Arquitectura Tecnológica para la Asignación de Ambulancias en Ciudades Inteligentes

Technological Architecture for Ambulances Assignment in Smart Cities.

Luis Carlos Ospina Tobon[1]
luis.ospina03@usc.edu.co

Simena Dinas, PhD[2]
simena.dinas00@usc.edu.co

Universidad Santiago de Cali, Facultad de Ingeniería, Programa de Maestría en Informática (1)
Universidad Santiago de Cali, Facultad de Ingeniería, Programa de Maestría en Informática (2)

Resumen
La asignación de ambulancias para atender a pacientes involucrados en accidentes es uno de los mayores problemas en muchos países del tercer mundo. Esto es un problema debido principalmente a dos factores: movilidad y disponibilidad limitada de recursos. Tradicionalmente, el problema se aborda solicitando asistencia al paciente mediante la transmisión por radio a todos los operadores de ambulancias. El problema es que algunas ambulancias notificadas pueden no estar cerca del accidente, otras podrían estar de servicio en camino para atender otros accidentes, o no estar disponibles. Adicionalmente, se ha evidenciado que la transmisión genera competencia entre las ambulancias, o incluso falta de asistencia a pacientes cuando muchos operadores deciden atender al mismo accidente. Además, la competencia por los servicios en algunos momentos ha generado nuevos accidentes. En otros casos, se han visto impactos negativos desde las perspectivas económica, ética, social, política y de movilidad. Debido a esta situación, se diseñó una Arquitectura Tecnológica para la Asignación de Ambulancias en Ciudades Inteligentes, adicionalmente, se desarrolló un prototipo funcional intuitivo y fácil de usar, aplicando la metodología Design Thinking. Es una propuesta que pone la tecnología al servicio de la comunidad y tiene como objetivo mejorar la calidad de vida de los ciudadanos, como se propone en el manifiesto de la ciudad inteligente. Esta propuesta se realizó bajo el paradigma computacional en la nube, el cual implementa el front-end de una aplicación móvil para notificaciones de emergencias médicas en tiempo real. Además, el back-end gestiona la lógica empresarial para asignar ambulancias de forma inteligente teniendo en cuenta la disponibilidad y la georreferenciación. La aplicación fue posible mediante el uso de tecnologías como React JS, React Native, Firebase, Realtime Database, Cloud Functions, Google Maps y Directions API. La arquitectura del software tiene un nivel de madurez TRL 4 y cumple con los 10 principios de usabilidad de Jakob Nielsen, adicionalmente, es completamente funcional, escalable y adecuada, dando una solución a las necesidades descritas. Las pruebas fueron enfocadas en el registro de emergencia y la asignación de ambulancia, y en el 98% de los casos, el prototipo eligió la ambulancia más cercana, la respuesta fue inferior a 2 segundos y el sistema se comportó de acuerdo con los eventos necesarios para dar solución al problema.

Palabras Clave: ciudades inteligentes, aplicaciones móviles, ambulancias, arquitectura tecnológica, computación móvil, emergencias médicas, back-end, front-end.

Abstract
The ambulances assignment to attend patients involved in accidents is one of the biggest problems in many third-world countries. It is a problem due mainly to two factors: mobility and limited resource availability. Traditionally, the problem is addressed by requesting the patient assistance by using radio broadcasting to all ambulance operators. The problem is that some notified ambulances may not be close to the accident, others could be on duty, on their way to attend other accidents, or not available. Nonetheless, it has also been notices that broadcasting generates competition among ambulances, or even lack of assistance to patients when many operators decide to attend the same accident. Furthermore, the competition to assist, has sometime generated new accidents. In other cases, negative impacts have been seen from the economic, ethic, social, politic, and mobility perspectives. Because of this situation, A Technological Architecture for the Assignment of Ambulances in Smart Cities was designed. Additionally, an intuitive and easy-to-use functional prototype was developed, applying the Design Thinking methodology. It is a proposal that puts technology at the service of community and aims to improve citizen quality of life, as proposed in the smart city manifest. This proposal was made under the cloud computational paradigm, which deploy the front-end of a mobile application for medical emergency real-time notifications. Additionally, the back-end manages the business logic to assign ambulances in a intelligent way and taking into account availability and georeferencing. The application was made possible by using technologies such as React JS, React Native, Firebase, Realtime Database, Cloud Functions, Google Maps, and Directions API. The software architecture is

completely functional, scalable, and adequate. It has a TRL 4 maturity level, and complies with Jakob Nielsen's 10 usability principles, giving a solution to the outlined needs. Tests was focused on Emergency logging and ambulance assignment, and in 98% of the cases, the prototype chose the nearest ambulance, the response was lower than 2 seconds, and the system behaved according to the necessary events to give solution to the problem.

Keywords: smart cities, mobile applications, ambulances, technological architecture, mobile computation, medical emergency, back-end, front-end.

I. INTRODUCCIÓN

El concepto de ciudades inteligentes permite a los lectores pensar en ciudades futuristas, según Quijano et al. (2020), el creciente desarrollo de iniciativas de ciudades inteligentes ha sido motivado por el uso de soluciones ubicuas de computación y móviles, los avances recientes en infraestructuras tecnológicas, el desarrollo de sensores de bajo costo, la miniaturización de la electrónica y por los avances en tecnologías de comunicaciones inalámbrica. En particular, surgen por la necesidad de mejorar la calidad de vida de los usuarios haciendo uso de las tecnologías de la información y comunicaciones (EIP SCC, 2017). Las ciudades inteligentes están equipadas con diferentes dispositivos informáticos que van desde sensores integrados en objetos cotidianos hasta teléfonos inteligentes, recopilan información en tiempo real y están interconectados a través de las redes de comunicación, lo que les permite enviar y recibir datos (Batty et al. 2012).

Basado en lo anterior, el Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo - CYTED al cual pertenecen varios países de Iberoamerica, han aunado esfuerzos para crear el proyecto CITIES con el objetivo de impulsar la construcción de ciudades inteligentes totalmente integrales, eficientes y sostenibles a través de la cooperación y transferencia de conocimiento entre los grupos de investigación participantes. Como ejemplo, la participación de la Universidad Santiago de Cali con el proyecto CAMoN (Co-Creative Air Monitoring Network) que tuvo como objetivo el desarrollo de una red de monitorización de la calidad de aire combinando dispositivos de bajo coste y aplicaciones móviles (CAMoN, 2017). El proyecto integró crowdsensing, crowdsourcing, los beneficios de los datos de sensores junto con los informes de los ciudadanos, bajo el framework de experimentación europeo para ciudades inteligentes desarrollado por OrganCity (OrganCity, 2017). CAMoN proporciona información a los usuarios sobre calidad del aire, recomendaciones o históricos de contaminación.

Uno de los aspectos importantes de las ciudades inteligentes es la movilidad. Las administraciones de las ciudades desean un tráfico vehicular que avance sin inconveniente y que respete las normas de transito para evitar accidentes (Siegfried, 2018). Sin embargo, de acuerdo con el tipo de transporte las condiciones en las que se movilizan los vehículos pueden variar. Por ejemplo, es el caso de las ambulancias que tienen excepciones al transitar, avanzan cuando los semáforos están en rojo, invaden carriles exclusivos para transporte masivo, entre otras situaciones permitidas por Código Nacional de Tránsito Terrestre, artículo 64.

En Colombia, la asignación de ambulancias para atención de emergencias es uno de los problemas recurrentes según Franco y Mercado (2016). Los ciudadanos reportan las urgencias médicas al Sistema de Emergencias Médicas – SEM (Ministerio de Salud, 2017) y los operarios de este sistema trasmiten una difusión amplia para convocar a las ambulancias al siniestro. Esto como consecuencia genera competencias y desplazamientos innecesarios por parte de los operadores de las ambulancias. En este documento se planteará el diseño de una arquitectura tecnológica para la asignación de ambulancias en ciudades inteligentes. Este se desarrollará enmarcado en la metodología Design Thinking que permitirá implementar el proyecto bajo el enfoque de conocer exactamente la problemática de los interesados para dar una solución satisfactoria.

II. METODOLOGÍA

El desarrollo de este proyecto se hizo con la metodología Design Thinking. La metodología se enfoca en dar solución a problemas de innovación y diseño, por tanto, es útil para abordar problemas particulares de la industria. Design Thinking según Smith (2015) cuenta con seis etapas que son en su respectivo orden: Empatizar, Definir, Idear, Prototipar, Testear

e Implementar (ver figura 1) y tiene un comportamiento iterativo, esto quiere decir, que se puede volver a cada una de las etapas anteriores si se encuentran dificultades o mejoras específicas.

Figura 1: Fases de la Metodología Design Thinking

Adaptada de: https://www.aiontech.com.mx/design_thinking-en.html

Las etapas y la manera como se apropian en el proyecto de grado es la siguiente:

EMPATIZAR: En esta fase se inició con la comprensión profunda de la necesidad de los interesados, La idea fue conocer la problemática real desde la perspectiva del cliente. Para esta fase se uso como instrumento una entrevista que permitió entender las motivaciones, emociones y formas de pensar que llevan a los interesados a proponer la necesidad de la plataforma. También se usó la estrategia de "Observación encubierta" en donde se visualizó toda la dinámica de los operadores de ambulancias en el momento de atender las emergencias médicas. Esto permitió tener una visión más amplia de lo que se propuso para mejorar los tiempos y procedimientos de esta tarea.

DEFINIR: Durante la etapa de definición se determinaron los requerimientos claves que permitieron el diseño del proyecto. Posteriormente se realizó un documento de especificación de requerimiento SRS (IEEE, 2012) en donde se plasmaron los requerimientos funcionales y no funcionales que soportaría la arquitectura tecnológica. Una vez el documento fue definido, se validó con los interesados.

IDEAR: En esta etapa se hizo un estudio comparativo y una evaluación de las herramientas. El estudio permitió caracterizar las herramientas por rendimiento evaluándolas desde diferentes puntos de vista como lo son el rendimiento, escalabilidad, concurrencia, entre otras. Se definieron los motores de base de datos, conexiones, protocolos, tramas, infraestructura y otros aspectos a utilizar teniendo en cuenta los requerimientos funcionales y no funcionales de la arquitectura a implementar.

PROTOTIPAR: Se inició con la construcción de las diferentes alternativas de la arquitectura de acuerdo con lo investigado en la etapa de ideación, para esto se diseñaron los diagramas de despliegue y de componentes que dieron una idea mas concreta de los componentes de hardware y software que compondrán la plataforma. Estos diagramas definidos bajo el enfoque del modelo 4+1 (Kruchten, 1995). Finalmente, se plasmó la arquitectura tecnológica de acuerdo con los requerimientos funcionales y no funcionales del sistema.

TESTEAR (PROBAR): En esta etapa se probó el prototipo funcional desarrollado. Vincular a los interesados dio la posibilidad de obtener una retroalimentación que fue usada para hacer mejoras. Adicionalmente se realizó el seguimiento mediante las herramientas de analítica de datos que provee la plataforma Cloud Firebase permitiendo corroborar el buen comportamiento del prototipo funcional. Una vez el prototipo fue validado, se realizaron las conclusiones de la viabilidad de la arquitectura de acuerdo con las pruebas realizadas.

III. RESULTADOS Y DISCUSIÓN

De acuerdo con el estudio de las diferentes plataformas Cloud, se identificaron diferentes herramientas que funcionan como Plataforma como servicios (PaaS), plataformas que soportan el Backend de muchas de las aplicaciones móviles que hoy en día existen; Entre ellas se encuentran Cloud de Google, Azure de Microsoft, AWS de Amazon y Parse de Facebook (Marozzo, 2020). Sin embargo, un nuevo concepto llamado Backend como servicios (BaaS), ofrece herramientas orientadas al desarrollo del Backend de una aplicación viendo la infraestructura como una caja negra, pues los servicios son elásticos y permiten crecer y decrecer de acuerdo con la necesidad de la aplicación. Teniendo en cuenta lo anterior se seleccionaron dos plataformas: Firebase de Google y Parse de Facebook para integrarlas con las herramientas Frontend. Estas ofrecen tecnologías Cloud orientadas a la creación del Backend de una aplicación en la nube. En la tabla 1 se hace una caracterización de las dos plataformas seleccionadas.

Característica	Firebase	Parse
Propósito General	Desarrollo de aplicaciones web y móviles rápidas	Desarrollo de aplicaciones web y móviles rápidas
Hosting	Alojamiento en google, 100 conexiones gratis	Se puede alojar en cualquier hosting
Código Personalizado	Código no personalizado en la capa de google	Código totalmente personalizado
Base de datos	Soporta esquemas observables, introduce almacenamiento de archivos y manejo de seguridad	Buen manejo con base de datos relacionales
Push	Soporta notificaciones Push. Configuración ajustable	Soporte notificaciones Push
Configuración	Fácil configuración PaaS	Fácil configuración, pero se complica si se hace uso local
Almacenamiento	Almacenamiento en JSON con copias de seguridad que se pueden subir a Amazon Cloud o Google Cloud	No tiene restricción
Proveedor	Desarrollado por Google	Desarrollado por Facebook

Tabla 1: Caracterización de las plataformas Backend mas robustas

Adaptado de: https://www.quora.com/Should-I-move-my-app-from-Parse-+-Heroku-to-Firebase

Teniendo en cuenta la selección de Firebase por su trayectoria, bases de datos no relacionales y facilidad de implementación como plataforma Backend (Guido et al. 2020), se decidió buscar un Framework Frontend que tuviera alto acoplamiento con Firebase; para ello se inició una caracterización para identificar un Framework móvil que permitiera realizar un desarrollo híbrido y que tuviera todas las librerías disponibles para conectarse con el Backend. Como resultado del proceso de búsqueda, se obtuvieron los Frameworks que se describen en la tabla 2.

Característica	React Native	Xamarin	IONIC	Flutter
Lenguaje de programación	JavaScript + Swift, Objective-C, Android, Kotlin	C# con .NET	HTML5, CSS y JavaScript	Dart
Rendimiento	Parecido al nativo ★★★★	Xamarin IOS ANDROID ★★★★	Moderado ★★	Sorprendente ★★★★★
Interfaz de usuario	Usa controles nativos	Usa controles nativos	HTML CSS	Componentes propios, interfaz asombrosa
Mercado y comunidad	Muy fuerte y con una comunidad grande	Fuerte	Fuerte	No muy popular
Donde se puede usar	Todas las aplicaciones	Aplicaciones simples	Aplicaciones simples	Todas las aplicaciones
Reusabilidad de código	90% del código es reusable	96% del código es reusable	98% del código es reusable	50% - 90% del código es reusable
Proyectos exitosos	Facebook, Instagram, Airbnb, UberEats	Olo, The World Bank, Storyo	JustWatch, Pacifica, NationWide	HamiilTon
Precio	Código libre	Código libre con versión de pago	Código libre con versión de pago	Código libre

Tabla 2: Caracterización de los Frameworks Frontend más robustos

Adaptada de: https://www.apptunix.com/blog/frameworks-cross-platform-mobile-app-development/

Con los resultados arrojados en las anteriores caracterizaciones, sus funciones y sus posibles implementaciones, se propuso el diagrama de despliegue de la figura 2, el cual muestra la comunicación que hay entre los elementos del sistema.

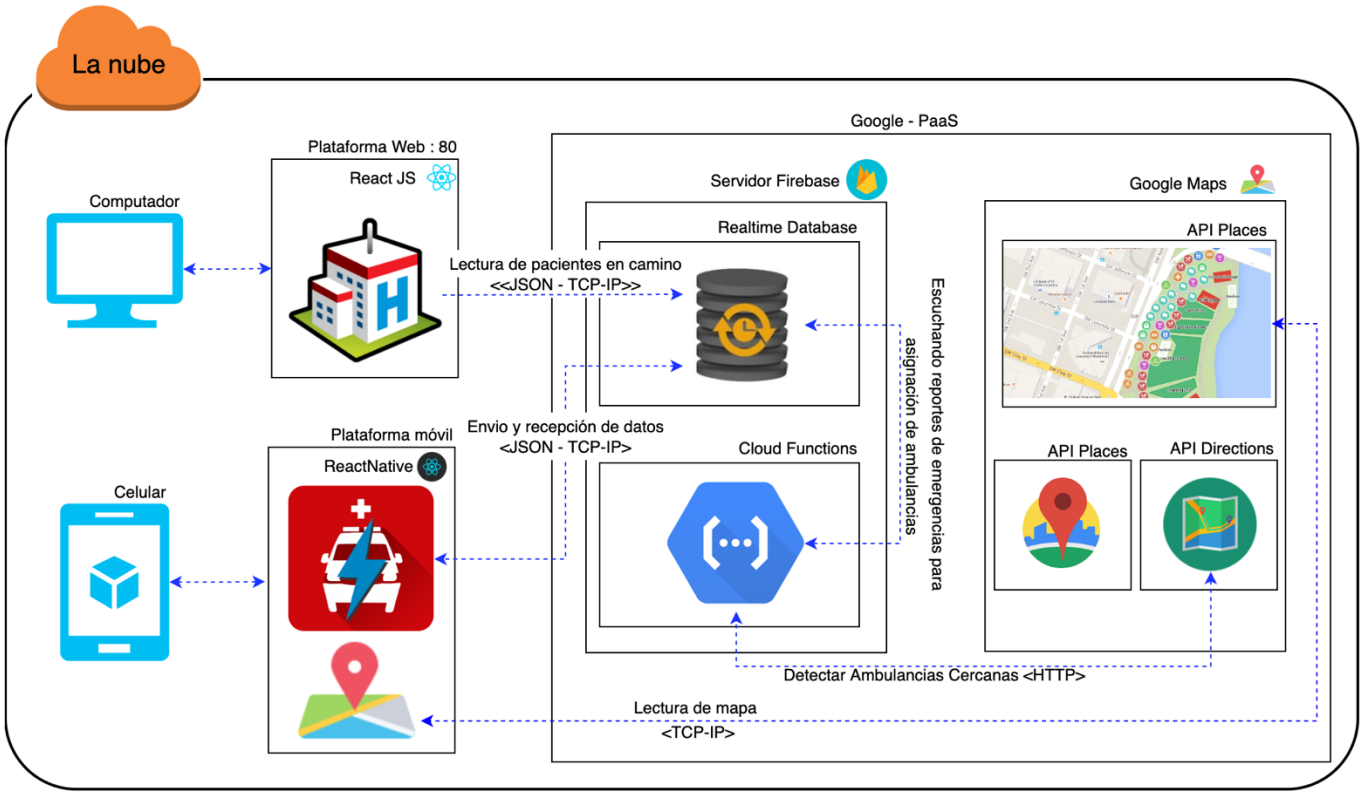


Figura 2: Arquitectura - Diagrama de despliegue
Fuente: Propia

En el diagrama de despliegue de la figura 2 se definen 3 subsistemas, el primer subsistema hace referencia a todo el Backend del proyecto diseñado bajo el paradigma PaaS de Google, en el se define el uso de dos plataformas, Firebase con las herramientas Realtime Database y Cloud Function y dentro de la plataforma de Google Maps el API Maps, API Directions (Oksana et al. 2020) y API Places.

Dentro de Firebase se usó Realtime Database como herramienta para la persistencia de los datos, el cual contiene un WebSocket que permite a los dispositivos móviles estar en escucha de los diferentes cambios que hay en la base de datos, esto posibilita que se puedan programar diferentes eventos para que los usuarios reciban notificaciones en tiempo real. Cloud Function se usa como plataforma para desplegar el algoritmo de búsqueda de ambulancia que se activa mediante un disparador que esta en escucha de los cambios sobre la colección Reportes de la base de datos. Una vez la emergencia es capturada, la función busca las ambulancias más cercanas gracias a diferentes algoritmos de geolocalización.

Dentro de la plataforma de Google Maps, el API Maps que permite desplegar el mapa de la ciudad. El API Directions despliega un servicio HTTP:80 que permite al algoritmo de búsqueda definir las distancias de las posibles ambulancias con la emergencia para ordenar las notificaciones priorizando el orden de llegada. El API Places que permite encontrar el hospital más cercano al cual deberá dirigirse la ambulancia, este ofrece una interfaz de comunicación que entrega los datos en formato JSON.

El segundo subsistema hace referencia a una aplicación móvil diseñada en React Native, Framework multiplataforma que permite el despliegue de aplicaciones móviles en diferentes sistemas operativos. Esta proporciona las interfaces gráficas a los operadores de ambulancias y reporteros de emergencia, toda la transaccionalidad se realizó con una comunicación bidireccional dispuesta por el WebSocket dispuesto por la Realtime Database (Ohyver et al. 2019).

El tercer subsistema está enfocado a una aplicación Web con React JS que le dá la posibilidad a los hospitales de obtener información de las emergencias. Este subsistema se desplegó mediante un servidor web Apache por el puerto 80 y se conectará a la base de datos Realtime Database mediante las APIS dispuestas por el proveedor de Firebase.

Una vez diseñado el diagrama de despliegue, se procedió a diseñar los componentes de software que se implementaron en los diferentes subsistemas (ver figura 3).

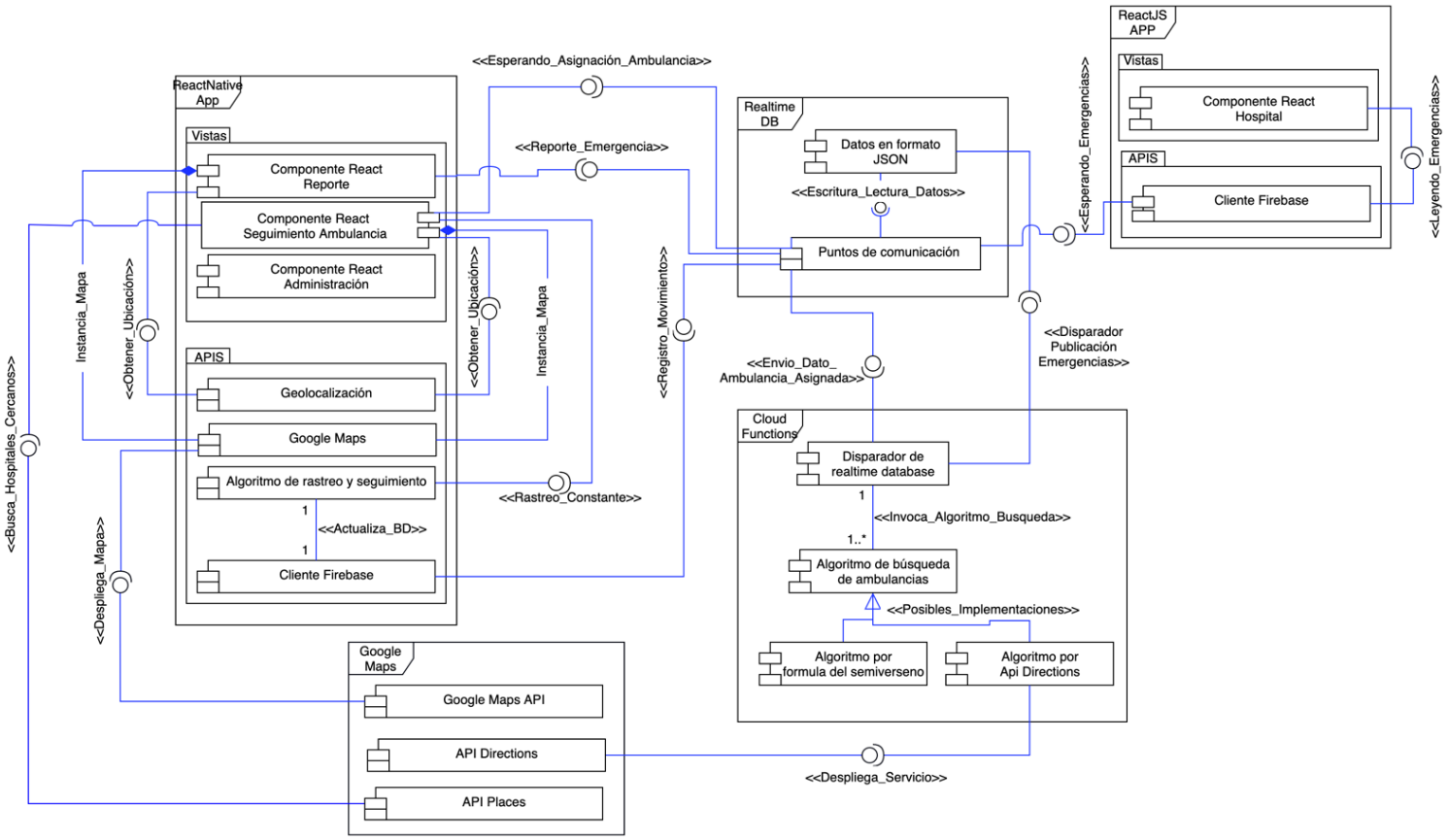


Figura 3: Arquitectura – Diagrama de componentes

Fuente: Propia

En el diagrama de componentes se puede visualizar más a fondo cada uno de los componentes de software que cumplen con una función fundamental dentro del sistema.

A nivel de Frontend se diseñaron dos plataformas, una con React Native debido a su buen rendimiento y desarrollo multiplataforma (Brito et al. 2019) que les permite a los operadores de las ambulancias y los reporteros de las emergencias. Esta plataforma está compuesta por dos grupos de componentes, uno a nivel de interfaces de usuario y el otro a nivel APIS para determinar la ubicación y para consumir los servicios de la Realtime Database. El paradigma computacional usado para estos componentes fue el de la orientación a objetos y el de la creación de componentes. Esta plataforma permite a los reporteros de emergencias determinar su ubicación por medio de la georreferenciación para que posteriormente se pueda enviar el reporte del siniestro. También ofrece las vistas del operador de ambulancia que le permitirá habilitarse dentro de la plataforma para que el Backend Firebase pueda notificarle los servicios disponibles. Incluye un subproceso que determina el movimiento de la ambulancia y actualiza la ubicación en la base de datos central. Una vez el operador toma el servicio, le indicará la mejor ruta para llegar al siniestro y posteriormente le indicará a que hospital debe llevar al paciente.

La otra plataforma es la de los Hospitales, fue desarrollada una aplicación Web con React JS que les permite a los hospitales identificar que pacientes van en camino al hospital y obtener una descripción del estado del paciente. Todo lo

anterior bajo la implementación de un Listener Realtime Database que detecta los siniestros en camino.

A nivel de Backend se manejan dos plataformas como servicios. La primera es la de Realtime Database, en ella se persisten los datos del sistema mediante el formato JSON como se ve en la figura 4.

Figura 4: Estructura de base de datos no relacional SIRSEM
Fuente: Propia

Al ser una base de datos no relacional, los datos se guardan en una colección JSON y permite la redundancia de datos para obtener información rápidamente. Se definieron las estructuras Usuarios, Reportes, Notificaciones. Con este modelo de datos se pueden determinar los usuarios por tipos, los reporteros de emergencias podrán notificar sus reportes de siniestros y el sistema Cloud Function estará atento para leer y generar nuevas notificaciones a los operadores de las ambulancias.

La segunda plataforma Backend hace referencia a Cloud Function de Google (Scheuner et al. 2020), en ella se diseño un algoritmo mediante la programación orientada a eventos. Se definió un disparador a nivel de base de datos que esta en escucha de los registros agregados en la colección Reportes. Una vez se detecta un nuevo reporte, la Cloud Function identifica las ambulancias más cercanas de acuerdo con sus ubicaciones usando la implementación de dos algoritmos de distancia. Lo anterior para determinar el orden de notificación de acuerdo con su distancia o tiempo (ver figura 5).

```
//Función para identificar los metros lineales de distancia entre 2 puntos.
getDistanceBySemiverseno(p1, p2) {
  rad = x => (x * Math.PI) / 180;
  var R = 6378137; //radio de la tierra en metros
  var dLat = rad(p2.latitude - p1.latitude);
  var dLong = rad(p2.longitude - p1.longitude);
  var a =
    Math.sin(dLat / 2) * Math.sin(dLat / 2) +
    Math.cos(rad(p1.latitude)) *
      Math.cos(rad(p2.latitude)) *
      Math.sin(dLong / 2) *
      Math.sin(dLong / 2);
  var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
  var meters = R * c;
  return meters;
}
```

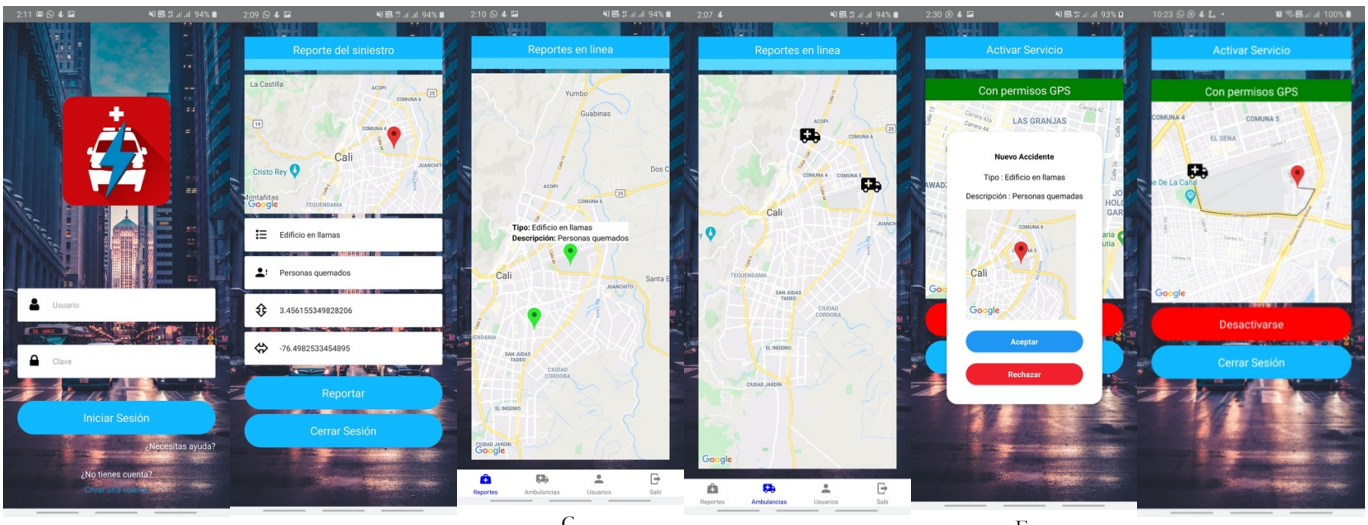
```
//Función para identificar los metros de distancia entre 2 puntos por Maps
async getDistanceByApiDirectionsMaps(p1, p2) {
  let APIKey = 'AIzaSy29F6tN9wlpAD2cw0SdrVeC-96qB0oThHU';
  let resp = await fetch(
    'https://maps.googleapis.com/maps/api/directions/json?origin=' +
      p1.latitude +
      ',' +
      p1.longitude +
      '&destination=' +
      p2.latitude +
      ',' +
      p2.longitude +
      '&key=' +
      APIKey,
  );
  let respJson = await resp.json();
  var meters = 0;
  if (respJson.status == 'OK') {
    var meters = respJson.routes.legs[0].distance.value;
  }
  return meters;
}
```

Figura 5: Algoritmos para el cálculo de las distancias

Fuente: Propia

Basado en lo que plantea Prakash et al. (2018), se usaron lo siguientes algoritmos; el primer algoritmo de búsqueda conocido como el algoritmo por formula del semiverseno (ver figura 5a), una ecuación para la navegación astronómica que calcula la distancia entre dos puntos de un globo sabiendo su longitud y su latitud, esta entrega la distancia lineal en metros entre la ambulancia y la emergencia. El segundo algoritmo con base al API Directions de Google Maps (ver figura 5b) que entrega la distancia real contemplando las calles habilitadas y el tiempo estimado de llegada. Una vez organizadas la ambulancia de acuerdo con su tiempo o distancia, el algoritmo procede a almacenar las notificaciones en la colección de la BD que a su vez gracias al sistema de Realtime Database, notifica de manera automática a los operadores sobre la solicitud.

Con la arquitectura definida y teniendo en cuenta que la metodología Design Thinking tiene como ultima fase el testeo, se implementó un prototipo funcional de la arquitectura que permite validar el correcto funcionamiento y la viabilidad de la arquitectura (ver figura 6).



A. Inicio de sesión

B. Reporte de emergencia

C. Generación de reportes

D. Identificación de ambulancias

E. Asignación de ambulancias

F. Sugerencia de la mejor ruta

Figura 6: Prototipo funcional usando la arquitectura definida

Fuente: Propia

En la figura 6 se puede identificar el funcionamiento del prototipo, en la figura 6a se evidencia el inicio de sesión del prototipo, validando los 3 posibles roles: reportero, operador y administrador. Cuando el reportero inicia sesión, el sistema le dá la posibilidad de seleccionar el punto del siniestro y le solicita algunos datos para generar el reporte (ver figura 6b). De inmediato el sistema obtiene el reporte y los grafica en la interfaz del administrador como se puede ver en la figura 6c. Posteriormente, se detectan las ambulancias más cercanas para generar una pila de las ambulancias a las cuales les podrá enviar la solicitud (ver figura 6d). El sistema envía las notificaciones a las ambulancias en orden de cercanía y cuando esta solicitud es aceptada por el operador de la ambulancia, recomienda la mejor ruta para llegar al lugar del siniestro, como se puede visualizar, en las figuras 6e y 6f, respectivamente.

El prototipo fue evaluado con los 10 principios de usabilidad de Jakob Nielsen (Nielsen, 1993), a continuación, se describen los criterios que se cumplen para el prototipo funcional:

- **Visibilidad del estado del sistema:** Es visible el estado del sistema mediante las herramientas de analítica de datos de Firebase.
- **Relación entre el sistema y mundo real:** La aplicación contempla el mismo lenguaje del mundo real y es de fácil uso.
- **Libertad de control y uso:** La aplicación permite que usuario pueda volver a las interfaces anteriores sin ningún problema
- **Consistencia y estándares:** Es consistente con los términos y estándares.
- **Prevención de errores:** Todos los algoritmos están encapsulados mediante estructuras de control para prevenir errores bloqueantes.
- **Reconocer antes que recordar:** La aplicación es intuitiva y no exige memorizar pasos.
- **Flexibilidad y eficiencia de uso:** Permite ser desplegada desde los diferentes dispositivos.
- **Diseño minimalista y estética:** El diseño es simple y no carga la vista de los usuarios.
- **Ayuda a los usuarios con errores:** La aplicación no exige el ingreso de mucha información, los campos están validados e indica a los usuarios los errores que esta cometiendo.
- **Ayuda y documentación:** La aplicación contiene alertas y opciones para que los usuarios puedan comprender como usarlo.

Finalmente, el prototipo funcional cumple con el nivel de madurez TRL 4, esto indica que la tecnología fue validada en laboratorio (Straub, 2015).

IV. CONCLUSIONES

Este artículo presenta una arquitectura tecnológica basada en el paradigma computacional de la nube para el reporte, asignación y seguimiento de ambulancias en la atención a emergencias médicas. Esta arquitectura se enfoca inicialmente en la disposición de servicios que permitan reportar las emergencias en el menor tiempo posible, teniendo en cuenta que el tiempo de notificación del accidente es el factor más importante. Posteriormente se ejecutan los algoritmos del API de Directions y del Semiverseno que permiten identificar las ambulancias más cercanas para realizar la asignación y evitar la competencia entre operadores de las ambulancias. Todo lo anterior se describe bajo una arquitectura completa que permite que las notificaciones de los eventos sean en tiempo real.

La arquitectura propuesta genera una alternativa efectiva, flexible, adaptable y rápida que el sistema convencional de difusión masiva que actualmente se utiliza. Proporciona información en tiempo real creando una mejor experiencia de usuario y atención cuando se reportan las emergencias. Como una propuesta para ciudades inteligentes se ampara en el

fortalecimiento de soluciones basadas en tecnología, minimizando el uso de los recursos, promoviendo formas eficientes de desarrollo sostenible, impactando positivamente la movilidad y generando bienestar y calidad de vida en los usuarios finales.

La definición de esta arquitectura esta directamente relacionado con la idea de crear ciudades inteligentes, ciudades que por medio del uso de la tecnológica puedan priorizar y atender las emergencias médicas de manera más eficiente y propiciar el uso adecuado de los recursos médicos que están disponibles para la ciudadanía.

Todo lo anterior basado en la metodología Design Thinking que permitió desarrollar el proyecto bajo un enfoque iterativo, entender la necesidad de las partes interesadas, desarrollar un prototipo funcional y dar respuesta de manera satisfactoria dando una posible solución a una problemática real.

En el futuro, las funcionalidades de la arquitectura podrán ampliarse debido a la utilización del paradigma PaaS, entre las mejoras se encuentra la posibilidad de que la plataforma pueda identificar de forma automática el hospital más idóneo de acuerdo con el tipo de afección que tenga el paciente. Además, se podrá realizar una aplicación móvil más completa basada en el prototipo funcional con la que se validó la arquitectura. Al ser soportada por una base de datos no relacional, se podrá incorporar en un futuro las técnicas de análisis de datos (minería de datos, aprendizaje automático, inteligencia artificial, entre otras.) que posibilitará la predicción de incidentes mejorando la atención de estas emergencias en cuanto a tiempo. Finalmente, se espera hacer la validación con usuarios finales, en condiciones que permita evaluar a fondo el prototipo construido.

REFERENCIAS

Batty, M. Axhausen, K,W. Giannotti, F. Pozdnoukhov, A. Bazzani, A. Wachowicz, M. Ouzounis, G & Portugali, Y.(2012). Smart cities of the future. The European Physical Journal Special Topics, 214(1), 481–518. https://doi.org/10.1140/epjst/e2012-01703-3

Brito, H. Santos, Á. Bernardino, J. Gomes, A. (2019). Mobile development in Swift, Java and React Native: An experimental evaluation in audioguides. berian Conference on Information Systems and Technologies, CISTI. Volume 2019-June, June 2019, Article number 8760864. https://doi.org/10.23919/CISTI.2019.8760864

CAMoN. (2017-2018). COMBA – USC. Co-Creative Air Monitoring Network. Recuperado de: http://comba.usc.edu.co/index.php/investigacion/15-proyectos/136-camon

EIP SCC. (2017). Ciudades Inteligentes Inclusivas: Un Manifiesto Europeo De Compromiso Ciudadano. Europa. Recuperado de: https://eu-smartcities.eu/sites/default/files/2017-09/Manifesto%20on%20Citizen%20Engagement_Spanish%20translation.pdf

Equipo de alta consejeria distrital de TIC. Bogotá, Ciudad Inteligente. (2018). Descargado de: https://bogota.gov.co/sites/default/files/inline-files/doc_smartcity.pdf

Franco, S (2016). Las ambulancias un negocio desalmado. El espectador. Recuperado de: https://www.elespectador.com/opinion/opinion/las-ambulancias-un-negocio-desalmado-columna-646961

Guido, A. Fikru, G. Waqar, H. (2020). On the performance of web services, google cloud messaging and firebase cloud messaging. Digital Communications and Networks Volume 6, Issue 1, February 2020, Pages 31-37. https://doi.org/10.1016/j.dcan.2019.02.002

IEEE. (2012). A new traceable software requirements specification based on IEEE 830. 2012 International Conference on Computer Systems and Industrial Informatics. https://www.doi.org/10.1109/ICCSII.2012.6454481

Kruchten, P. (1995). The 4+1 View Model of Architecture. IEEE Software, Volumen 12, Issue: 6, Pag 42-50. https://www.doi.org/10.1109/52.469759

Marozzo, F. (2019). Infrastructures for High-Performance Computing: Cloud Infrastructures. Encyclopedia of Bioinformatics and Computational Biology, Volume 1, 2019, Pages 240-246. https://doi.org/10.1016/B978-0-12-809633-8.20374-9

Mercado, J, G. (2016). Paseo de la muerte. El espectador. Recuperado de: https://www.elespectador.com/noticias/nacional/asi-el-paseo-de-muerte-de-ambulancias-articulo-646338

Ministerio de salud. (2017). Sistema de emergencias medicas – SEM. Recuperado de : https://www.minsalud.gov.co/salud/PServicios/Paginas/Sistema-de-emergencias-medicas-SEM.aspx

Nielsen, J. (1993). Usability engineering. Boston: Academic Press. ISBN: 0-12-518406-9. https://doi.org/10.1016/C2009-0-21512-1

Ohyver, M. Jurike, V. Sungkawa, I. Bonifasius, E. Argus, I. (2019). The Comparison Firebase Realtime Database and MySQL Database Performance using Wilcoxon Signed-Rank Test. Procedia Computer Science, Volume 157, 2019, Pages 396-405. https://doi.org/10.1016/j.procs.2019.08.231

Oksana, L. Ihor, T. Pavlo, L. (2020). Navigation Assistive Application for the Visually Impaired People. 11th IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT 2020. https://www.doi.org/10.1109/DESSERT50317.2020.9125013

OrganCity. (2017). Co-creating digital solutions to city challenges OrganiCity. Recuperado de: https://organicity.eu/

Prakash, M. Nithyanantham, S. Nishanth, V. Prakash, A. Kaviyarrasu, D. (2018). Smart city ambulance for tracking shortest path using global position system. International Journal of Engineering & Technology. Pag 187-190. http://dx.doi.org/10.14419/ijet.v7i1.3.10668

Quijano, Lara. Cantado, Ivan. Cortes, Maria. Gil, Olga. (2020). Recommender systems for smart cities. Information Systems, Volume 92, September 2020. https://doi.org/10.1016/j.is.2020.101545

Scheuner, J. Leitner, P. (2020). Function-as-a-Service performance evaluation: A multivocal literature review. Journal of Systems and Software Volume 170, December 2020, Article number 110708. https://doi.org/10.1016/j.jss.2020.110708

Siegfried, R,W. (2018). The governance of smart cities: A systematic literature review. Cities. Pag 1-23. https://doi.org/10.1016/j.cities.2018.02.014

Smith, C, S. Iversen, O, S. HJorth, M. (2015). Design thinking for digital fabrication in education. International Journal of Child-Computer Interaction. Volume 5, September 2015, Pages 20-28. https://doi.org/10.1016/j.ijcci.2015.10.002

Straub, J. (2015). In search of technology readiness level (TRL) 10. Aerospace Science and Technology. Volume 46, October–November 2015, Pages 312-320. https://doi.org/10.1016/j.ast.2015.07.007