JavaScript dentro de los más populares se encuentran: *React*, Angular, Angular.js, *Vue.js*, *Polymer*, *Backbone*.js, entre otros.

Cada uno de estos *frameworks* poseen maneras distintas de manipular el DOM, así como la estructura de modificación del *software* que se esté desarrollando.

React es una librería utilizada para construir interfaces de usuario mantenida por Facebook y por una basta comunidad en el mundo como se muestra en la figura 26 debido a su versatilidad y curva de aprendizaje corta.

Figura 26. ***Ranking* de *frameworks* en JavaScript más usados**



Fuente: GitHut 2.0. *Ranking de framewors en JavaScripts más usados.* Consultado el 23 de agosto de 2021. Recuperado de Github Language Stats (madnight.github.io).

4.1.3.1. Características de *react*

A continuación, se presentan las propiedades y conceptos más importantes para entender cómo funciona el desarrollo con *react.*

4.1.3.1.1. Componentes

En *react* todo es un componente, generalmente retorna una pieza compuesta por JavaScript, html y css, escrito en JSX descrito posteriormente y que para desarrollar en *react* se codifican pequeños componentes que pueden ser utilizados en componentes más grandes, y así sucesivamente hasta crear los aplicativos webs.

4.1.3.1.2. JSX

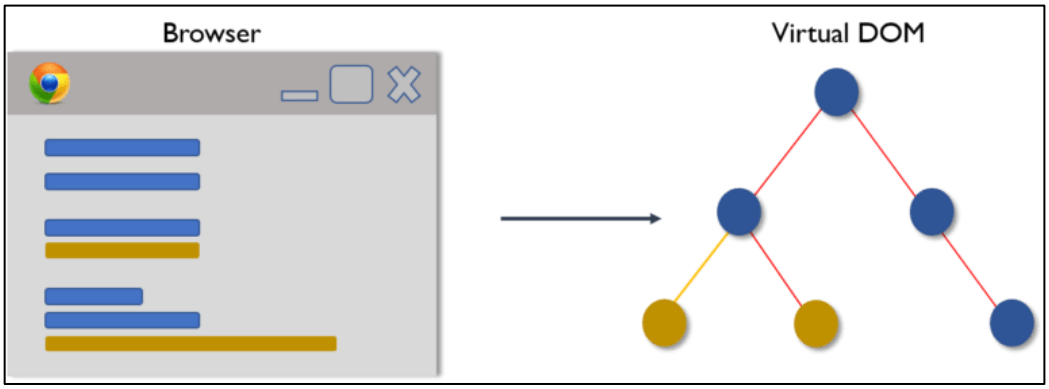
Es un concepto que, por sus siglas significa JavaScript XML y es una forma de escribir HTML en un archivo de JavaScript. Utilizando librerías externas se compila de manera que se pueden programar dichos componentes, mezclando JavaScript con html y css en un mismo archivo jsx.

4.1.3.1.3. Virtual DOM

Es un árbol de nodos que tienen los elementos y atributos de un HTML, pero la diferencia es que son componentes de *react* y que al alterar algún componente solo se actualiza el nodo afectado para luego ser actualizado en el DOM real.

Siempre que cambia cualquier dato subyacente, toda la interfaz de usuario se vuelve a representar en la conceptualización del DOM virtual.

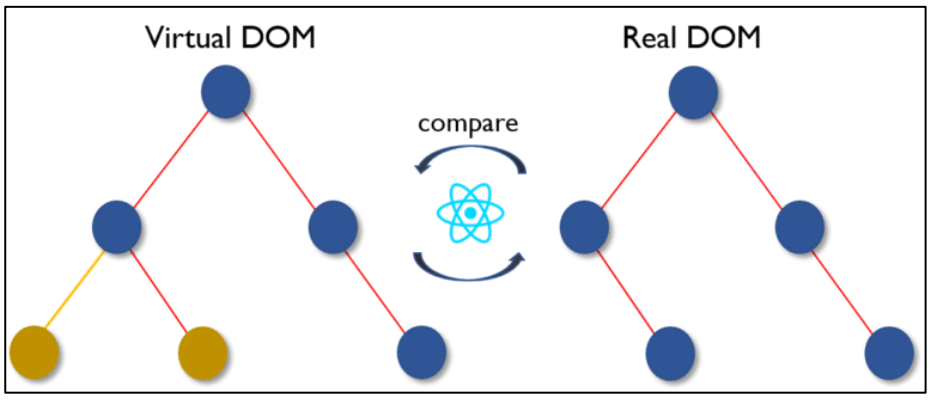
Figura 27. **Representación virtual DOM**



Fuente: Edureka. *Differentiate between Real DOM and Virtual DOM*. Consultado el 22 de mayo de 2021. Recuperado de https://bit.ly/3MTaxJE.

Luego se calcula la diferencia entre la representación DOM anterior y la nueva.

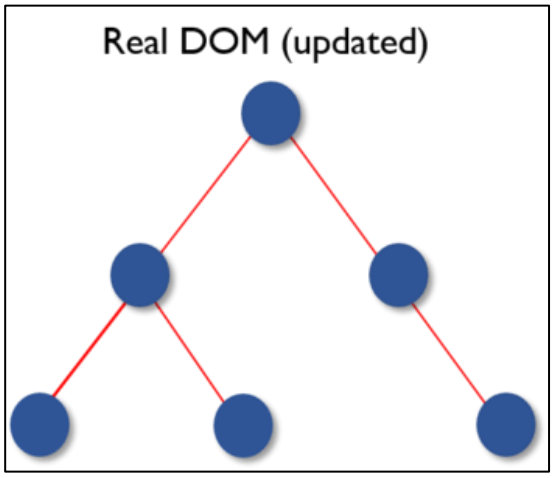
Figura 28. **Actualización del virtual DOM**



Fuente: Edureka. *Actualización del virtual DOM*. Consultado el 22 de mayo de 2021. Recuperado de https://bit.ly/3MTaxJE.

Una vez que se realizan los cálculos, el DOM real se actualizará solo con las cosas que realmente han cambiado. Se puede pensar en ello como un parche. Como los parches se aplican solo al área afectada, de manera similar, el DOM virtual actúa como parches y se aplica a los elementos que se actualizan o modifican en el DOM real.

Figura 29. **Virtual DOM actualizado**



Fuente: Edureka. *Actualización del virtual DOM actualizado*. Consultado el 22 de mayo de 2021. Recuperado de https://bit.ly/3MTaxJE.

4.1.3.1.4. Fácil de probar

Las vistas de *react* se pueden usar como funciones del estado (el estado es un objeto que determina cómo se representará y se comportará un componente). Por lo que, se puede manipular fácilmente el estado de los componentes que pasan a la vista *ReactJS* y validar la salida y las acciones activadas, eventos, funciones, entre otros. Esto hace que las aplicaciones *react* sean bastante fáciles de probar y depurar.

4.1.3.2. Propiedades de los componentes

Los componentes en *react* tienen características que le permiten interactuar consigo y con otros componentes, a continuación, se describen dichas propiedades.

4.1.3.2.1. *Props*

Es una abreviatura de propiedades, estas propiedades son solo de lectura y son trasladadas desde el componente padre al componente hijo y no en viceversa, permiten la configuración de los componentes.

4.1.3.2.2. *State*

Es la característica fundamental de los componentes, los *states* son la fuente de datos y deben ser lo más simples posible. En general, los estados son los objetos que determinan la representación y el comportamiento de los componentes. Son mutables a diferencia de los *props* y crean componentes dinámicos e interactivos.

4.1.3.2.3. *Events*

Son las reacciones desencadenadas a acciones específicas, por ejemplo, pasar el *mouse*, hacer clic con el *mouse*, presionar una tecla, entre otros. El manejo de estos eventos es similar al manejo de eventos en elementos DOM, pero existen algunas diferencias sintácticas como: los eventos se nombran usando mayúsculas y minúsculas en lugar de usar solo minúsculas. Los eventos se pasan como funciones en lugar de cadenas.

El argumento de evento contiene un conjunto de propiedades, que son específicas de un evento. Cada tipo de evento contiene sus propias propiedades y comportamientos a los que se puede acceder solo a través de su controlador de eventos.

4.1.3.3. Tipos de componentes

Debido a sus propiedades, estos pueden clasificarse en componentes clase, componentes puros y componentes de orden superior

4.1.3.3.1. Componentes de clase

Los componentes de la clase pueden hacer todo lo que hace un componente funcional, pero más. Utiliza las funciones principales de *react*, *state*, *props* y métodos de ciclo de vida del componente.

La sintaxis del componente de clase difiere de la sintaxis del componente funcional. Los componentes de la clase *exti enden de react.* El componente después de declarar el componente clase y requiere un método *render ()* para devolver el código JSX. En este componente de clase puede declarar un estado, establecerlo en un objeto JavaScript y usar accesorios para ser el estado inicial o cambiar el estado en los métodos del ciclo de vida. Algunos métodos de ciclo de vida son *componentDidMount (), componentDidUpdate ()* y *componentWillUnmount ().*

4.1.3.3.2. Componentes puros

Son componentes clase que extienden de *React.PureComponent* y una diferencia importante entre un *React.Component* regular y un

React.PureComponent es que los componentes puros realizan comparaciones superficiales en el cambio de estado. Los componentes puros se encargan de *shouldComponentUpdate ()* por sí mismos. Si el estado anterior y los accesorios son los mismos que los siguientes, el componente no se vuelve a renderizar. Normalmente se usa para optimizaciones.

4.1.3.3.3. Componentes de orden superior

Los componentes de orden superior (HOC) es una técnica avanzada en *react* para reutilizar la lógica de los componentes. No es un componente de *react* que pueda encontrar en la API. Es un patrón que surgió de la naturaleza compositiva de *react*. Básicamente, las HOC son funciones que devuelven un componente (s). Se utilizan para compartir lógica con otros componentes.

4.1.4. *React Native*

Combina las mejores partes del desarrollo nativo con *react*, la mejor biblioteca de JavaScript de su clase para crear interfaces de usuario.

Escrito en JavaScript, renderizado con código nativo. Los primitivos de reacción se procesan en la interfaz de usuario de la plataforma nativa, lo que significa que su aplicación utiliza las mismas API de plataforma nativa que otras aplicaciones.

Muchas plataformas, un *react*. Crea versiones de componentes específicas de la plataforma para que una única base de código pueda compartir código entre plataformas. Con *React Native*, un equipo puede mantener dos plataformas y compartir una tecnología común: *react*.

React Native permite crear aplicaciones verdaderamente nativas y no compromete las experiencias de sus usuarios. Proporciona un conjunto básico de componentes nativos independientes de la plataforma, como ver, texto e imagen, que se asignan directamente a los componentes básicos de la interfaz de usuario nativa de la plataforma.

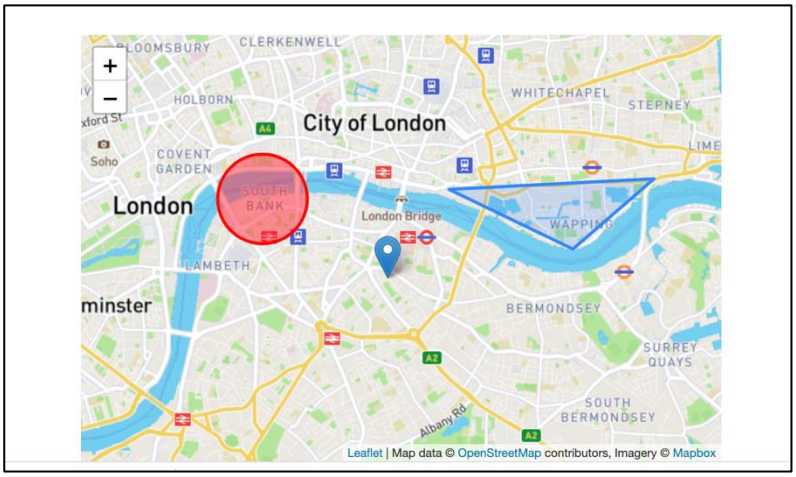
4.1.5. *Leaflet*

Es la biblioteca JavaScript de código abierto líder para mapas interactivos compatibles con dispositivos móviles. Con un peso de aproximadamente 39 KB de JS, tiene todas las funciones de mapeo que la mayoría de los desarrolladores necesitan.

Está diseñado teniendo en cuenta la simplicidad, el rendimiento y la facilidad de uso. Funciona de manera eficiente en todas las principales plataformas móviles y de escritorio, se amplía con muchos complementos, tiene una API hermosa, fácil de usar y bien documentada, y un código fuente simple y legible al que es un placer contribuir.

Marcadores, círculos y polígonos: se crean marcadores, círculos y polígonos totalmente personalizados, así como utilizar cualquier mapa disponible en la web con características que pueda utilizar *leaflet*.

Figura 30. **Mapa usando *leaflet***

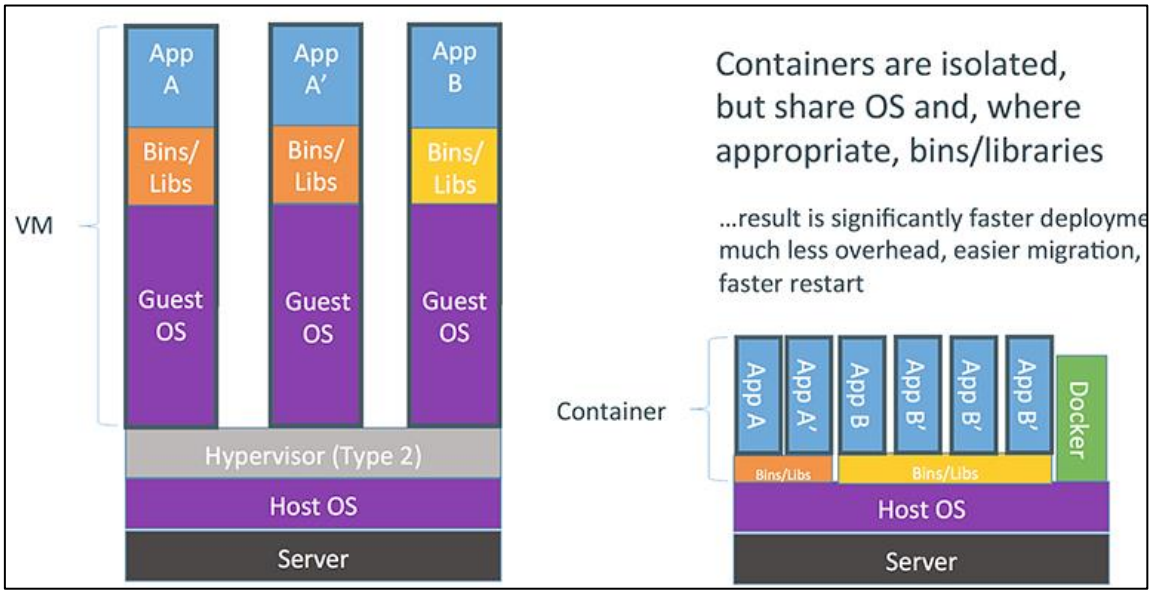


Fuente: Leaflet. *Coordenadas leaflet.* Consultado el 23 de agosto de 2021. Recuperado de https://leafletjs.com/examples/quick-start/.

4.1.6. *Docker*

Es una tecnología que ofrece virtualización a nivel de sistema operativo y es usada comúnmente en el despliegue de aplicaciones actuales.

Figura 31. ***Docker***



Fuente Docker. *Arquitectura de virtualización.* Consultado el 10 de septiembre de 2021. Recuperado de https://www.docker.com/.

En la figura 31, se muestra la arquitectura de cómo se hace virtualización tradicionalmente y cómo se hace con *docker*. La virtualización tradicional significa que la máquina que correrá la virtualización tiene que hacerse cargo de virtualizar todo el sistema operativo, este proceso es bastante pesado porque se trata de correr un sistema operativo dentro de otro sistema operativo y para esto utiliza una herramienta llamada *hypervisor*.

En contraparte, *docker* no utiliza el mismo acercamiento, en lugar de tener que replicar el sistema completo; *docker* hace uso del sistema operativo del *host* en lugar de tener replicarlo.

4.1.6.1. Imagen

Es un conjunto de instrucciones que están encapsuladas en un archivo el cual contiene paquetes, librerías y archivos que se necesita para correr la aplicación.

Una imagen provee la flexibilidad de que, cualquier contenedor que se lance a partir de ella tendrá las mismas características el uno del otro, esto es de gran utilidad en el desarrollo de aplicaciones porque provee que los sistemas de pruebas y de producción sean iguales por ello se asegura de que, si las pruebas en un ambiente son satisfactorias, lo serán en el de producción, porque son idénticas.

4.1.6.1.1. *Layer*

Una imagen es un conjunto de *layers* y una *layer* es un conjunto de instrucciones que *docker* se asegura de que en la imagen estén presentes.

Por ejemplo, una *layer* puede ser la instalación de algún paquete, por ejemplo, Java, Python entre otras, pero también puede ser cualquier instrucción que se necesite correr dentro de la imagen.

4.1.6.2. *Container*

Este es uno de los conceptos claves de *docker*, porque un *container* es un proceso por medio del cual *docker* realiza la virtualización del sistema operativo.

Los *containers* tienen la cualidad que están totalmente aislados del sistema operativo y de los demás *containers* que tiene el sistema y de ninguna manera afectará el funcionamiento de otro contenedor porque está totalmente aislado.

4.1.6.3. Repositorio

Cuando se trabaja con *docker* es necesario tener un lugar donde almacenar imágenes para que los servidores puedan descargarla y así hacer uso de ella, un repositorio de *docker* es básicamente eso, un lugar donde se pueden almacenar las imágenes para posteriormente usarlas.

4.1.6.3.1. Tags

Cuando se guarda una nueva versión de una imagen en un repositorio se necesita darle un único ID, en este caso los repositorios de *docker* manejan este nombre como *tag*.

4.1.7. *Amazon Web Services*

Amazon Web Services es uno de los principales proveedores de nube en el mundo. Este provee servicios de cómputo, de almacenamiento, de redes, de correo electrónico, entre otros.

La nube ofrece muchos beneficios en comparación con utilizar servidores *on-premise*, en primer lugar, no se necesita hacer una gran inversión al equipo de cómputo, que es muy importante porque cuando se crea un *software,* realmente no se sabe si va a ser utilizado o no.

Al no tener que hacer la compra del equipo desde el inicio, Amazon cobra únicamente por la cantidad de horas que se usan los servidores, da la flexibilidad de que se puede empezar con menos dinero y ese restante se puede utilizar en otro tipo de actividades que sean más beneficiosas para la aplicación.

Además, en el caso de que aplicación se empiece a usar por más usuarios no es necesario comprar más equipo físico de cómputo, sino que rápidamente se puede crear más instancias y pagar justamente por lo que se utiliza. También, en el caso de horarios donde la aplicación no se utilice tanto, se pueden crear políticas para hacer que la cantidad de servidores que se tiene sea óptima respecto al uso que se tenga, entonces esto provee que los costos de cómputo sean menores.

4.1.7.1. Tipos de servicios ofrecidos en la nube

Cuando se refiere a la nube, en general, se tiene que entender las tres formas principales que las empresas ofrecen servicios, las cuales son infraestructura, plataforma y *software*.

4.1.7.1.1. Infraestructura como servicio

La principal característica de estos servicios es que el cliente se encarga de manejar desde el sistema operativo, esto incluye actualizaciones del sistema operativo, la seguridad de la máquina virtual, por ejemplo, quién va a ingresar a la misma, qué aplicaciones se instalarán en la máquina, qué información se va a guardar y, por último, también el cliente es encargado de crear los procesos para que las aplicaciones finales corran correctamente.

Los principales beneficios de utilizar el servicio de infraestructura en Amazon *Web Services*, es que se puede tener control total de la infraestructura, no se ata a alguna nube en específico y si es bien utilizada, se obtienen precios más bajos.

Su principal desventaja es que se necesita a algún experto que conozca bastante de la materia, para que realmente haga una buena implementación y así no desperdiciar recursos.

El mejor ejemplo de esta este tipo de recursos son las máquinas virtuales que ofrece Amazon *Web Services*.

4.1.7.1.2. Plataforma como servicio

Esta, a diferencia de la infraestructura como servicio, todos los servidores que se son creados van a ser manejados por el proveedor de nube y el cliente únicamente será encargado de la capa de aplicación y de la capa de los datos.

Eso significa que, únicamente se proveerá el código de la aplicación y el proveedor se encargará de que los servidores se mantengan corriendo, a diferencia de la infraestructura como servicio donde el cliente tiene que asegurarse de que los servidores se mantengan en buen estado.

La principal ventaja de la plataforma como servicio es que la empresa puede salir más rápido al mercado y no se necesita de algún experto que realice la infraestructura, ya que estas plataformas son mucho más fáciles de utilizar.

La principal desventaja de la plataforma como servicio es que el cliente se compromete a la tecnología del proveedor y cambiar de una a otra tiende a ser

complicado debido a que el código se crea específicamente para que corra en la plataforma.

El mejor ejemplo de este este tipo de recursos ofrecidos por Amazon *Web Services* es llamado *Elastic Bean Stalk.*

4.1.7.1.3. *Software* como servicio

En el *software* como servicio, el proveedor proporciona el producto completo, él es el encargado de mantener la aplicación funcionando, de asegurarse de que la información se guarde de forma correcta. El beneficio de este tipo de soluciones es que los clientes no se tienen que preocupar por nada más que en aprender a usar el *software* correctamente.

El mejor ejemplo de esta este tipo de recursos ofrecidos por *Amazon Web Services* sería la llamada *Simple Email Service*.

4.1.8. Servicios específicos utilizados en la aplicación

A continuación, se describirán las tecnologías que se utilizaron para hacer el despliegue de la aplicación en la nube de Amazon *Web Services*.

4.1.8.1. AWS ECS

ECS es un servicio que Amazon *Web Services* provee para orquestar contenedores. Es un sistema fácil de usar, manejar y de escalar aplicaciones que utilizan contenedores.

De los principales beneficios de ECS es que está fuertemente integrado con otros servicios de Amazon *Web Services,* por lo que provee una forma segura y de fácil uso para integrarse con el resto de los servicios de Amazon *Web Services*.

Figura 32. **Logotipo de AWS ECS**

Fuente: AWS products. *Logotipo de AWS ECS.* Consultado el 10 de septiembre de 2021. Recuperado de https://aws.amazon.com/ecs.

4.1.8.1.1. Integración con AWS

Puesto que es una herramienta de Amazon *Web Services*, ECS es muy fácil de integrarlo con las herramientas de integración continua y herramientas de automatización desarrollados por Amazon *Web Services*.

4.1.8.1.2. No existe panel de control

En la mayoría de las herramientas de orquestación de contenedores, generalmente se tiene un panel de control conformado por nodos principales.

En este caso, ECS no necesita de ningún panel de control que estar monitoreando, porque está embebido en la solución.

4.1.8.1.3. Opción para no manejar nodos

Utilizando ECS, se puede utilizar fargate, el cual hace que no se tenga que manejar instancias para lanzar los contenedores, Amazon *Web Services* se encargará de que los contenedores corran en instancias sin necesidad de administración.

4.1.8.1.4. Reducir costos de cómputo

Al utilizar contenedores se hace uso total de la cantidad de memoria y CPU de los servidores, por medio de realizar una correcta reservación de recursos para cada contenedor y teniendo que escalar a utilizar nuevos servidores solo cuando sea necesario.

4.1.8.2. Amazon API Gateway

Es un servicio manejado por Amazon *Web Services* que hace que sea muy sencillo para los desarrolladores crear, publicar, mantener, monitorear y hacer seguras API a cualquier escala.

Una API actúa como la puerta de entrada de las aplicaciones para acceder a la data, lógica del negocio o funcionalidad del *backend*.

API Gateway maneja todas las tareas que están relacionadas con aceptar y procesar desde cientos hasta miles de llamadas concurrentes, esto incluye el

tráfico del internet, soporte para CORS, autorización y control de acceso, monitoreo y el manejo de las versiones de las API.

Figura 33. **Amazon *API Gateway***

Fuente: AWS products. *Amazon API Gateway.* Consultado el 10 de septiembre de 2021. Recuperado de https://aws.amazon.com/api-gateway/.

4.1.8.2.1. Desarrollo eficiente de API

Con Amazon *Gateway API*, se puede lanzar distintas versiones de una API, para probar cambios, nuevas características o simplemente nuevos ambientes y los costos que se generan son únicamente de las llamadas a las API que se haga, es decir, si no se están haciendo peticiones no tendrá ningún costo.

4.1.8.2.2. Alto rendimiento y bajo costo

Amazon *Gateway API* está diseñado para manejar desde cientos a miles de peticiones por segundo, por lo que es una solución que se adaptar a cualquier tipo de condiciones, además tiene costos muy bajos, comenzando desde un millón de peticiones por menos de un dólar.

4.1.8.2.3. Monitoreo

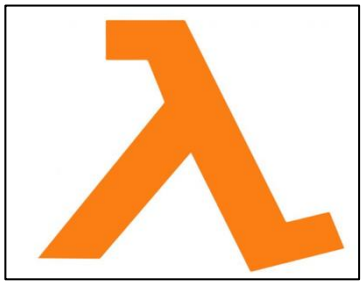
Amazon *Gateway API* se integra de manera nativa con todos los servicios de monitoreo de Amazon *Web Services*, por lo que no se tiene ningún problema para saber cuál es el estado de una API.

4.1.8.3. AWS LAMBDA

Lambda constituye una o muchas funciones escritas en algún lenguaje de desarrollo soportador por la herramienta, ejemplo: Java y Python.

Estas funciones tienen la peculiaridad de que no se necesita proveerle nada más que el código de la función y Amazon se va a encargar de crear la infraestructura necesaria para que el código que se le configuró se ejecute correctamente, es por lo que lambda se le conoce como un servicio *serverless*.

Figura 34. **AWS LAMBDA**



Fuente: AWS products. *AWS Lambd.a* Consultado el 20 de septiembre de 2021. Recuperado de https://aws.amazon.com/lambda/.

Entre las características que tiene Lambda es que es dirigida por eventos, es decir, se crean las funciones, pero se tiene que especificar en qué momento

se requiere ejecutar, es por esa naturaleza que se pueden usar para una variedad de tareas, por ejemplo:

- *Backends* de aplicaciones móviles y web
- Procesamiento de datos
- Tareas de monitoreo y alerta

Como se puede observar en la lista anterior, Lambda tiene una gran gama de utilidades, en este caso Lambda será usada como el *backend* del *software* y será *AWS API Gateway* quien decida qué Lambda correr.

Además, Amazon garantiza que en el 99 % del tiempo existirá la capacidad suficiente para lanzar cualquier cantidad de funciones, teniendo ese respaldo, la API será construida con base en esta herramienta.

4.1.8.4. AWS ROUTE 53

Este servicio es un sistema de nombres de dominio, diseñado para que sea altamente escalable y con una alta disponibilidad.

Al ser un servicio que Amazon *Web Services* provee, ofrece mucha flexibilidad en la manera en que se puede conectar los *DNS records* a los servidores, en este caso a *AWS API Gateway*.

4.1.8.4.1. Flexible

Amazon *Route 53* permite especificar reglas basadas en múltiples criterios como: ubicación geográfica, latencia y estado del *endpoint*. Se pueden configurar

múltiples políticas para el tráfico, de tal manera que se asegura que, donde sea los usuarios siempre tendrán la mejor respuesta.

4.1.8.4.2. Facilidad de uso

Una de las principales características es que al ser un servicio de Amazon *Web Services* da una gran facilidad para conectarlo con otros servicios, por lo que no se tiene que recurrir a algún tipo de especialización, porque ya *Route 53* se integra con los servicios de AWS.

4.1.8.4.3. Escalable

Diseñado para escalar automáticamente ante cualquier volumen de peticiones sin intervención manual.

4.1.8.4.4. Seguro

Por medio de políticas es posible restringir el acceso y dar permisos de forma específica, únicamente a las personas que necesitan acceder a este servicio.

Figura 35. **AWS ROUTE 53**

Fuente: AWS products. *Amazon Route 53.* Consultado el 20 de septiembre de 2021.Recuperado de https://aws.amazon.com/route53/.

4.1.8.5. AWS DynamoDB

Es una base de datos NoSQL de tipo clave valor, *serverless*, totalmente administrada por Amazon. Ofrece seguridad integrada, *backups* continuos, replicación automática en múltiples regiones y caché en memoria.

Una de sus cualidades es que permite iniciar con poca cantidad de información y si la aplicación lo demanda puede escalar globalmente hasta soportar *peta bytes* de data y cientos de miles de millones de *request* de lectura y escritura por segundo.

Figura 36. **Amazon DynamoDB**

Fuente: AWS products. *Amazon DynamoDB.* Consultado el 10 de octubre de 2021. Recuperado de https://aws.amazon.com/dynamodb/.

4.1.8.5.1. NoSQL

DynamoDB soporta bases de datos tipo clave valor y basada en documentos, por lo que cada fila puede tener cualquier número de columnas en determinado momento, dando la flexibilidad de hacer una estructura de la base de datos como más convenga.

4.1.8.5.2. Baja latencia

Si se activa la funcionalidad de alto rendimiento, *DynamoDB* es capaz de proveer acceso a datos con microsegundos de latencia.

4.1.8.5.3. Replicación multiregión

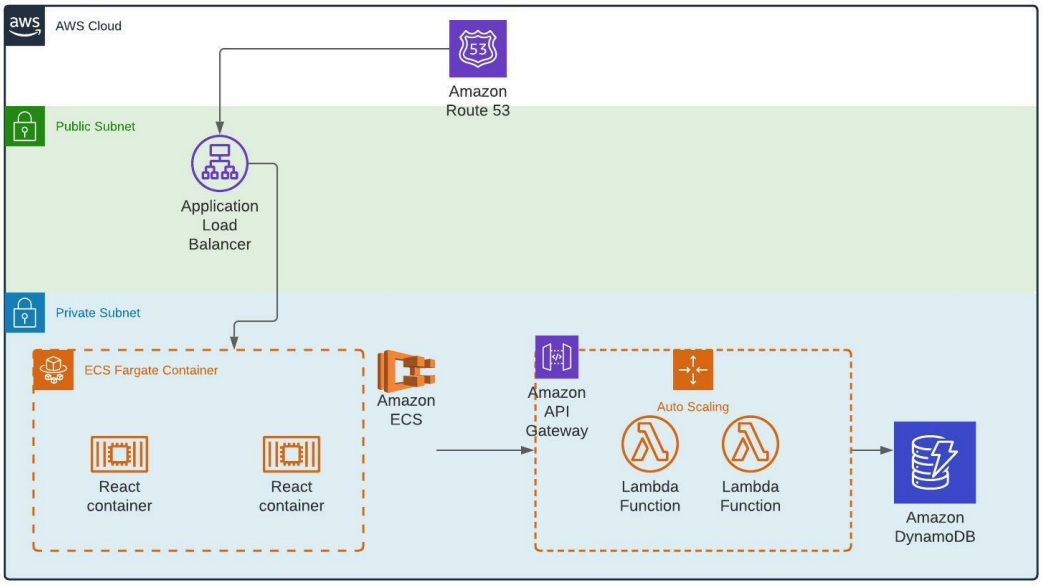
DynamoDB soporta replicación automática en múltiples regiones dando la flexibilidad de tener los datos seguros.

4.2. Arquitectura de la aplicación

En la sección anterior, se mostraron todas las herramientas utilizadas para desarrollar la arquitectura de la aplicación, en esta sección se explica cómo los componentes se relacionan entre sí para dar como resultado la aplicación funcional.

La figura 37 muestra el diagrama de la arquitectura de la aplicación.

Figura 37. **Arquitectura de la aplicación**



Fuente: elaboración propia, realizado con Lucidchart.

Como se puede observar en la imagen anterior, todo comienza por una petición de *Route 53* que es creada por un usuario al hacer un *request* a la aplicación, luego esa petición pasa por un balanceador de carga, el cual redirige el tráfico a un *cluster* de ECS, este *cluster* está configurado para escalar automáticamente y corre contendores *react*, los que a su vez hacen peticiones a

la API creada con Amazon *API Gateway* y esta ejecuta funciones Lambda, las cuales actualizan la base de datos *DynamoDB*.

La explicación anterior ilustra todo el flujo de trabajo que realiza la aplicación para atender todas las peticiones de los usuarios.

CONCLUSIONES

1. Los Bomberos Voluntarios del municipio de San José Pinula, Guatemala, actualmente informan a las personas de cómo utilizar la aplicación y los beneficios que se pueden obtener a partir de dicha aplicación, ya que los agentes encuentran con rapidez a las personas que están siendo afectadas.

2. Actualmente, los bomberos cuentan con conexión a internet y una computadora en la que utilizan el *dashboard* de la aplicación sin problemas.

3. La aplicación cuenta con características que tienen las aplicaciones estudiadas, como puede ser la llamada a los bomberos de una forma eficiente.

4. Según entrevistas con los Bomberos Voluntarios de San José Pinula, los agentes se sienten satisfechos con la aplicación, ya que les ayuda a localizar a las personas por medio de aplicaciones externas de forma rápida y efectiva.

RECOMENDACIONES

1. Leer las especificaciones de las conclusiones, dado que las recomendaciones poseen exactamente el mismo formato.

2. Tomar en cuenta, para el desarrollo de aplicaciones, la utilización de herramientas de código abierto, ya que pueden reducir costos al cliente.

3. Utilizar herramientas de versión de código para el desarrollo de aplicaciones y el manejo de cambios, mejoras y nuevas características que se quieran realizar sobre el programa.

4. Añadir funcionalidades que le dieron un valor agregado al programa y la realización de un análisis sobre aplicaciones similares.

5. Integrar aplicaciones de terceros puede ser útil para la rapidez del lanzamiento de la aplicación.

6. Tener al alcance los números de emergencia de los bomberos para reaccionar ante cualquier vicisitud.

REFERENCIAS

1. Amazon Web Services, (s.f.). *Amazon API Gateway Developer Guide*. Recuperado de https://cutt.ly/MH7GCoU.

2. Amazon Web Services, (s.f.). *AWS Lambda Developer Guide*. Recuperado de https://cutt.ly/qH7Hxhr.

3. Batres A. (2014). *Guatemala, un país con más celulares que habitantes.* Soy 502. Recuperado de https://cutt.ly/UH7CyCL.

4. Carly. (2014). *Before GPS and geocaching existed: Three Navigation Systems.* Geocaching Official Blog. Recuperado de https://www.geocaching.com/blog/2014/11/before-gps-andgeocaching-existed-three-navigation-systems/.

5. Central Android, (2015). *Android pre-history*. Blogspot. Recuperado de http://www.androidcentral.com/android-pre-history.

6. Docker docs, (s.f.). *Dockerfile reference.* Recuperado de https://docs.docker.com/engine/reference/builder/.

7. Greenberg, P. (s.f.). *21 Essential React.js Interview Questions.* Toptal. Recuperado de https://www.toptal.com/react/interview-questions.

8. Reacts. (s.f.). *Components and Props.* Facebook Open Source. Recuperado de https://reactjs.org/docs/components-and-props.html.

9. Valencia, H. (s.f.). *Documentación de arquitectura*. Software Guru. Recuperado de http://sg.com.mx/revista/30/documentacion-arquitectura#.V6FqAmgrJIe.