Desarrollo de un Gateway Server para decodificar y retransmitir mensajes del sistema de alerta temprana en servicios de televisión digital terrestre

Salas Ladino, José Alejandro

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Ing. Olmedo Cifuentes, Gonzalo Fernando, Ph.D.

8 de febrero del 2023

17/2/23 12:36

Informe de originalidad

NOMBRE DEL CURSO
Revisión de Tesis

NOMBRE DEL ALUMNO
Alejandro Salas

NOMBRE DEL ARCHIVO
Alejandro Salas - Documento sin título

SE HA CREADO EL INFORME
16 feb 2023

Resumen

Fragmentos marcados	5	0,7 %
Fragmentos citados o entrecomillados	2	0,5 %

Coincidencias de la Web

kinsta.com	1	0,3 %
apkcombo.com	1	0,2 %
ionos.es	1	0,2 %
openwebinars.net	1	0,2 %
academia-lab.com	1	0,1 %
ciberninjas.com	1	0,1 %
udemy.com	1	0,1 %

1 de 7 fragmentos

Fragmento del alumno MARCADO

PythonPython es un lenguaje de programación de propósito general y **alto nivel**. **Su filosofía de diseño enfatiza la legibilidad del código con el uso de sangría significativa**

Mejor coincidencia en la Web

Python programming tutorial app **Python es un lenguaje de programación de propósito general** de **alto nivel** interpretado. **Su filosofía de diseño enfatiza la legibilidad del código con el uso de** una ...

Python Programming Tutorial APK (Android App) - Descarga Gratis https://apkcombo.com/es/python-

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Certificación

Certifico que el trabajo de titulación, "**Desarrollo de un Gateway Server para decodificar y retransmitir mensajes del sistema de alerta temprana en servicios de televisión digital terrestre**" fue realizado por el señor **Salas Ladino José Alejandro** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 8 de febrero del 2023

Firma:

Firmado electrónicamente por:
GONZALO FERNANDO
OLMEDO CIFUENTES

.....

Ing. Olmedo Cifuentes, Gonzalo Fernando, Ph.D.

C. C. 171169634-2

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Responsabilidad de Autoría

Yo, **Salas Ladino, José Alejandro** con cédula de ciudadanía n° 0250044179, declaro que el contenido, ideas y criterios del trabajo de titulación: **"Desarrollo de un Gateway Server para decodificar y retransmitir mensajes del sistema de alerta temprana en servicios de televisión digital terrestre"** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 8 de febrero del 2023

.....

Salas Ladino, José Alejandro

C.C.: 025004417-9

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Autorización de Publicación

Yo **Salas Ladino, José Alejando,** con cédula de ciudadanía n° 0250044179, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Título: "Desarrollo de un Gateway Server para decodificar y retransmitir mensajes del sistema de alerta temprana en servicios de televisión digital terrestre"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Sangolquí, 8 de febrero del 2023

.....

Salas Ladino, José Alejandro

C.C.: 025004417-9

Dedicatoria

Este trabajo de investigación está dedicado a mis abuelos y a mi madre, por confiar en mí, por su apoyo incondicional

Dedicación especial a mi hija Danna Salas por darme fuerzas y ganas de salir adelante.

José Alejandro Salas Ladino

Agradecimientos

Agradezco a mi madre que me ha apoyado y ayudado durante toda mi etapa universitaria.

Agradezco a mis abuelos maternos que siempre se preocuparon por mí, y fueron un pilar importante para que pueda terminar mi educación universitaria.

Agradezco a mi esposa que me acompañó y me brindo su apoyo durante toda la carrera universitaria.

Agradezco a mi hermana y a mi padre que por su ayuda sin ellos nada de esto sería posible.

Agradezco al Ing. Andy Muñoz que fue un amigo incondicional durante la mayor parte de la carrera.

Finalmente, quiero expresar mi gratitud a mi tutor Ing. Gonzalo Olmedo Ph.D en este trabajo de investigación, por sus conocimientos, enseñanzas y especialmente por su forma de llegar y tratar a las personas que tenemos la fortuna de conocerlo.

Índice de contenido

Índice de tablas

Índice de figuras

Resumen

El presente trabajo de investigación contiene el análisis, diseño, desarrollo e implementación de un Gateway Server para decodificar y retransmitir mensaje de alerta temprana en servicios de televisión digital terrestre, es importante mencionar que Ecuador se encuentra en una zona de riesgo propensa a sufrir algún desastre natural, el Sistema de Alerta Temprana usando EWBS (en inglés, *Emergency Warning Broadcasting System*) es de gran ayuda para notificar a la población pero debido a su baja popularidad y condiciones, con el desarrollo de un servidor de Gateway garantizamos la notificación a los usuarios o clientes de las diferentes aplicaciones móvil y de escritorio.

El sistema fue desarrollado para ser multiplataforma (Escritorio y Móvil) usando servicios propios de iOS y Android, para las notificaciones. Además de un servidor público alojado en Digital Ocean, plataforma amigable que provee servicios accesibles de *Cloud Computing* para gestionar la lógica de usuarios, tokens, identificadores de dispositivos (en inglés, *Device ID*), y mensajes de alerta. Mediante la orquestación de todos estos servicios se obtiene un Gateway Server que lee el mensaje de alerta temprana y la gestiona mediante APNs y Google FCM para clientes móviles y mediante consulta a un API del servidor para clientes de escritorio. En la actualidad mucha información importante se maneja mediante notificaciones en el celular o computador y gran cantidad de usuarios siempre están alertas a un nuevo mensaje en sus dispositivos.

Palabras clave: gateway server, computación en la nube, multiplataforma, identificador de dispositivo.

Abstract

This research work contains the analysis, design, development and implementation of a Gateway Server to decode and retransmit early warning message in digital terrestrial television services, it is important to mention that Ecuador is in a risk zone prone to suffer any natural disaster, the Early Warning System using EWBS (Emergency Warning Broadcasting System) is of great help to notify the population but due to its low popularity and conditions, with the development of a Gateway server we guarantee the notification to users or customers of the different mobile and desktop applications.

The system was developed to be multiplatform (Desktop and Mobile) using iOS and Android own services, for notifications. In addition to a public server hosted on Digital Ocean, a friendly platform that provides accessible Cloud Computing services to manage user logic, tokens, device identifiers (Device ID), and alert messages. By orchestrating all these services, we obtain Gateway Server that reads the early warning message and manages it through APNs and Google FCM for mobile clients and by querying a server API for desktop clients. Nowadays a lot of important information is handled through notifications on the cell phone or computer and a large number of users are always alert to a new message on their devices.

Key Words: gateway server, cloud computing, multiplataform, device id.

Capítulo I

Introducción

Antecedentes

En septiembre de 1983 un gran terremoto con epicentro en Tokio acabó con la vida de 100.000 personas, por esta razón se conmemora el primero de septiembre como el día de prevención de desastres en Japón. El estudio del sistema de difusión de alerta de emergencia EWBS (en inglés, *Emergency Warning Broadcasting System*) empezó en 1980 y en septiembre de 1985 inició su implementación en Japón. La primera operación fue en una alerta de tsunami en marzo de 1987. Desde su primera operación, el sistema de alerta de emergencia ha sido utilizado 14 veces más durante 20 años. (Shogen et al., 2006).

Debido a desastres naturales como lo ocurrido en abril 2016 en Ecuador, llegando a ser uno de los sismos más críticos, registrando una magnitud 7.8 en la escala de Richter, referente al número de pérdidas humanas, se reportó 655 fallecidos y 48 desaparecidos, sin contar miles de heridos.

En trabajos similares como el de Sambrano y Acosta (2019), se ha buscado que la alerta y la ayuda llegue lo más pronto posible alcanzando el mayor público posible, por consiguiente, se ha tratado de expandir la red de sintonía mediante la difusión de la alerta temprana por la red de telefonía IP de la Universidad de las Fuerzas Armadas – ESPE.

La televisión digital terrestre (TDT) en el Ecuador se viene implementando desde el año 2012, el gobierno de turno dictaminó un plan maestro para que transición de la televisión convencional análoga a la televisión digital con un plazo hasta el año 2018, en el cual las entidades que brinden el servicio de televisión adopten la nueva tecnología que está en concordancia con el estándar ISDB-Tb (en inglés, *International Integrated Services Digital Broadcasting – Terrestrial*). (Flores et. al., 2018).

La Universidad de las Fuerzas Armadas ESPE ha contribuido con múltiples trabajos de investigación como en el 2011 se implementa un transmisor de televisión digital terrestre

(Villamarín et. al., 2011) y en el 2021 se implementó un receptor de señal de emergencia EWBS utilizando un SDR (Castillo y Olmedo, 2021), proyectos que anteceden a este trabajo de titulación, buscando lograr comunicar el mensaje de alerta temprana a la mayor cantidad de personas por múltiples dispositivos terminales.

Justificación e importancia

Los sistemas basados en alerta temprana en servicios de televisión digital terrestre han sido objeto de estudio en el área de televisión digital de la Universidad de las Fuerzas Armadas ESPE como por ejemplo el desarrollo de varias herramientas de generación de flujo de transporte “TS” (en inglés, *Transport Stream*) por parte del grupo de investigación *WiCOM* del laboratorio de televisión digital de la ESPE (Olmedo et. al., 2015), incluyendo la configuración de códigos de área y alarmas sonoras dentro del TS (Segura et. al., 2016), la incorporación del mensaje de emergencia a través de sobreimposición de texto (Recalde y Olmedo, 2016), entre otros.

El Instituto Nacional de Estadística y Censo INEC de acuerdo al censo realizado en 2019 en lo que se refiere a tecnologías de la información y comunicación, estima que el 55,3% de la población ecuatoriana tiene acceso a un teléfono móvil celular activado, el 42,4% su teléfono móvil es un smartphone. Con respecto al acceso a Internet, el 45,5% de la población ecuatoriana tiene acceso a Internet (INEC, 2019), por lo cual el desarrollo de una aplicación móvil y la implementación de una aplicación móvil es viable debido a que al menos a una de cada dos personas que cuentan con su teléfono móvil activado será informada del fenómeno natural para prevenir daños y complicaciones mientras está sucediendo un desastre.

La información y comunicación a la mitad de un fenómeno natural es importante debido a que se puede salvar vidas, así como también alertar de forma oportuna y rápida a la ciudadanía afectada y a otros sectores sobre la afectación que acontezca en cualquier parte de territorio nacional.

El Ecuador en el año 2022 ha incrementado los usuarios que utilizan el Internet a 13,6 millones que equivale al 75,6% y el porcentaje que acceden al Internet desde su teléfono móvil es 98,1% lo que nos da un claro indicador de que tres de cada cuatro ciudadanos podrían llegar a ser notificados con la alerta en un tiempo prudente. (Medina, 2022).

El sistema de mensajes EWBS es transmitido por un servidor de broadcast (canal de televisión). La transmisión depende de él o de los canales que envían la señal, debido a esto se requiere un sistema complejo de transmisión y no todas las personas podrán recibir la señal ya que dependen del canal que están sintonizando.

El presente trabajo pretende ayudar a las personas en el caso de suscitarse un fenómeno natural con la implementación de un sistema 24/7 de alerta para desastres naturales mediante un Gateway Server con el cual se puede tener informada a la ciudadanía de la situación en el momento que se está produciendo el fenómeno, esta información es importante ya que alerta en situaciones de emergencia ayudando a disminuir el número de víctimas que podría llegarse a tener y así prevenir daños de mayor magnitud.

El SAT (Sistema de Alerta de Temprana), es una tecnología que monitorea 24/7 y alerta oportunamente en el caso de que un desastre natural se esté suscitando; la interacción con este sistema está a cargo del Servicio Integrado de Seguridad ECU 911, el Instituto Geofísico de la Escuela Politécnica Nacional, Instituto Nacional de Meteorología e Hidrología, Instituto Oceanográfico de la Armada y la Secretaría de Gestión de Riesgos. (Ecu911, 2018).

En general, un servicio digital consta de múltiples componentes como audio, video, texto y gráficos, cada uno en una representación comprimida. Para la compresión de audio, por ejemplo, se aplica el estándar MPEG-1 y para la compresión de video el estándar MPEG-2. El texto y los gráficos se pueden transmitir para diversos fines; los ejemplos son los subtítulos, periódicos electrónicos, compras desde el hogar y otras aplicaciones interactivas. Dependiendo del nivel aplicado de integración, la decodificación de video, audio, texto y gráficos se puede implementar en un dispositivo (Meer, 1995). En lo que se refiere a televisión digital el objeto de

estudio es el flujo de transporte TS (en inglés, *Transport Stream*), protocolo de comunicación de los estándares MPEG-2, donde se multiplexan los flujos o streams de video, audio y texto codificados. (Recalde y Olmedo, 2016).

Un Gateway Server es un conjunto de definiciones y protocolos programados que se utiliza para la integración de sistemas sin importar como están construidos usualmente el protocolo de comunicación es HTTP combinado con JSON, las interfaz de programación de aplicaciones API permiten que su información y servicios se comuniquen con otros, actualmente existe dos brechas importantes en el desarrollo las interfaces (front-end) y la lógica de negocio (back-end), con esto se logra comunicar un sistema de escritorio, una aplicación móvil, una aplicación Web y cualquier otra aplicación o tecnología que soporte peticiones HTTP esto otorga flexibilidad al diseño e implementación de soluciones. (RedHat, 2016).

Alcance del Proyecto

El presente trabajo de investigación da como resultado un servidor privado virtual VPS (en inglés, *Virtual Private Server*), un script en Python utilizado para leer el mensaje de alerta en el receptor y dos aplicaciones multiplataforma una para móvil iOS, Android y otros computadores de escritorio y/o laptops que tenga el sistema operativo Windows o Linux.

El servidor alojado en Digital Ocean contiene la aplicación Web desarrollada en Python con el framework Django Rest Framework, que mediante los resultados analizados puede servir a satisfactoriamente a 500 usuarios de escritorio en simultáneo y lotes ilimitados de 1000 clientes móviles garantizando que cada cliente de escritorio o móvil registrado reciba el mensaje, inclusive después de la alarma, si no tenía conexión a Internet en ese momento.

Las aplicaciones móviles se desarrollaron en React Native con Expo Cli y para la aplicación de escritorio se utilizó Java.

Objetivos

Objetivo General

Desarrollar un Gateway Server para decodificar y retransmitir mensajes del sistema de alerta temprana en servicios de televisión digital terrestre.

Objetivos Específicos

- Desarrollar el estado del arte sobre el protocolo del sistema de mensajes de alerta temprana y Gateway server.
- Desarrollar un Gateway server que sirva como medio de autenticación y almacene mensajes e identificadores de dispositivo (Device ID).
- Desarrollar dos aplicaciones para clientes móviles y de escritorio para retransmitir el mensaje de alerta temprana EWBS.
- Realizar pruebas de verificación del correcto funcionamiento del sistema.

Metodología

Las metodologías que permitirán cumplir con los objetivos planteados serán:

Teórica de tipo cualitativa

Enfocada en la recopilación de información y desarrollo del estado del arte sobre la decodificación del mensaje de alerta y la implementación de un servicio de Gateway Server que pueda ser consumido por aplicaciones iOS, Android y ordenadores, teniendo como fuente de consulta artículos científicos, revistas y repositorios de IEEE y Scopus.

Aplicada

Uso de material desarrollado para EWBS por investigadores de la Universidad de las Fuerzas Armadas ESPE y herramientas de programación como son los lenguajes Python, Java, JavaScript que en conjunto cumplirán funciones de decodificación, almacenamiento y envío de notificaciones a los dispositivos móviles de los mensajes de alerta.

Transversal

Recopilación del comportamiento del sistema de alerta, análisis de resultados, validación de la recepción de los mensajes de alertas los clientes finales.

Capítulo II

Fundamentos Teóricos

Introducción

El capítulo abarca conceptos, estándares, definiciones que utiliza la televisión digital, el sistema de radiodifusión de alerta de emergencia *EWBS*, y hace hincapié o está más enfocado en las herramientas de programación utilizadas para el despliegue y construcción del servidor Web, el diseño y desarrollo de las aplicaciones de escritorio y móviles utilizados en el mundo moderno abarcando temas de desarrollo con un lenguaje robusto como *Java* de fuerte tipado, fáciles de codificar como *Python* y relativamente nuevos que facilitan la creación de interfaces de aplicativos como lo es *React Native* con *JavaScript.*

Estándar ISDB-Tb

El estándar de Radiodifusión Digital de Servicios Integrados *ISDB* (en inglés, *Integrated Services Digital Broadcastring*), fue desarrollado en Japón para transmitir señal fija y móvil (portátiles y celulares), los estudios iniciales se remontan al año 1960 cuando la NHK STRL (Laboratorio de Investigaciones de la Asociación de Radiodifusión de Japón), fundó lo que hoy se conoce como la televisión de alta definición HDTV (en inglés, *High Definition Television*), fue hasta 1999 que se oficializó el estándar ISDB-T para transmisiones de televisión digital por último en 2003 Japón empezó a emitir señales de TV digital terrestre. (Fuentes et. al., 2012).

Brasil en el 2008 fue el país que modificó y adoptó el modelo de Japón, entre las más destacables modificaciones es utilizar como compresor de datos el *codec* de video MPEG-2, de la mano de Japón y Brasil nace ISDB-Tb como estándar de radiodifusión digital para televisión digital terrestre a continuación se mostrará el funcionamiento de la transmisión (ver Figura 1). (Oñate et. al., 2018).

Figura 1

Sistema General de transmisión ISDB-Tb

Nota. La figura muestra el diagrama de bloques de un Sistema de transmisión ISDB-Tb. Tomado de *Análisis del Transport Stream para el estándar de televisión digital ISDB-TB* (p. 29)*,* por Granja y Olmedo, 2011.

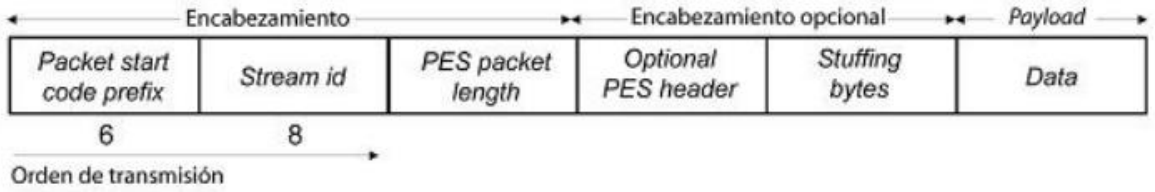
Transport Stream - TS

El flujo de transporte TS (en inglés, *Transport stream*), (ver Figura 4), es un conjunto de bits que contiene información necesaria para el sincronismo, corrección de errores, control y una carga o payload de la transmisión estos pueden ser audio, video y datos (Granja y Olmedo, 2011), está dividido en:

- PES: Consta de 188 bytes conformados por 4 bytes de cabecera y 184 bytes de datos.

Figura 2

Estructura del paquete PES

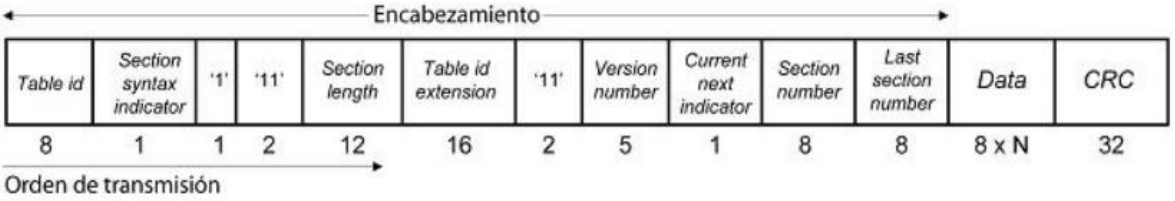


Nota. La figura muestra la estructura dentro de un paquete PES utilizado en el flujo de transporte TS. Tomado de *Análisis del Transport Stream para el estándar de televisión digital ISDB-TB* (p. 84)*,* por Granja y Olmedo, 2011.

- Sección: De tamaño variable está conformada por una cabecera, un payload y por código de detección de errores como se puede observar en la Figura 3.

Figura 3

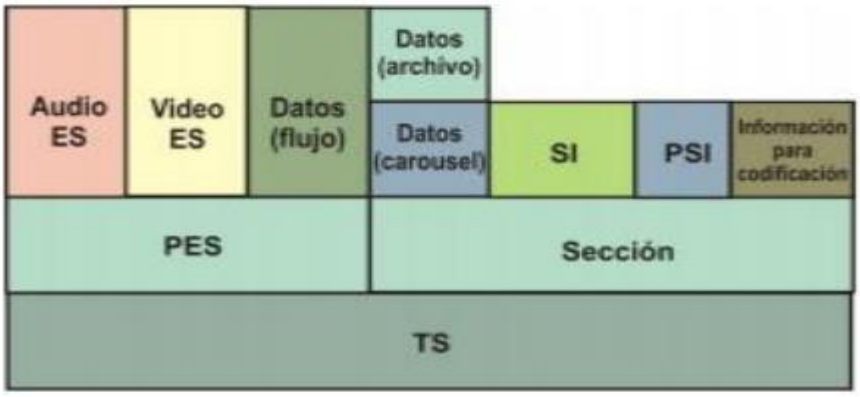
Estructura de una sección



Nota. El gráfico muestra la estructura de una sección de un flujo de transporte TS. Tomado de *Análisis del Transport Stream para el estándar de televisión digital ISDB-TB* (p. 82)*,* por Granja y Olmedo, 2011.

Figura 4

Modelos de Capas para la formación del TS



Nota. La figura muestra la estructura de un flujo de transporte. Tomado de *Análisis del Transport Stream para el estándar de televisión digital ISDB-TB* (p. 95)*,* por Granja y Olmedo, 2011.

Radio definido por software (SDR)

El término *SDR* engloba tener un dispositivo que emula un sistema de telecomunicaciones completo como lo es el transmisor, receptor o ambos en una tarjeta depurada lista para ser utilizada, a disposición de las necesidades del usuario tomando en

cuenta aspectos de hardware como el rango de frecuencia de trabajo y la capacidad de cómputo. (Castillo y Olmedo, 2021).

Dicha tarjeta de desarrollo puede implementar complejos algoritmos en varios lenguajes de programación como C, C++, C# y Python, entre los estándares inalámbricos los *SDRs*, pueden implementar ISDB-T para televisión, AM y FM para radio, Wifi, LTE y GSM para telefonía móvil entre otros.

En capítulos posteriores para levantar el escenario de pruebas se utilizó un *SDR Adalm Pluto* (ver Figura 5), controlado por GNU Radio (software de procesamiento digital de señales), como receptor de EWBS de bajo costo; implementado por la empresa Analog Devices, este es un dispositivo accesible de bajo costo en orden los de cientos de dólares con respecto a otros del mercado que trabaja en frecuencias desde los 325 MHz hasta los 3800 MHz. (Analog Devices, 2017) .

Figura 5

SDR Adalm Pluto

Nota. La figura muestra un SDR Adlam Pluto. Tomado de *Adalm – Pluto SDR Active Learning Module* (p. 2)*,* por Analog Devices, 2017.

Receptor y Sistema de radiodifusión de alerta de emergencia (EWBS)

El sistema *EWBS* implementado por el estándar ISDB-T en señales digitales y/o analógicas que tienen como característica en los datos su flujo de transporte TS enviar el mensaje de alerta codificado para informar a la población sobre algún fenómeno natural que esté pronto a suscitarse como los Tsunamis o estén provocando daños en el momento, según códigos de área que distinguen una zona de otra, aportando a la población afectada manteniéndolos preparados para cualquier eventualidad, la información en medio de un desastre es vital, se puede informar del fenómeno, así como cualquier otro tipo de mensaje de auxilio que se quiere difundir en general durante dicha eventualidad, como sucedió en Japón el 2006. (Shogen et. al., 2006).

Con esta tecnología se puede estar alerta ante cualquier desastre, siempre y cuando el receptor sea compatible con *EWBS*. En Japón existe infraestructura especializada para este tipo de eventualidades como torres, alarmas, inclusive radios y celulares soportaban *EWBS.* (Shogen et. al., 2006). En Ecuador es obligatorio este tipo de tecnología, dejando abierta la posibilidad de implementación a escala nacional.

Existen receptores comerciales de *EWBS* que tienen alcance limitado a una red de Intranet. A través del diseño de un receptor sobre SDR, se puede proponer un prototipo para detectar y recibir la señal de *EWBS*, además de decodificar el mensaje de alerta temprana, una vez decodificado y almacenado es posible con el uso de herramientas de programación como lo son los API, para interconectar sistemas difundiendo el mensaje de forma más eficiente por Internet a la población mediante una aplicación móvil y una aplicación de escritorio con alcance global. (Castillo y Olmedo, 2021).

Ubuntu 20.04

Sistema operativo con distribución Linux *open source* para equipos de escritorio y servidores, basado en Debian, popular como servidor en aplicaciones Web y computación en la

nube en la actualidad, por su gran cantidad de usuarios activos y comunidad de soporte y actualizaciones.

Esta versión de *Ubuntu* presenta mejoras en cuanto a seguridad y rendimiento, mejor rendimiento a nivel de ahorro de energía, así como también un tiempo de arranque menor.

A medida que salen versiones mejoradas de *Ubuntu*, presentan novedosas mejoras y esta no es la excepción, ya que como anteriormente se mencionó a nivel de rendimiento, esta versión hace más sencillos los despliegues de inteligencia artificial gracias a NVIDIA CUDA, esto representa mejores en el soporte y consumo de memoria al hablar del hardware. (Stackscale, 2020).

En cuanto a lo que tiene que ver con seguridad, se han adicionado más características y a la vez se han mejorado las existentes, tales como: medidas de autoprotección del Kernel, protección frente a stack-clash, integridad del flujo de control, autenticación universal sin contraseña y multifactor con Fast ID Online(FIDO), para protegerse de los ataques de ingeniería social y un arranque seguro para protegerse de ataques de bajo nivel y rootkits. (Stackscale, 2020).

En el presente trabajo se utilizó Ubuntu 20.04 LTS como sistema operativo de la máquina virtual alojada en la nube para el despliegue del servidor de aplicación Web.

Python

Python es un lenguaje de programación de propósito general y alto nivel. Su filosofía de diseño enfatiza la legibilidad del código con el uso de sangría significativa. *Python* se tipifica dinámicamente, admite múltiples paradigmas de programación, incluida la programación estructurada, orientada a objetos y funcional. (Robledeano, 2019).

Gracias al ser un lenguaje multiplataforma se convierte en uno de los preferidos para la enseñanza tanto a nivel de colegio, así como en la universidad.

Una de sus características principales es que se puede trabajar con grandes cantidades de datos, ya que este lenguaje trabaja muy bien la extracción y procesamiento de datos.

Cabe recalcar que, si previamente se tiene conceptos de programación o ya se ha aprendido otros lenguajes, no representará un nivel de complejidad alto a la hora de analizar y entender los códigos de programación que se creen en Python. (Robledeano, 2019).

Junto con Django, *Python* fue el núcleo central del servidor y es el que gestiona el ingreso y salida de mensajes, y *tokens*.

Django Rest Framework (DRF)

DRF es una herramienta de desarrollo de software escrita en Python utilizada para simplificar y reducir considerablemente la cantidad de código escrito para la creación de un interfaz REST.

De sus características más importantes es que cuenta con una interfaz desde la cual se pueden realizar diferentes pruebas sobre los procedimientos HTTP, los cuales son: POST y GET. (inLab, 2016).

Una manera de conseguir datos de distintas fuentes es usando una *API REST*, la cual brinda diferentes *endpoints* los cuales son accesibles con el fin de obtener información determinada. (inLab, 2016).

Otro punto a recalcar que los iniciadores del proyecto *Django* se dedican a utilizar con frecuencia las *Generic Views*, ya que las versiones actuales del programa se basan en clases, mas no en funciones, pero esto tiene un fin, el cual es aprovechar todas las ventajas de lo que se denomina programación orientada a objetos. (Romero, 2023).

MySQL Server

En el vasto mundo de la programación se hace casi imposible no trabajar con algún sistema que permite gestionar bases de datos. Todos los programas creados más tarde que temprano tendrán la necesidad de guardar información en algún espacio de memoria, lo esencial que se necesita almacenar será un listado de usuarios autorizados, sus credenciales y características. (Robledano, 2019).

Es así como nace *MySQL*, un programa de administración de bases de datos que tiene una doble licencia. Ya que es de código abierto, pero por otro lado tiene también una versión comercial administrada por la compañía *Oracle*. (Robledano, 2019).

Características de MySQL

Cuenta con varias ventajas que lo convierten en un programa muy llamativo para los programadores. Una de las principales es que trabaja con bases de datos relacionales, en otras palabras, significa que usa tablas múltiples que tiene conexión entre sí para guardar todos los datos y ordenarlos adecuadamente (Robledano, 2019)..

Gracias a que su código es abierto, hace que sea de fácil acceso y por tanto la gran cantidad de desarrolladores que trabajan en creación de páginas Web se han dedicado a trabajar con *MySQL* en al menos uno de sus proyectos de programación, ya que al estar extendido cuenta con una comunidad inmensa que proporciona soporte a los demás usuarios. A continuación, se presenta otras características importantes:

- Arquitectura Cliente y Servidor: Esto hace que clientes y servidores sean capaces de estar comunicados entre sí de forma diferenciada para obtener un mejor rendimiento. De esta manera cada cliente realiza consultas mediante el sistema de registro y así obtener información, modificarla, guardar dichos cambios o crear nuevas tablas de registros.
- Compatibilidad con SQL: *MySQL* no tiene inconveniente a la hora de haber trabajado con otros motores de bases de datos, ya que ofrece plena compatibilidad, de esta manera una migración a *MySQL* no conlleva un problema.
- Vistas: Centrándonos en bases de datos inmensas las vistas se convierten en un recurso muy necesario. Por tanto, proporciona compatibilidad para realizar configuraciones de vistas personalizadas.

- Procedimientos almacenados: *MySQL* no procesa la información directamente, pues primero mediante procesos almacenados incrementan la eficiencia de cualquier implementación.
- Desencadenantes: Permite la automatización de algunas tareas en el interior de las bases de datos. Esto hace que, al momento de producirse un evento rápidamente otro evento es lanzado para actualización de registros u optimización de sus funciones.
- Transacciones: Esto representa la actuación de varias operaciones sobre la base de datos como un dispositivo. La sección que avala que absolutamente todos los procesos se establezcan de forma correcta o ninguna de estas, es el sistema de base de registros.

MySQL es un sistema de código abierto de manejo de base de datos relacional, organiza los datos en una o varias tablas relacionadas que ayudan con la estructura de los datos. Se puede encontrar en modo stand-alone, o empaquetado en otros componentes como XAMPP, LAMP. WordPress. Varios sitios populares usan este sistema incluyendo Twitter, Facebook, Youtube.

MySQL se utilizó como motor de persistencia o base de datos para almacenar usuarios, mensajes, y tokens id de los diferentes dispositivos.

Nginx

Es un software libre que se usa para servicios Web, caracterizado por alto desempeño y estabilidad, compatible con aplicaciones o desarrollos *en* Django Rest Framework.

Adicionalmente sirve para proxy y balanceador de carga. Puesto a que sus inicios datan sobre la optimización del rendimiento bajo escala, *Nginx* con frecuencia prevalece ante otros servidores Web en términos de pruebas de rendimiento, puntualmente en circunstancias con contenido estático y/o un gran número de solicitudes concurrentes (Kinsta, 2022)..

El diseño de *Nginx* se basa en brindar un bajo uso de memoria y alta concurrencia. Usa una orientación asincrónica basada en eventos en los cuales las solicitudes se conducen en un único hilo.

Las características relevantes de *Nginx* incluyen: Proxy inverso con caché, IPv6, balanceo de carga, soporte *FastCGI* con almacenamiento en caché, Websockets, manejo de archivos estáticos, archivos de índice y auto indexación, TLS/SSL con SNI. (Kinsta, 2022).

Nginx es el servidor Web en el cual la aplicación de Gateway se desplegó.

Java

Es una plataforma de lenguaje de programación Open Source de propósito general, muy robusto se basa principalmente en la programación orientada a objetos, es multiplataforma debido a que el código es interpretado por una máquina virtual JVM (en inglés, *Java Virtual Machine*). (Java, 2022).

Java con el pasar del tiempo ha ido evolucionando desde sus inicios hasta la actualidad donde ha sido de gran ayuda para impulsar a todo el mundo digital, puesto a que se ha convertido en una plataforma muy confiable en la cual se crean cada día nuevos servicios y aplicaciones.

Actualmente las aplicaciones Java acoplan el tiempo de ejecución y la aplicación de *Java*, sin embargo, aún existen aplicaciones y páginas Web que no logran funcionar sin previamente instalar *Java*; de ahí radica la importancia que ha tenido *Java*. Cabe recalcar que existen sitios Web oficiales tanto para desarrolladores y usuarios que deseen aprender programación; así como también para profesionales que están inmersos en el mundo del desarrollo de software. (Java, 2022).

Utilizado por ser multiplataforma para el desarrollo de las aplicaciones de escritorio, mostrando la notificaciones o alerta del sistema de alerta temprana en las interfaces de Windows y Linux; se usó la versión Java JDK 1.8.0_291.

Android Studio

Es el Ambiente integrado de desarrollo (IDE), oficial de Google específico para el sistema operativo Android, disponible para varias plataformas y sistemas operativos. (AndroidStudio, 2023).

Las aplicaciones se pueden programar en *Java* o *Kotlin* en este caso se usó la versión *Java* JDK 11.0.12.

Android Studio contiene un editor de códigos completo y en cuanto a las herramientas que brinda a los desarrolladores proporciona varias funciones que incrementan la productividad al momento de desarrollar aplicaciones para Android, a continuación, se nombran a las siguientes:

- Un emulador veloz y cargado de varias funciones.
- Un entorno unido en el que se puede desarrollar sin ninguna inconveniente para todos los dispositivos Android.
- Incorporación con *GitHub* y plantillas de código para facilitar la compilación de funciones de las aplicaciones; así como también lograr la importación de código de muestra.
- Compatibilidad con *C++* y *NDK.* (AndroidStudio, 2023).

JavaScript

Se denomina como un lenguaje de programación que hace posible la implementación de funciones complejas en sitios Web, cada ocasión que una página realice actividades a parte de mostrar información estática, permite observar las actualizaciones de contenido, animación de gráficos 2D/3D, mapas interactivos. (Mozilla, 2022).

Este lenguaje de programación permite la creación de contenidos y que su actualización sea dinámica, permite el control multimedia, realizar animación de imágenes, entre otros.

Es un lenguaje de programación que es núcleo del 98% de sitios Web, es *client-side* principalmente, es decir que se ejecuta en el lado del cliente.

Del lado del cliente, JavaScript tiene algunas características de programación, a continuación, se presenta algunas relevantes:

- Guardar información útil dentro de variables.
- Operaciones sobre fragmentos de texto.
- Ejecuta código de programación en respuesta a varios eventos que suceden en una página Web.
- Se usó para desarrollar las aplicaciones móviles con la ayuda de una librería denominada React Native. (Mozilla, 2022).

React Native

Es un framework para aplicaciones nativas multiplataforma, cuyo pilar se sustenta en *JavaScript* y *ReactJS*. (Jiménez, 2019).

Para aportar con la información sobre este framework, a continuación, enlistamos lo que no es *React Native*:

- No es un framework que permite la ejecución de aplicaciones que estén desarrolladas en *ReactJS* en un dispositivo móvil.
- No puede funcionar como aplicaciones con *PhoneGap* o *Cordova.*

Este framework contiene un “VirtualDOM”, en el cual se tiene el *JSX*, donde se define los archivos HTML, para luego convertirse en elementos del navegador con la ayuda de JavaScript. (Jiménez, 2019).

Ahora con *React Native* sucede algo similar, ya que se tiene componentes JSX, en este caso se tiene diferentes tags en comparación a los de HTML.

Lo que ocurre después es que el compilador que cuenta con *React Native* convertirá los componentes en elementos nativos de la interfaz tanto para *iOS* como para *Android*. (Jiménez, 2019).

Es una librería del popular lenguaje JavaScript desarrollada por Meta o Facebook, con la ayuda de esta librería se realizó el desarrollo para notificar las alertas o mensajes en dispositivos iOS (iPhone) y Android con el mismo código.

Interfaz de Programación de Aplicaciones API

Es un intérprete que utiliza por lo general protocolo *HTTP* para ayudar a dos sistemas o poder comunicarse bidireccionalmente.

Comúnmente y haciendo una analogía se habla de *API* como un mesero que toma la orden o petición del cliente (front-end) y la lleva a la cocina (back-end), una vez procesada la petición el mismo mesero es quien se encarga de llevar el plato al cliente (de Souza, 2020).

API REST

Antes de ver el concepto de *API REST*, se debe hablar sobre que es una *API, y* no es más que un grupo de requisiciones que permiten la comunicación de información entre aplicaciones.

REST significa transferencia de estado representacional, define un conjunto de funciones como Obtener (*GET*), Insertar (*POST*), Actualizar (*PUT*), Eliminar (*DELETE*), etc.

Una definición relevante para *API REST*, nos dice que se conforma de un grupo de buenas prácticas usadas en los requerimientos *HTTP* creadas por una API en una aplicación Web. Abarca la comprensión del funcionamiento de esta tecnología, los tipos de API existentes y su relevancia en una página Web.

Tenemos algunas ventajas que nos brinda *API REST*, uno de ellos es la separación entre cliente y servidor; esto conlleva a que se trata al *front-end* y *back-end* de las aplicaciones de manera separada, lo que permite una mayor protección en el almacenamiento de datos, porque los datos que se transmiten entre sí, es netamente para la recuperación de información o para insertar y eliminar registros nuevos. (de Souza, 2020).

Estas son las *API* más populares que se encuentran en la mayoría de sistema en la actualidad. Por lo general el cliente es quien inicia las comunicaciones enviando solicitudes al

servidor. Este último recepta datos y mediante métodos y funciones devuelve lo que el cliente solicitó, *JSON* es el tipo de datos que utiliza *REST* para enviar y recibir información.

DNS

Es un servidor especial que usa una base de datos de nombres de dominio *DNS* (en inglés, *Domain Name Service*), para responder ante las consultas relacionadas al *DNS*. Gracias a DNS las personas en Internet pueden colocar un dominio. El servicio de nombres conoce las distintas combinaciones de dominios y direcciones IP o conoce a qué otro servidor *DNS* le debe reenviar una solicitud. Así, al momento de acceder a un sitio Web, inicialmente se crea una solicitud a uno o varios servidores *DNS* para que al final se realice la conexión al sitio Web. (Ionos, 2022).

Dominio

Un dominio lo podemos denominar como la dirección física de una página Web. Es de gran ayuda para las personas al momento de ubicar con facilidad un sitio Web en lugar de usar su dirección IP. (Gustavo, 2018).

Por facilidad se obtuvo un dominio que ayuda a optimizar la declaración de variables globales como la *URL* y los endpoints, este último fue obtenido en un servicio de venta de *DNS* como lo es *GoDaddy.com*

https://alertaewbs.site/

SSL

La capa de sockets seguros o *SSL* (en inglés, *Secure Sockets Layer*), es la tecnología estándar que permite tener una conexión segura y fiable a Internet, encripta y protege toda la información que se transmite desde un cliente a un servidor Web, sin un certificado la información se transmite en texto plano, susceptible a espionaje o robo con el uso de herramientas como un *sniffer* de red. Existen certificados gratis como el usado en este trabajo de investigación con la ayuda de *Lets Encrypt* y pagados como lo usan páginas o servidores populares por ejemplo Facebook, Youtube, etc. (Gustavo, 2022).

JWT

JSON Web Token es un estándar abierto de la industria que se utiliza para compartir información entre dos entidades, generalmente un cliente (*frontend*) y un servidor (*backend*).

Los t*okens Web JWT* o *JSON* se usan con mayor frecuencia para identificar a un usuario autenticado. Son emitidos por un servidor de autenticación y son consumidos por el cliente-servidor (para proteger sus *API*). Se declara un mecanismo para poder transmitir entre dos partes de forma segura, la identificación de un determinado usuario, así como también privilegios y promesas. (López, 2020).

GitHub

Repositorio donde se alojó el proyecto y herramienta gratuita utilizada para versionar el código, se lo denomina como un portal desarrollado por Linus Torvals, se creó con la finalidad de almacenar el código de las aplicaciones de cualquier programador. Esta plataforma justifica su creación al ser una herramienta para que los programadores almacenen sus códigos y herramientas de las aplicaciones que están creando, es de gran ayuda para el versionamiento de código, para el desarrollo colaborativo cuando el grupo de trabajo grande y la aplicación demanda integración continua, existe control de usuario de acceso se puede publicar código público o privado para descargar dichas aplicaciones y va más allá ya que se puede ingresar al perfil de cada usuario registrado en el portal para de esa manera leer más acerca de los proyectos subidos por los programadores. (Fernández, 2019).

Cloud Computing (Digital Ocean)

Computing Cloud son recursos pagados o gratuitos que se pueden acceder desde cualquier lugar donde se tenga una conexión a Internet, se pueden desplegar máquinas virtuales, aplicaciones, base de datos, balanceadores de carga, funciones, etc., mediante el uso de Cloud como servicios, en este caso en particular se ha comprado una instancia de una máquina virtual (Droplet), con Ubuntu para la implementación del sistema de Gateway.

Digital Ocean es una empresa que tiene su sede en la ciudad de *New York*, la cual ofrece servicios de almacenamiento en la nube.

Entre sus características importantes está la escalabilidad, esto quiere decir que admite que se cuenta con más recursos a medida que progresa el desarrollo de algún proyecto digital. Esta escalabilidad nace producto de que los servicios de la empresa se encuentran basados en la nube, esto la convierte en la mejor opción ante sus competidores. (Matheus, 2020).

Capítulo III

Desarrollo e implementación de un Gateway Server para decodificar y retransmitir mensajes del sistema de alerta temprana en servicios de televisión digital terrestre

El presente capítulo describe el desarrollo del servidor Web público en el servicio de *Cloud Computing* de *Digital Ocean* con despliegue a producción en *Ubuntu 20.04* con el stack *Nginx*, *Gunicorn*, *Django Rest Framework*, las aplicaciones móviles con la ayuda de *React Native* con expo cli, la aplicación de escritorio multiplataforma codificada en *Java*, y el script de *Python* que está alojado en el ordenador que recibe el mensaje y lo replica a los diferentes clientes finales.

Arquitectura del Sistema Completo

La Figura 6 muestra el diagrama de bloques completo del proyecto el cual representa un escenario de transmisor y receptor de televisión digital terrestre completo que en su TS (en inglés, *Transport Stream*), contiene el mensaje de alerta temprana. En el presente trabajo de investigación tiene más relevancia el recuadro titulado como Réplica, que consiste en leer el mensaje y retransmitirlo por las diferentes Apps (Móvil o de Escritorio), que usan los clientes finales y garantizar que la notificación llegue de forma oportuna con el mensaje de alerta temprana, se dividió el proyecto en 3 partes listadas a continuación:

- Servidor público, VPS o Droplet: Persistencia de datos, tokens y consulta para clientes de escritorio.
- Script Python: Lee el mensaje, da la señal de alerta y notifica a los clientes móviles.
- Aplicaciones: Web o móviles para recepción del mensaje de alerta temprana.

Se realizará una corta descripción del hardware utilizado para tener una idea concisa del escenario en la arquitectura del sistema completo.

Transmisor

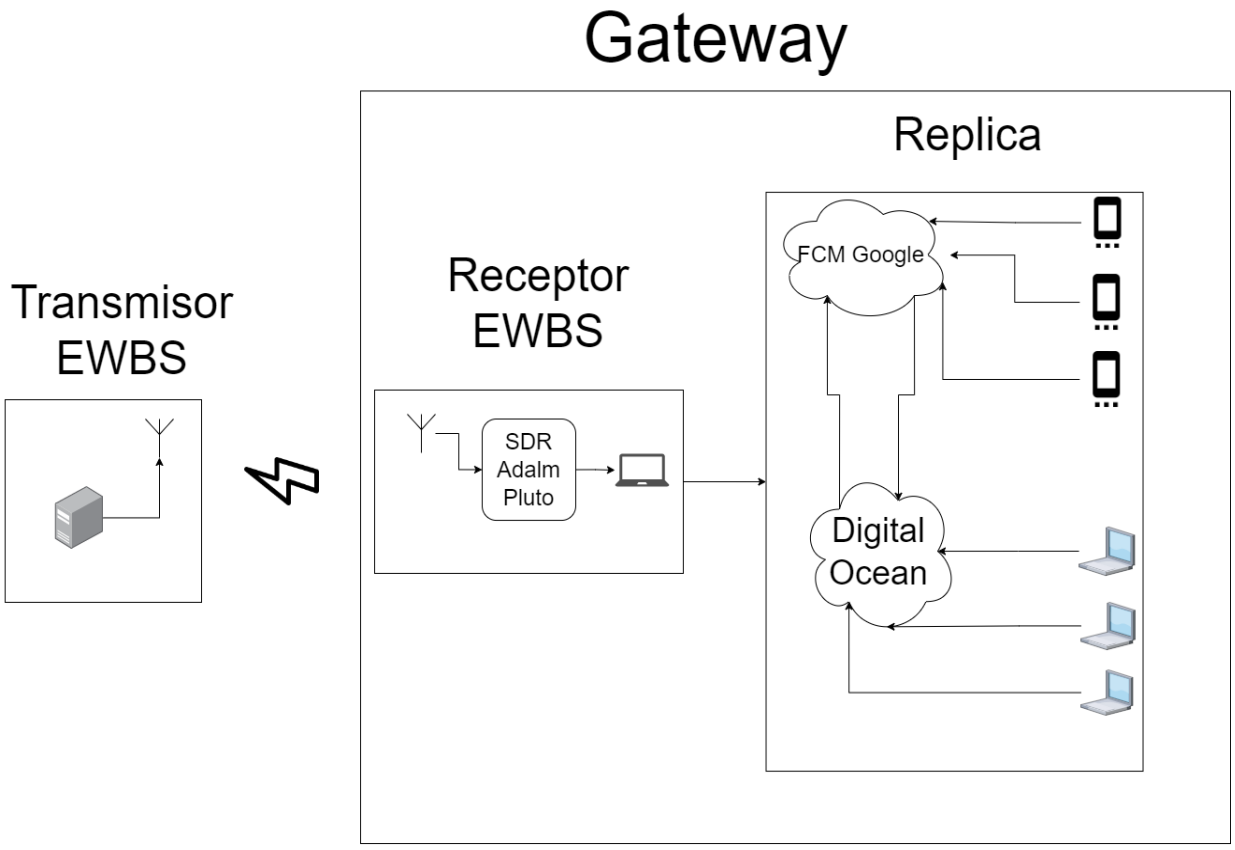
- Computador: Tiene una versión del software StreamXpress que soporta la carga de archivos con los datos de video, audio y el mensaje de alerta codificado TS (en inglés, *Transport Stream*).
- Modulador: Tarjeta DeKtec DTA-211 es un periférico USB del ordenador que le permite a través del software StreamXpress transmitir la señal al aire, trabaja en el rango de VHF a UHF, soportando estándares ISDB-Tb.
- Antena: Utilizada para transmitir la señal de salida.

Receptor

- Ordenador: Encargado del procesamiento de la señal, utilizando algoritmos de programación obtiene el mensaje y lo guarda en una ruta específica.
- SDR Adalm Pluto: Un dispositivo que permite procesar señales en el rango de *UHF* compatible con el estándar *ISDB-Tb*.

Figura 6

Diagrama de bloques del sistema de televisión digital terrestre

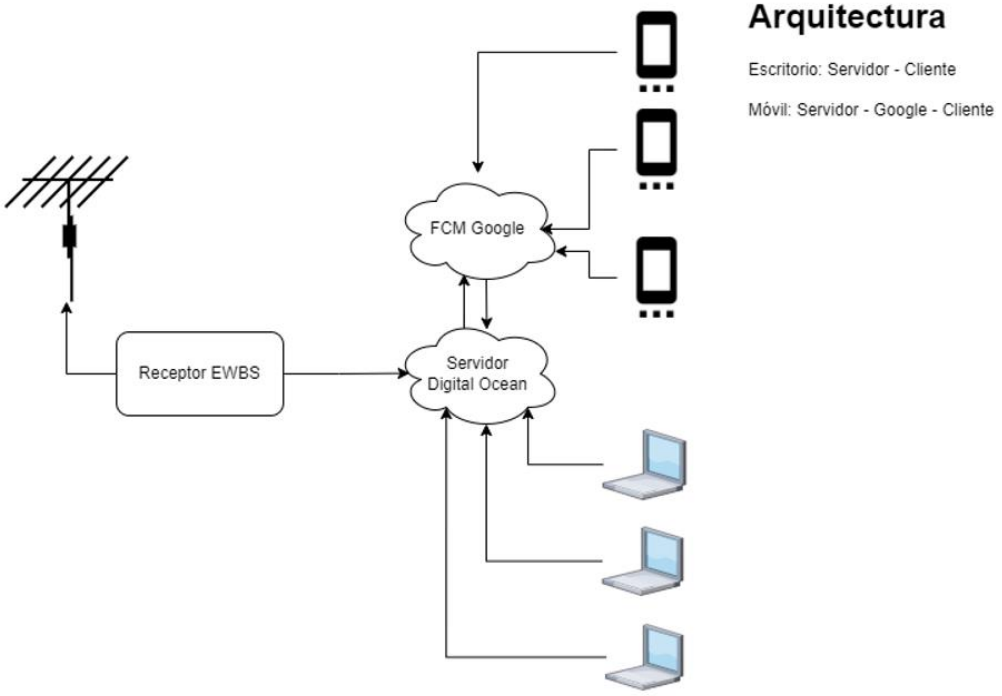


Nota. La figura muestra un sistema de televisión digital y un Gateway Server implementado en el laboratorio de televisión digital de la Universidad de las Fuerzas Armadas – ESPE.

A continuación, se observa en la Figura 7 el diagrama de bloques general del *Gateway Server*, el receptor recibe el TS que contiene los datos de audio y video de la transmisión y también el mensaje de alerta temprana encriptado, el bloque de Receptor *EWBS* recibe, decodifica el mensaje y lo guarda en una ruta específica, cabe recalcar que este último es un SDR (en inglés, *Software-Defined Radio*), Adalm Pluto conectado a un computador con sistema operativo Linux, que opera con GNU-Radio.

Figura 7

Diagrama de bloques del Gateway Server



Nota. La figura muestra un Gateway Server implementado en el laboratorio de televisión digital de la Universidad de las Fuerzas Armadas – ESPE utilizando cómputo en la nube.

Servidor Público VPS, Droplet

Arquitectura del Gateway Server.

A continuación, se puntualiza cómo funciona el sistema de replicación del mensaje, primero se explicará el servicio en la nube alojado en *Digital Ocean* en el cual tiene como sistema operativo *Linux* (Ubuntu 20.04 LTS), se empezará con la descripción de la base de datos.

Base de Datos

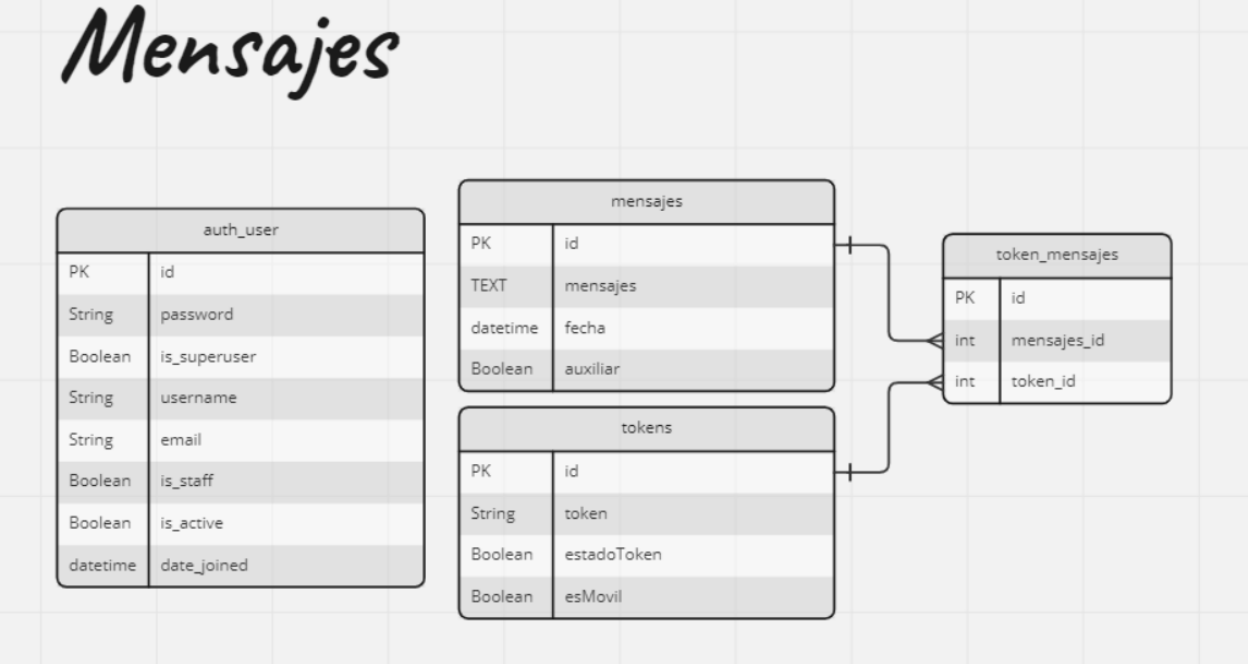
La base de datos es una instancia de *MySQL* 8 instalada en el *VPS*, utilizada como persistencia de datos como mensajes, tokens y mediante lógica relacional almacena los mensajes leídos por los diferentes clientes.

Diagrama Entidad Relación

En la Figura 8 se puede observar las diferentes tablas que se utilizan en la base de datos relacional *MySQL* denominada Mensajes a continuación se explicará el uso de cada una, para el correcto funcionamiento del Gateway Server.

Figura 8

Diagrama entidad relación de la base de datos Mensaje



Nota. La figura muestra un sistema de televisión digital y un Gateway Server implementado en el laboratorio de televisión digital de la Universidad de las Fuerzas Armadas – ESPE.

Tabla auth_user (Propia del framework)

Tabla que almacena información del super usuario, el cual tiene el permiso para guardar nuevos mensajes y consultar los tokens de los diferentes dispositivos. Un usuario no registrado no podría enviar alertas.

Tabla mensajes

Tabla que almacena los mensajes de alerta temprana, con su respectivo timestamp o fecha de creación.

Tabla tokens

Tabla que almacena los tokens id (*Device ID*), de cada dispositivo que ha instalado la aplicación como clientes de escritorio o app móviles. Contiene dos banderas para identificar si el token es móvil “esMovil”, y otra si el token es de iOS o Android “esAndroid”.

Tabla token_mensajes

Tabla que contiene la relación varios a varios (en inglés, *Many to Many*), entre los mensajes y los tokens, consultando esta tabla nos podemos dar cuenta que dispositivo ha recibido satisfactoriamente el mensaje enviado.

Existen otras tablas que usan el propio framework DRF para el control de logs, sesiones y migraciones que no son muy relevantes para el proyecto.

Servidor *Web (Nginx)*

El servidor Web de *Nginx* que está ejecutándose en el puerto 80 que sirve como proxy inverso, redirecciona las peticiones a *Gunicorn*, escalable a miles de conexiones simultaneas

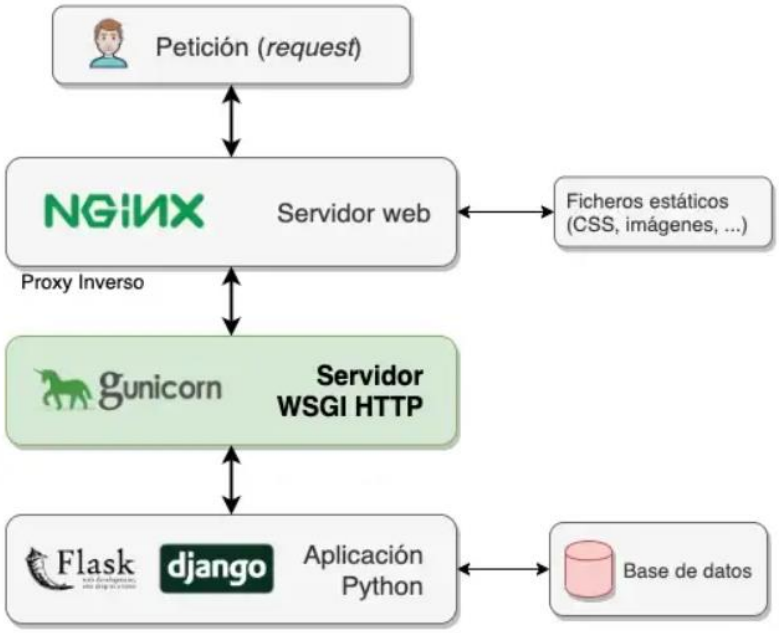
https://alertaewbs.site

Gunicorn

Servidor de aplicaciones que traduce las solicitudes *HTTP* en lenguaje que *Python* pueda entender mediante la interfaz de enlace del servidor Web WSGI (en inglés, *Web Server Gateway Interface*), que es un estándar entre servidores Web y aplicaciones Web, en la Figura 9 se puede observar cómo funciona la arquitectura *Nginx*, *Gunicorn* y *Python*,

Figura 9

Diagrama de Bloques funcionamiento de Nginx, Gunicorn y Python



Nota. La figura muestra el diagrama de bloques del funcionamiento de Nginx como servidor web de python con Django. Tomado de *¿Qué es un WSGI?,* por Ignacio Alonso, 2019.

Django Rest Framework

Es un framework de *Python* para aplicaciones Web que optimiza la cantidad de código para crear una aplicación Web maneja la lógica de la aplicación.

Endpoints

En la tabla muestra los 4 *endpoints* principales de la aplicación se describe este último su respectiva interacción con el sistema y la base de datos, además de adicionarles el *Autenticable* para identificar si es una ruta que está asegurada con *JWT*.

Tabla 1

Endpoints del servicio Web

Ruta	Tipo	Acción
/Show/	GET	Lista mensajes.
/Insert/	POST(Autenticable)	Inserta mensajes.
/tokens/	POST	Guarda Tokens
/ShowToken/	POST(Autenticable)	Listar Tokens

Nota. Esta tabla muestra los endpoints expuestos en el servicio Web.

Seguridad

Como primera medida de seguridad se implementó un certificado digital SSL para encriptar el intercambio de datos entre el servidor y los clientes, el certificado gratuito mediante *Let's Encrypt* se instaló en el *Droplet* de *Digital Ocean* bajo el dominio https://alertaewbs.site/ rentado en *GoDaddy*.

Como segunda medida de seguridad a posibles ataques se ha protegido dos endpoints del sistema:

- /Insert/ que inserta un mensaje nuevo en la base de datos
- /Tokens/ que inserta y registra un nuevo dispositivo (Device ID), en la base de datos

Usando Json Web Token (*JWT*), que es un estándar RFC 7519 que asegura que el intercambio de datos entre dos partes se den manera segura, con la implementación de estas medidas de seguridad usadas en la mayoría de sistemas y servicios protegemos al Droplet de posibles ataques; la autenticación con los *APNs* de Apple se realiza de la misma manera.

Api Rest (Django Rest Framework)

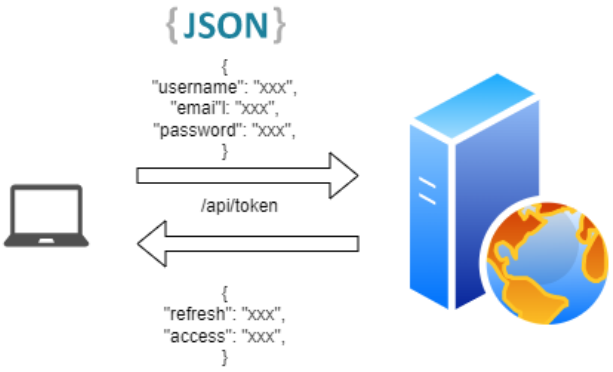
En los siguientes párrafos se describirán los diferentes endpoints y funciones que fueron desarrolladas en Django para que la aplicación pueda recibir autenticaciones, recibir mensajes, mostrar mensajes, mostrar mensajes según el token y almacenar tokens.

Autenticación

La autenticación en el receptor funciona enviando una petición *HTTP* de tipo *POST* al endpoint /api/token/ al servidor con datos de autenticación en este caso email, username, password, este responde con un código de respuesta 200 OK (ver Figura 10), y un access token en el cuerpo de la respuesta en el caso que la autenticación sea correcta, y para el caso que sea incorrecta responde un código 401 (ver Figura 12), que corresponde a No autorizado (en inglés, *Unauthorizated*), en la Figura 10 se puede observar un ejemplo como un cliente se puede autenticar en el servidor mediante el uso de *JWT* y un payload.

Figura 10

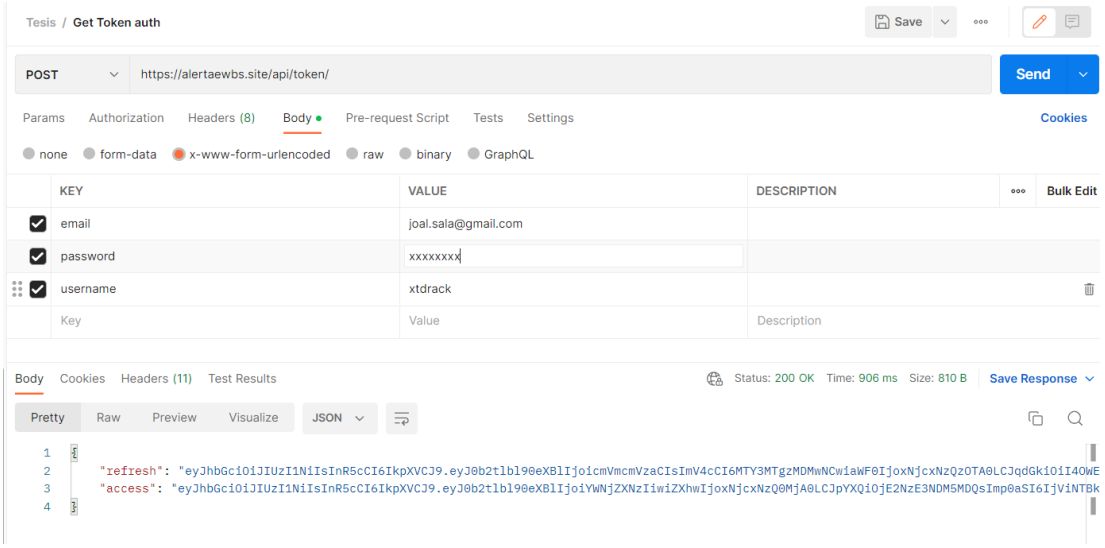
Ejemplo de autenticación JWT



Nota. La figura muestra la autenticación de un cliente hacia el Gateway Server.

Figura 11

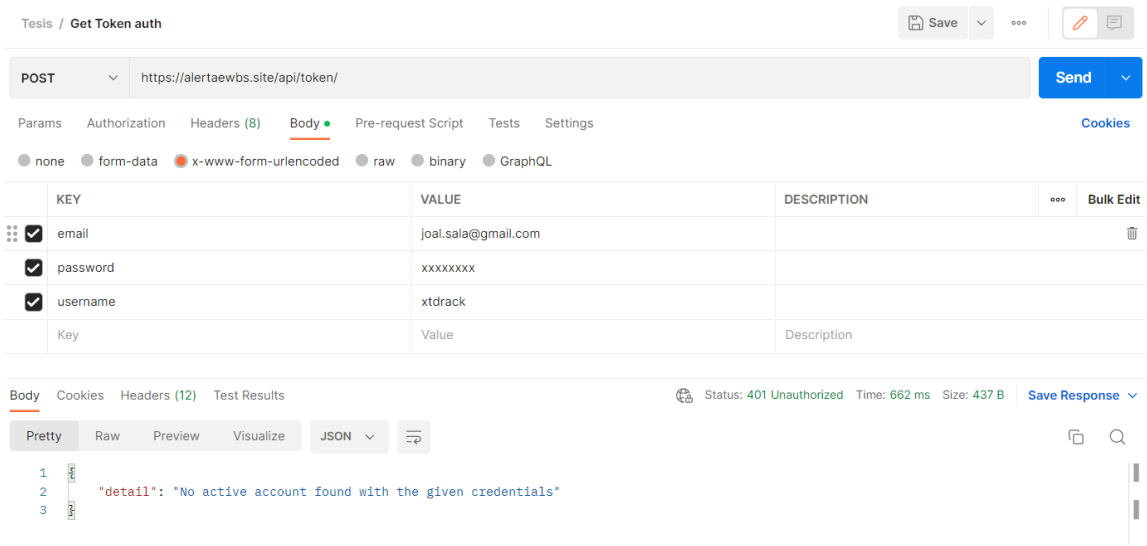
Ejemplo de petición mediante Postman y respuesta correcta



Nota. La figura muestra un ejemplo de petición de tipo POST mediante el software Postman para autenticación y el retorno de una respuesta correcta con estado 200.

Figura 12

Ejemplo de petición mediante Postman y respuesta incorrecta



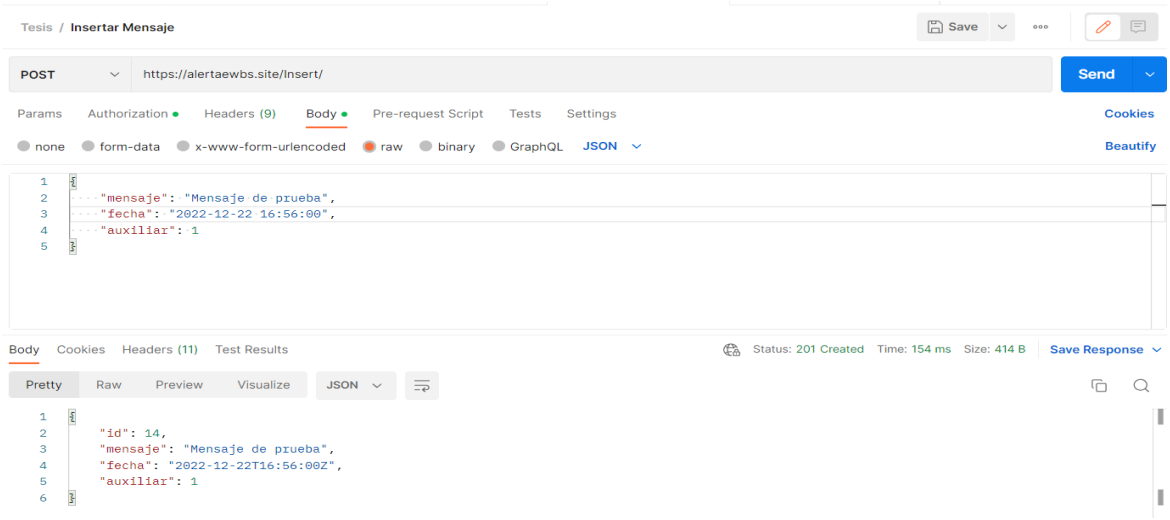
Nota. La figura muestra un ejemplo de petición de tipo POST mediante el software Postman para autenticación y el retorno de una respuesta incorrecta con estado 401 Unauthorized.

Inserción de Mensajes en la base de datos

El receptor *EWBS* lee el mensaje, se autentifica con un email, usuario y contraseña, recibe un Token de Autenticaciones “Bearer Token” y envía el mensaje de alerta temprana al servidor público montado en Digital Ocean (Cloud Computing), por medio de una petición HTTP Post (ver Figura 13), y cabecera con autorización tipo Bearer Token (ver Figura 14).

Figura 13

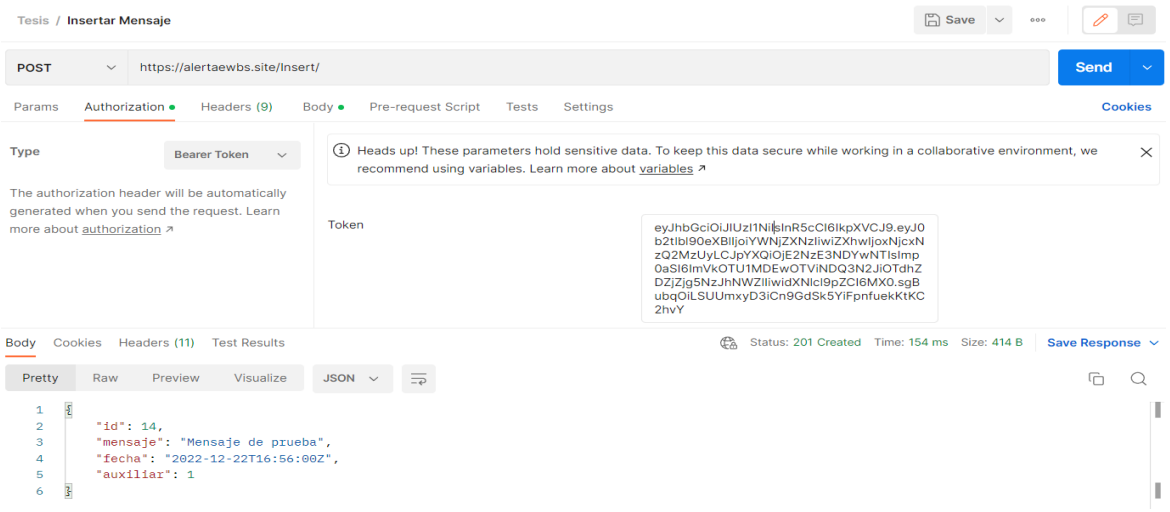
Ejemplo mediante Postman petición post y respuesta para insertar un mensaje



Nota. La figura muestra un ejemplo de petición POST mediante el software Postman para insertar un mensaje, además se muestra el payload necesario.

Figura 14

Configuración de autorización para Postman



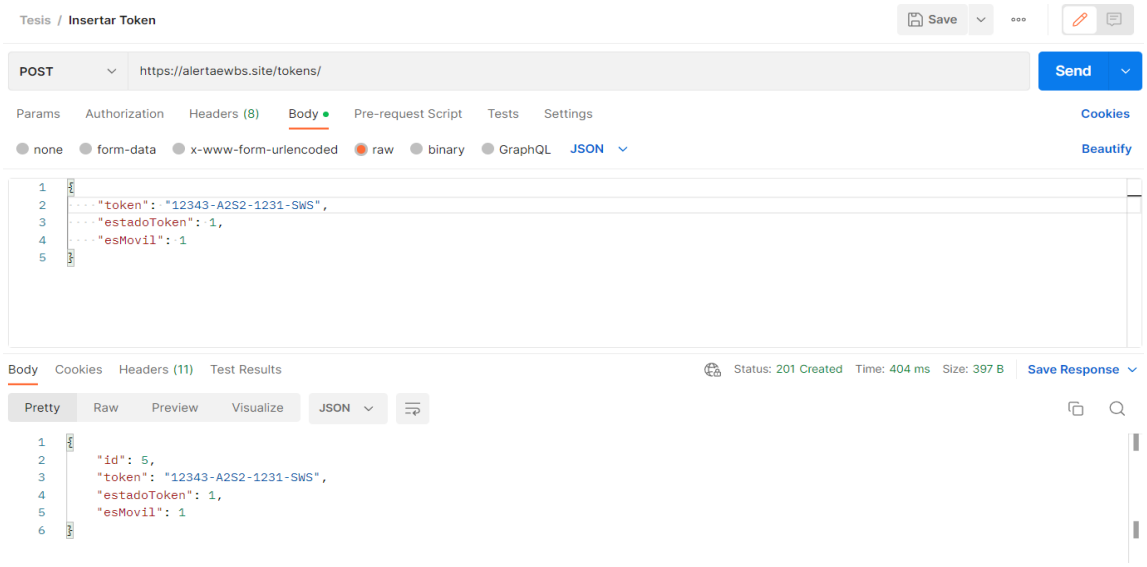
Nota. La figura muestra la configuración necesaria para enviar petición a un servidor con autenticación en este caso la autorización es de tipo Bearer.

Inserción de Token

Mediante este *endpoint* "/tokens/" se guardan los *tokens id* o *device id* dispositivos móviles y de escritorio, para posteriormente poder identificar al dispositivo y recibir notificaciones de mensaje de alerta temprana. En la Figura 15 se puede ver un ejemplo de la inserción de un token en la base de datos.

Figura 15

Ejemplo de petición http post mediante Postman para guardar un token id



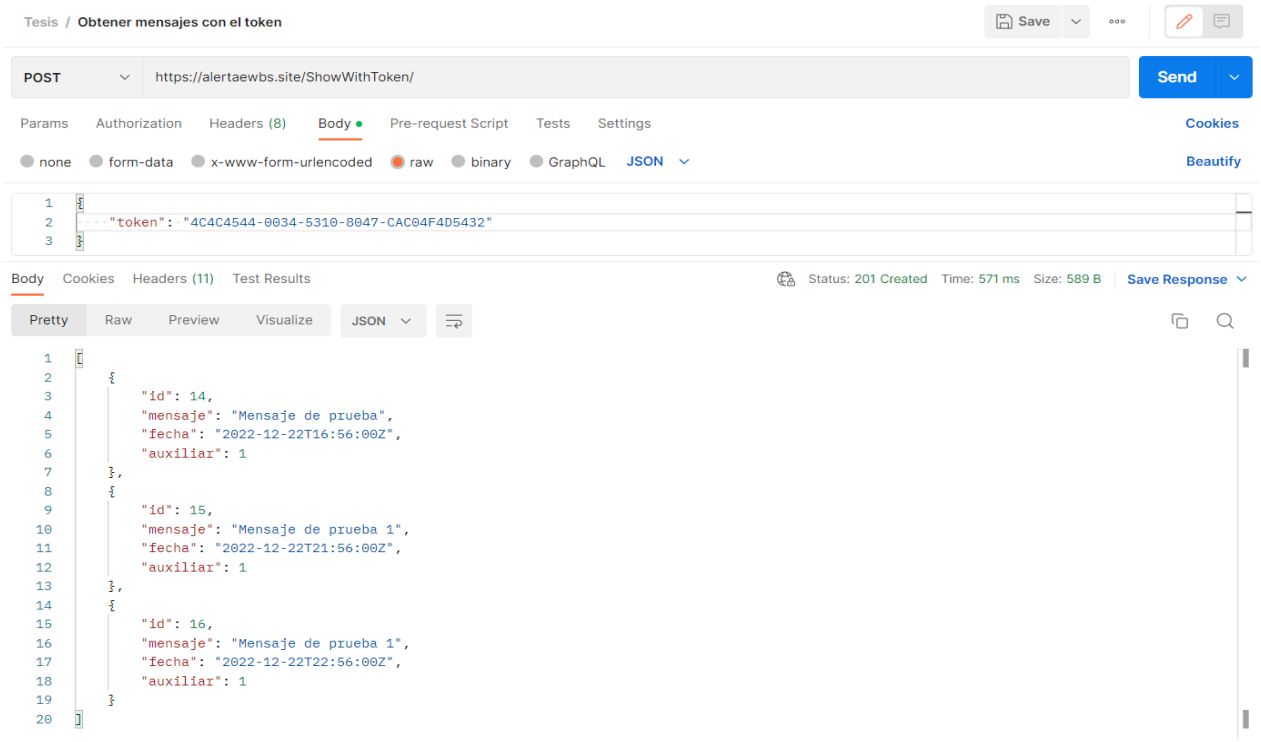
Nota. La figura muestra una petición POST mediante el software Postman para guardar o registrar un nuevo cliente del Gateway Server.

Obtener Mensaje por el Token

Para las aplicaciones de escritorio se utiliza la consulta por poleo (en inglés, *pooling*), y obtiene los mensajes no leídos por el token consultado en específico, una vez lee los mensajes los marca como leídos en la tabla token_mensajes. En la Figura 16 se puede observar un ejemplo de esta petición.

Figura 16

Ejemplo de petición post mediante Postman consulta de mensajes a través de un token id, obtiene solo mensajes no leídos



Nota. La figura muestra una petición POST mediante el software Postman para obtener una lista de mensajes no leídos de un cliente en específico enviando su token id.

Con la explicación de la lectura de mensajes no leídos termina la funcionalidad y el trabajo del *Droplet*, específicamente se utiliza para la consulta por poleo (en inglés, *pooling*) de los clientes de escritorio y para persistir los tokens id, de todos los dispositivos que utilizan el aplicativo.

Script

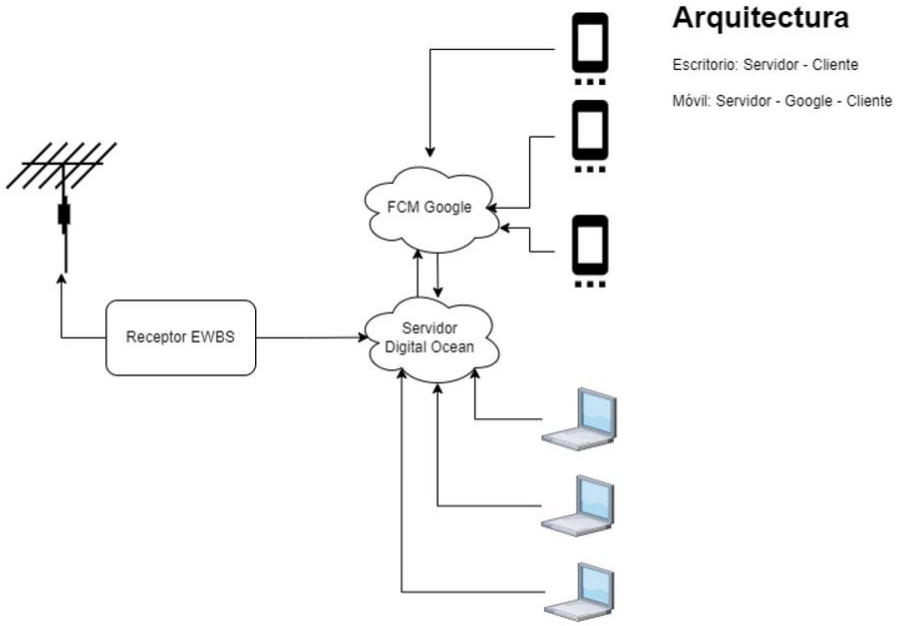
Lectura del Mensaje

A continuación, se explicará puntualmente la lectura del mensaje, esta funcionalidad se encuentra alojada en el ordenador encargado del procesamiento de la señal en conjunto con el

SDR. En la Figura 17 se puede observar el bloque “Receptor EWBS” a continuación se va a describir el funcionamiento y como se realiza la lectura del mensaje de alerta.

Figura 17

Diagrama de bloques del Gateway Server



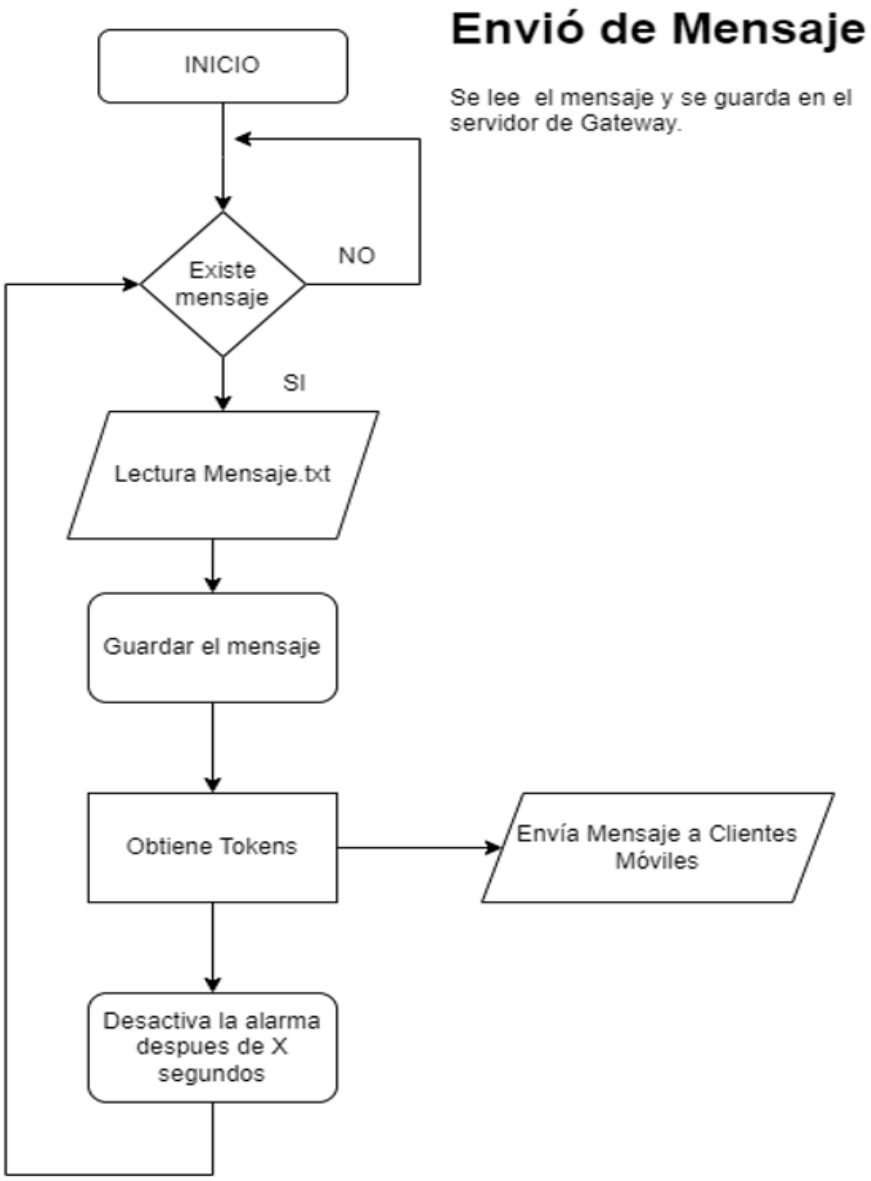
Nota. La figura muestra el diagrama de bloque del Gateway Server implementado.

El diagrama de bloques mostrado en la Figura 18, muestra los procesos y procedimiento a seguir para que el script alojado en la misma ruta donde se guarda el archivo con el mensaje decodificado pueda retransmitir el mensaje.

Mediante el script en *Python* se verifica que el contenido del archivo esté vacío, mientras está vacío el archivo no se realiza ninguna acción, hasta que el transmisor emite una alarma con un mensaje encriptado, este último es recibido cuando el computador que hace de receptor mediante el *SDR*, detecta una alerta y decodifica el mensaje, el mensaje decodificado es guardado en un archivo de texto de nombre “Alerta Emergercia.txt” la ruta “/home/user/Descargas”.

Figura 18

Diagrama de flujo del receptor para la lectura de un mensaje de alerta temprana



Nota. La figura muestra el diagrama de bloques de la lectura del mensaje decodificado, para luego enviar a los diferentes dispositivos.

Aplicaciones

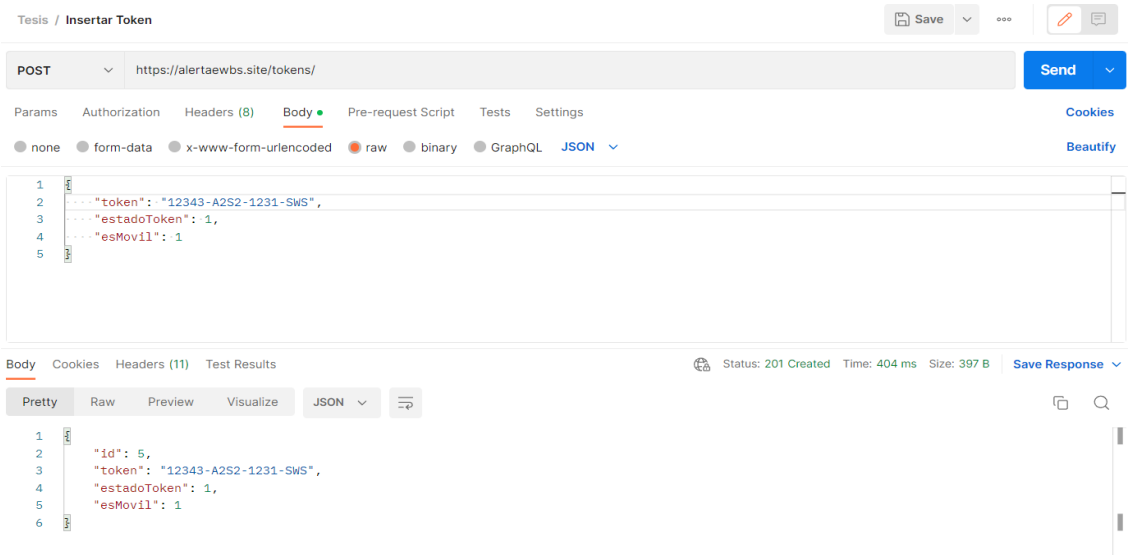
Aplicación de Escritorio

Realizada con el framework *Spring Boot* reduciendo considerablemente las líneas de código para la creación de una tarea programada (en inglés, *Cron Job*), que se ejecuta en el ordenador utilizando el puerto *28080*, se creó modelos de mensajes y tokens ID, para poder mapear respuestas y de igual forma enviar *Request o petición* con un Cliente *REST* al servidor.

Los clientes de escritorio periódicamente mediante una tarea programada, la cual se ejecuta cada 10 segundos, envía su token id por medio una petición *HTTP POST* ver ejemplo de la petición en la Figura 19, el servidor discrimina si es nuevo lo registra, si es un token ya registrado, busca mensajes nuevos para este dispositivo en específico con esto garantizamos que todos los dispositivos reciban el mensaje, indistintamente si tiene conexión a Internet al momento de la alerta en la Figura 20 se puede observar el flujo de la aplicación de escritorio.

Figura 19

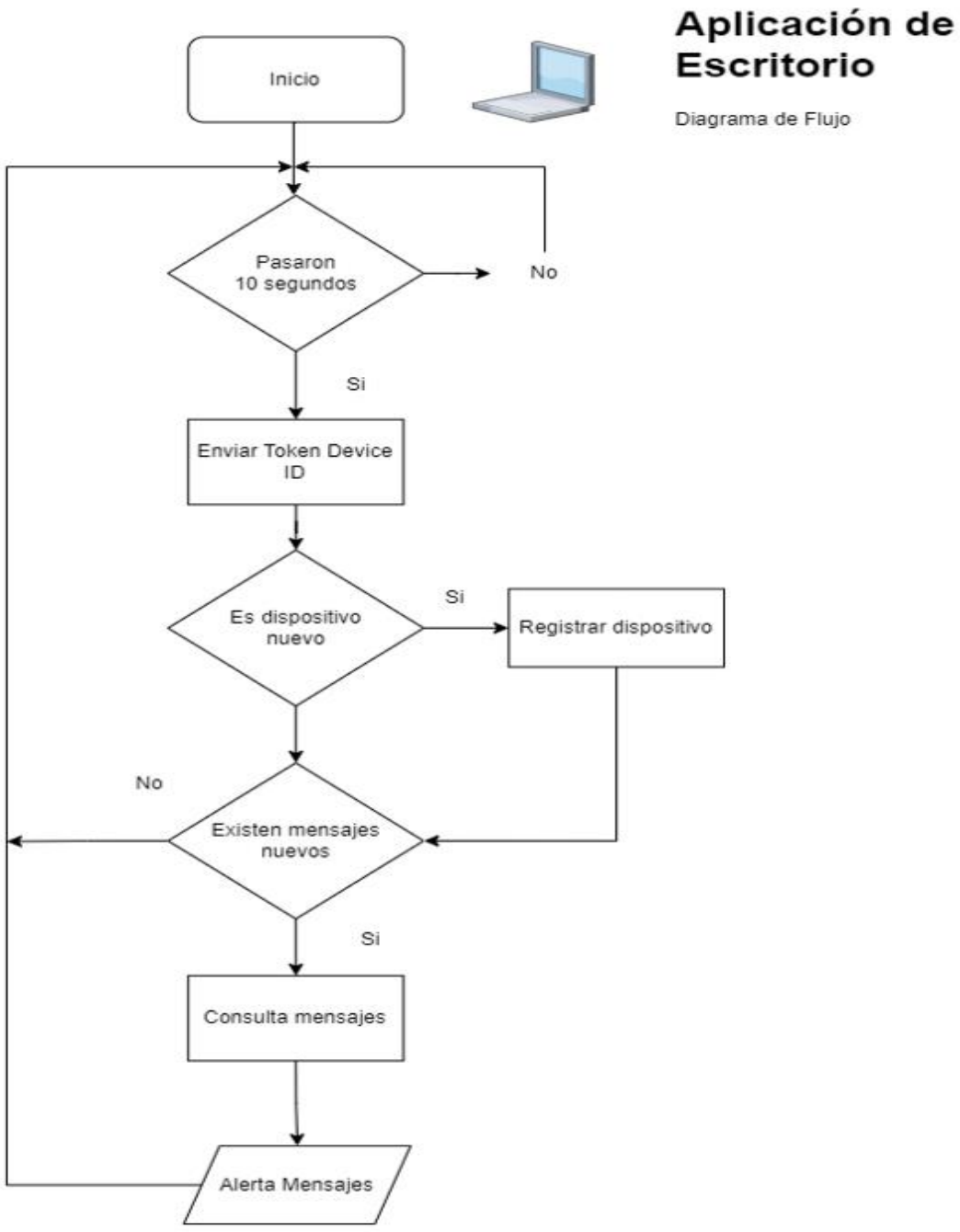
Ejemplo de petición HTTP POST para guardar un Token ID



Nota. La figura muestra una petición POST mediante el software Postman para registrar un nuevo cliente del Gateway Server con su respectivo token id.

Figura 20

Diagrama de flujo del funcionamiento de la aplicación de escritorio

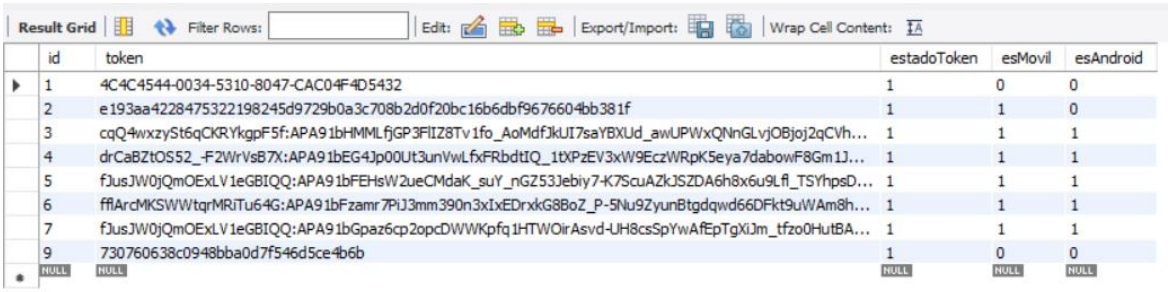


Nota. La figura muestra el diagrama de bloques del flujo de funcionamiento de la aplicación de escritorio desde su registro hasta la recepción del mensaje.

En la tabla tokens (ver Figura 21), se discrimina si es token ID de aplicación de escritorio o de móvil mediante una bandera llamada “esMovil” en la base de datos también existe diferenciación entre sí el móvil es *iOS* o *Android* mediante la bandera “esAndroid”.

Figura 21

Tabla tokens de la base datos



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	token	estadoToken	esMovil	esAndroid
1	4C4C4544-0034-5310-8047-CAC04F4D5432	1	0	0
2	e193aa4228475322198245d9729b0a3c708b2d0f20bc16b6dbf9676604bb381f	1	1	0
3	cqQ4wxzySt6qCKRYkgpF5f:APA91bHMMLfjGP3FIIZ8Tv1fo_AoMdfJkJI7saYBXUd_awUPWxQNnGLvjOBjoj2qCVh...	1	1	1
4	drCaBZtOS52_-F2WrVsB7X:APA91bEG4Jp00Ut3unVwLfxFRbdtIQ_1tXPzEV3xW9EczWRpK5eya7dabowF8Gm1J...	1	1	1
5	fJusJW0jQmOExLV1eGBIQQ:APA91bFEHsW2ueCMdaK_suY_nGZ53Jebiy7-K7ScuAZkJSZDA6h8x6u9Lfl_TSYhpsD...	1	1	1
6	fflArcMKSWWtqrMRiTu64G:APA91bFzamr7PiJ3mm390n3xIxEDrxkG8BoZ_P-5Nu9ZyunBtgdqwd66DFkt9uWAm8h...	1	1	1
7	fJusJW0jQmOExLV1eGBIQQ:APA91bGpaz6cp2opcDWWKpfq1HTWOirAsvd-UH8csSpYwAfEpTgXiJm_tfzo0HutBA...	1	1	1
9	730760638c0948bba0d7f546d5ce4b6b	1	0	0
NULL	NULL	NULL	NULL	NULL

Nota. La figura muestra la tabla tokens con la bandera “esMovil” es la columna que discrimina entre una app móvil o app de escritorio y la columna “esAndroid” que discrimina el sistema operativo del dispositivo móvil.

Aplicación Móvil

La aplicación *AlertaTemprana* móvil ha sido desarrollada con la popular librería de JavaScript desarrollada por Facebook “*React Native*” que a su vez se complementa con otra herramienta que es *expo cli*.

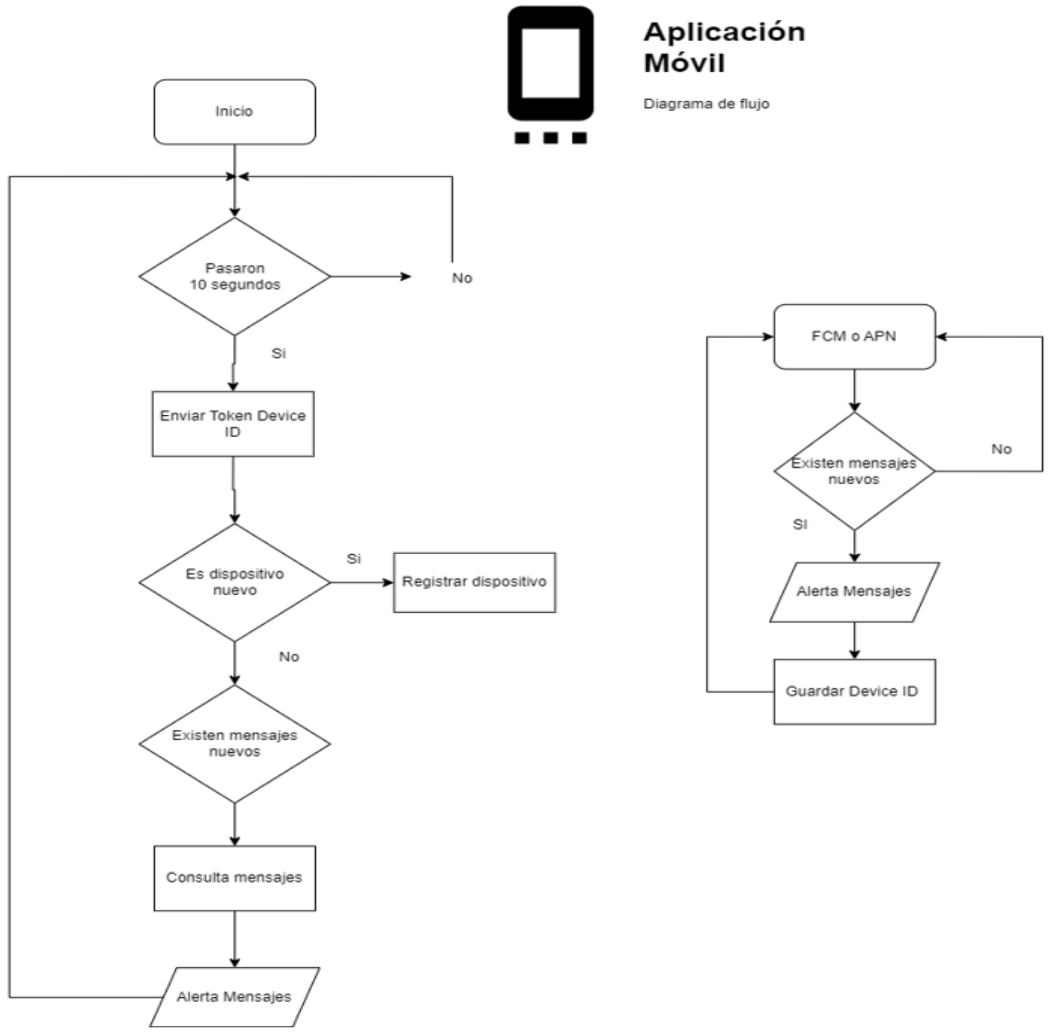
Se usó expo por la facilidad de integración, desarrollo y pruebas, debido a que con expo cli no se necesita tener un ordenador MAC para desarrollar Apps de iOS (iPhone).

El comportamiento es similar a la aplicación de escritorio se puede observar el flujo en el diagrama de bloques de la Figura 22 , cuando la app está abierta, existe un token id (Device ID) nativo el cual después de ser instalada la aplicación y puesta en marcha por primera vez se obtiene y se guarda en la tabla tokens de la base de datos con la bandera esMovil en true y dependiendo del sistema operativo del dispositivo se guardar la bandera esAndroid en true o en false ver ejemplos en la Figura 21, una vez guardado el token con la aplicación ejecutándose en primer plano con el mismo token cada 10 segundos consulta al servidor si existen mensajes

nuevos y se renderiza el componente carrusel (ver Figura 23), que nos muestra los últimos mensajes.

Figura 22

Diagrama de bloques aplicación de escritorio



Nota. La figura muestra el diagrama de bloques del funcionamiento de la aplicación móvil desde el registro y la recepción de mensajes.

Figura 23

Interfaz de la aplicación móvil

Nota. La figura muestra la interfaz de la aplicación móvil, en el recuadro rojo se muestra el componente carrusel que se renderiza cuando llega un nuevo mensaje.

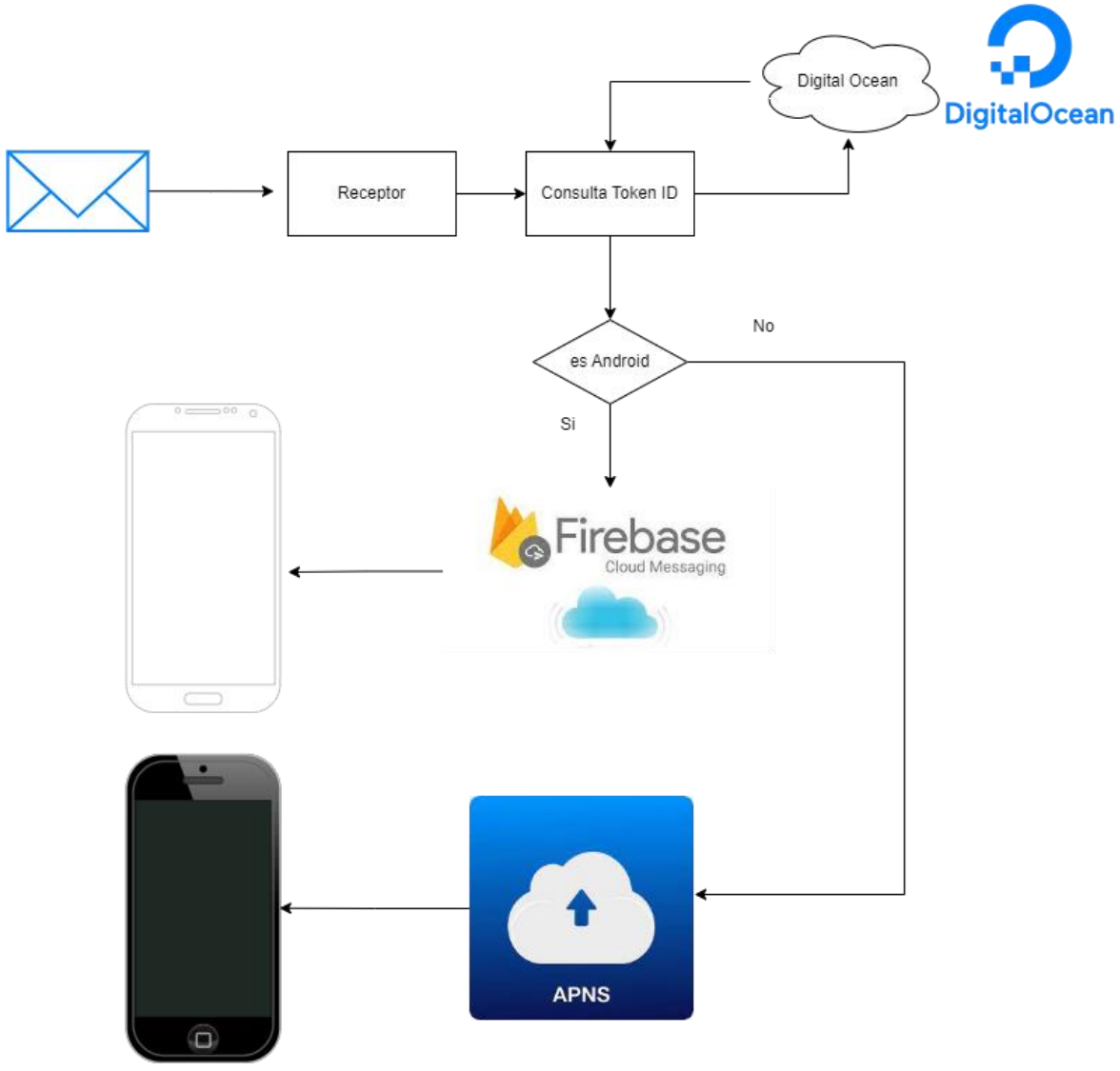
Funcionamiento de la Notificaciones

En la siguiente Figura 24 muestra el diagrama de cómo trabaja el script con las notificaciones, se puede observar que cuando llega un mensaje el receptor pregunta al servidor sobre los tokens móviles a los cuales se debe notificar, luego de tener los tokens id se pregunta

si son iOS o Android por la bandera que se configuró al registro del dispositivo y por último se notifica por medio de *APNs* o *FCM* respectivamente.

Figura 24

Diagrama de bloques del servicio de notificaciones



Nota. La figura muestra un diagrama de bloques del funcionamiento del envío de mensajes de alerta temprana mediante APNs o FCM.

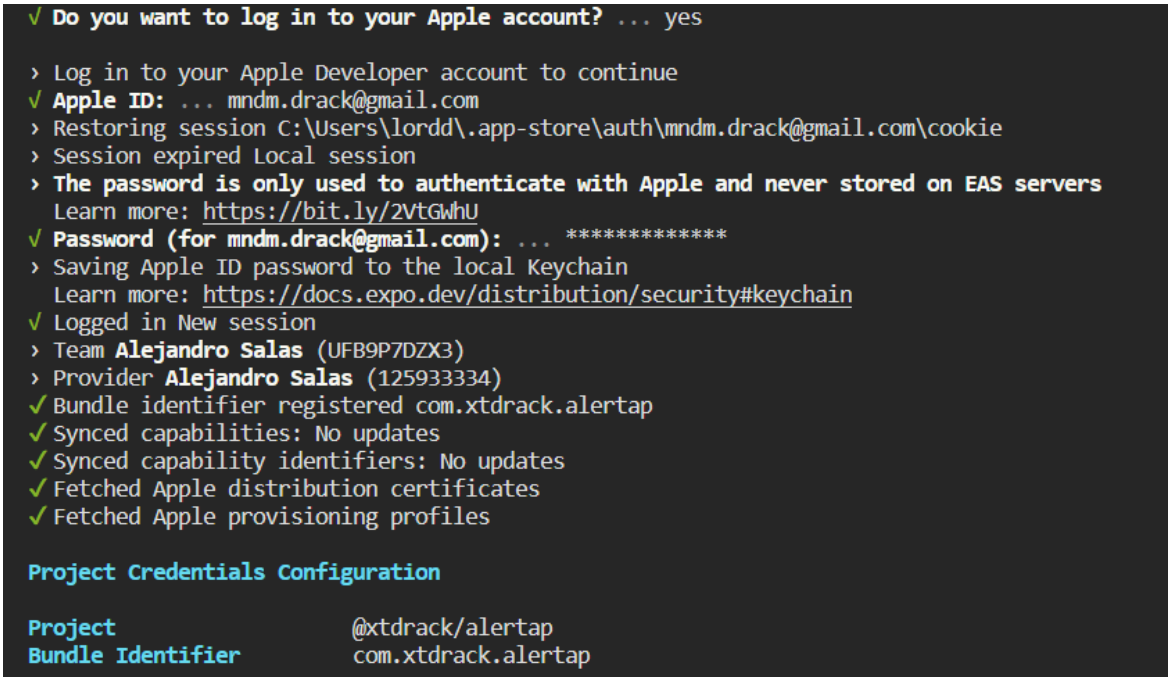
Configuración de las Notificaciones

APNs - IOS

Se debe tener una cuenta de desarrollador para probar la aplicación la cuenta de desarrollador tiene un precio de $ 99.00 anuales, para con la ayuda de la consola de *Expo*, crear el paquete (en inglés, *Bundle*), con el comando ***eas build -p ios --auto-submit,*** en la figura se puede ver la ejecución del comando.

Figura 25

Creación del paquete de la aplicación



Nota. La figura muestra la consola de expo-cli para el empaquetado de una aplicación iOS.

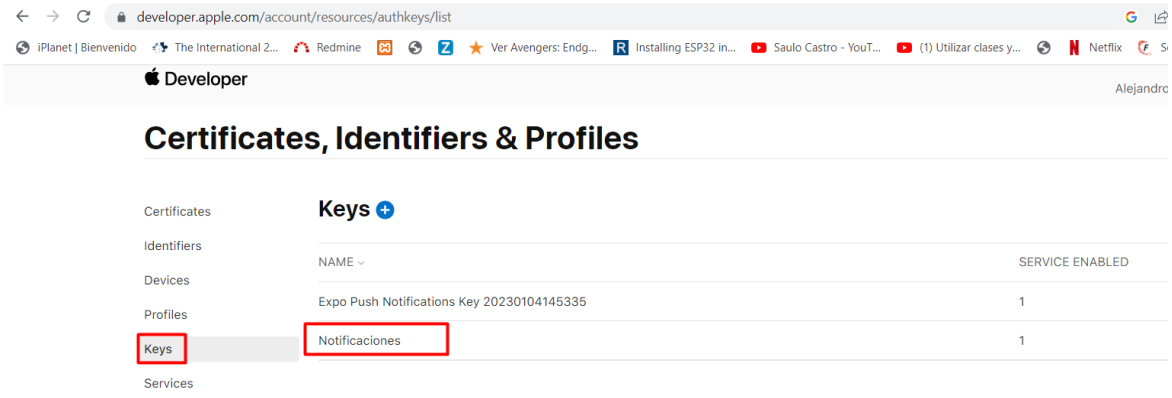
Los parámetros necesarios para poder logearse al servicio de *APNs* son:

- APNS_KEY_ID: Generado desde la consola de Apple. https://developer.apple.com/account/resources/ como se muestra en la Figura 26
- APNS_AUTH_KEY: Es un archivo de extensión p8 que se descarga desde la consola el panel de desarrollador de Apple. https://developer.apple.com/account/resources/
- TEAM_ID: Creado al pagar la suscripción.

- BUNDLE_ID: Creado al ejecutar el comando de empaquetado ***eas build -p ios --auto-submit***

Figura 26

Apple developer console



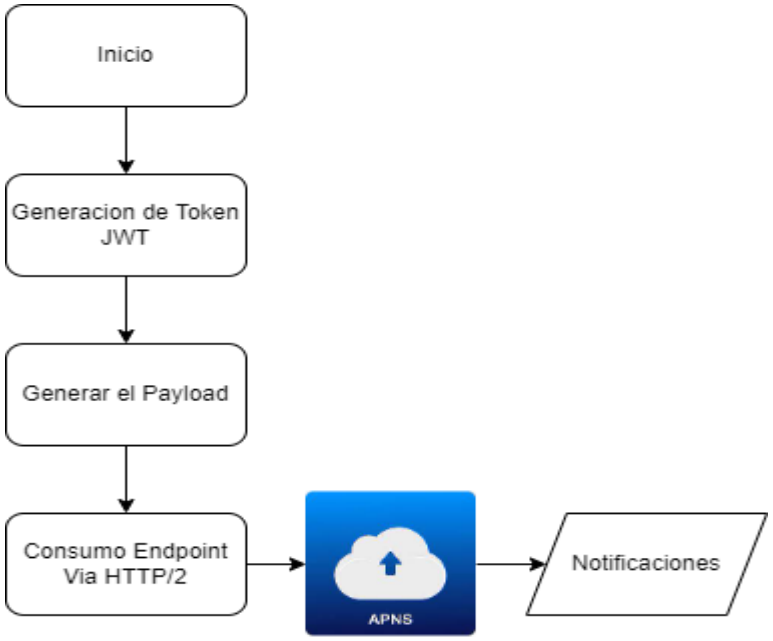
Nota. La figura muestra el panel de administración de Apple Developer donde se puede obtener la llave para notificaciones.

Login y envió de notificaciones en el servicio de APNs

En la siguiente Figura 27 se puede observar el diagrama de flujo para poder enviar notificaciones a un cliente *iOS*.

Figura 27

Diagrama de Bloques del Servicio de APNs



Nota. La figura muestra el diagrama de bloques del proceso de enviar notificaciones a un terminal iOS.

Se debe realizar una petición *HTTP/2* generando el token *JWT* de autenticación con las variables antes mencionadas, y enviando el payload con el mensaje de notificación a ***api.push.apple.com:443***, como llave se tiene el archivo con extensión *p8* descargado del panel de desarrollador de Apple. En la siguiente Figura 28 se muestra el fragmento de código utilizado para generar el token de autenticación y para con el mismo token generar una solicitud para notificación.

La librería *hyper* depende de Python 3.8.10 para funcionar correctamente para versiones posteriores de Python la librería *hyper* está deprecada.

Figura 28

Código usado para enviar notificaciones mediante *APNs*

```
import jwt
import time
import json

from hyper import HTTPConnection
ALGORITHM = 'ES256'

APNS_KEY_ID = ‘XXXXXXXX’
APNS_AUTH_KEY = ‘XXXXXXXXXX.p8'
TEAM_ID = 'XXXXXXXXXX'
BUNDLE_ID = ‘XXXXXXXXXXX’


# envia notificaciones, recibe el token para autenticación, el id del #dispositivo guardado en la base de datos, y el cuerpo de la notificacion

def sendMessageIphone(token, deviceId, msg):
 REGISTRATION_ID = deviceId
 path = "/3/device/" + format(REGISTRATION_ID)

 request_headers = {
 'apns-expiration': '0',
 'apns-priority': '10',
 'apns-push-type': 'alert',
 'apns-topic': BUNDLE_ID,
 'authorization': 'bearer ' + token
 }




 payload_data = {
 'aps': {
 'alert': {
 "title": "Sistema de Alerta Temprana",
 "body": msg
 }
 }
 }
 payload = json.dumps(payload_data).encode('utf-8')
 conn = HTTPConnection('api.push.apple.com:443')
 conn.request(
```

```
        'POST',
        path,
        payload,
        headers=request_headers
        )
        resp = conn.get_response()

        # print(resp.status)
        # print(resp.read())
        # devuelve el token para autenticarse

    def getToken():
        f = open(APNS_AUTH_KEY)
        secret = f.read()
        token = jwt.encode(
        {
        'iss': TEAM_ID,
        'iat': time.time() #timestamp
        },
        secret,
        algorithm=ALGORITHM,
        headers={
        'alg': ALGORITHM,
        'kid': APNS_KEY_ID,
        }
        )
        return token
```

Nota. La figura muestra el código utilizado para autenticarse al servicio de APNs y luego con el token JWT se consume otro endpoint para poder enviar notificaciones.

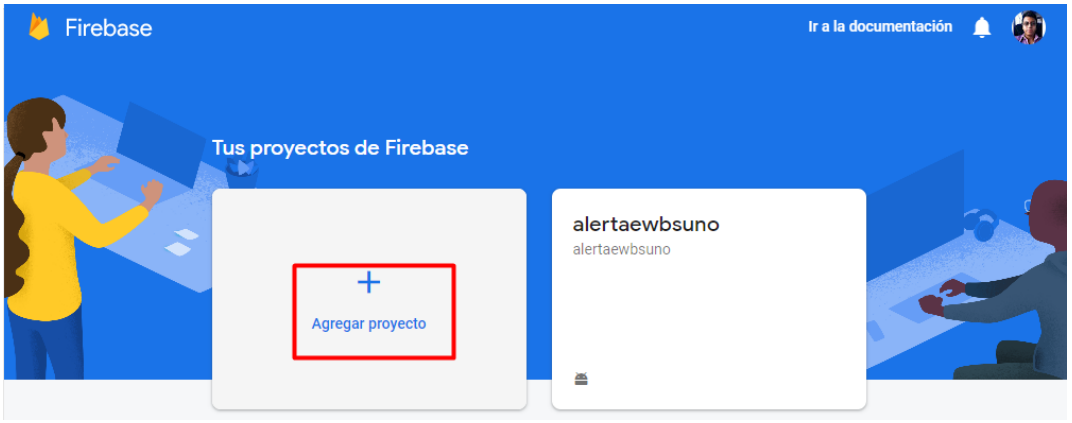
FCM – Android

Para poder enviar notificaciones a clientes Android se debe crear un proyecto (ver Figura 29), en la consola de *Firebase Google Cloud* en la siguiente url:

https://console.firebase.google.com/u/0/

Figura 29

Interfaz de Firebase Google Cloud

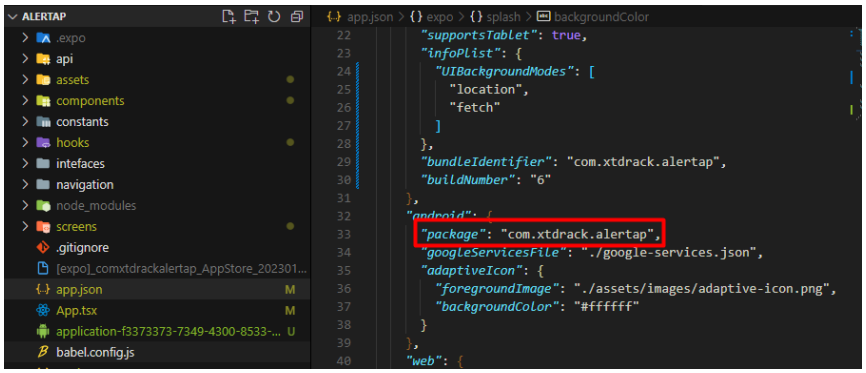


Nota. La figura muestra la interfaz del portal de Google Cloud para poder crear un proyecto que posteriormente recibirá notificaciones.

Creado el proyecto agregamos una app al proyecto con el identificador del nombre del paquete o *bundle id* inicializado al crear la aplicación, este último lo encontramos en el archivo "app.json" (ver Figura 30), en el código de la app de expo, en la Figura 31 se puede ver el formulario que hay que rellenar para poder registrar la App.

Figura 30

Estructura del código React Native expo, archivo "app.json"



Nota. La figura muestra la estructura de un proyecto en React Native con expo-cli y además el archivo de configuración app.json de donde se obtiene el package identificador de la aplicación para Google.

Figura 31

Formulario para registrar la aplicación

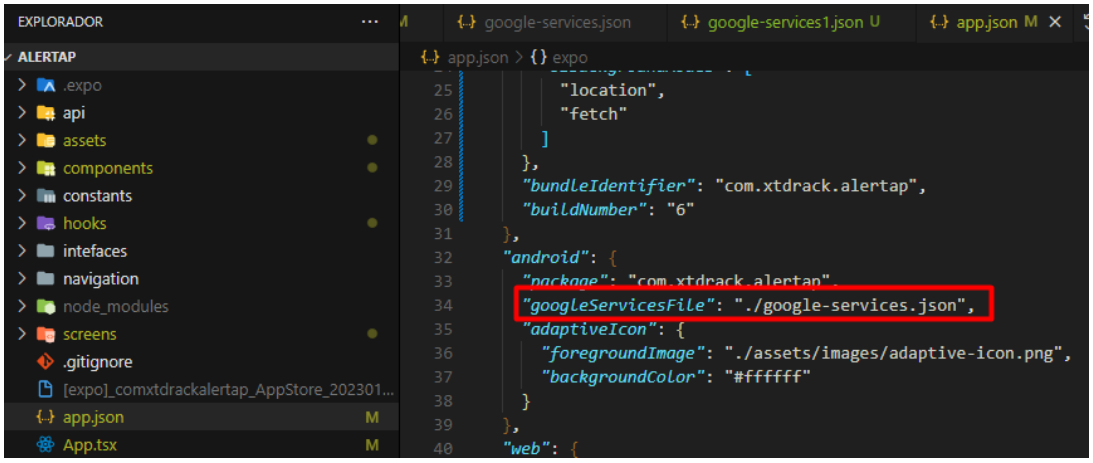
Nota. La figura muestra el formulario de registro de la aplicación en un proyecto de Google Cloud.

Se descarga el archivo *google-services.json* que contiene claves y demás configuraciones para la aplicación, se lo guarda en la raíz del proyecto de la aplicación adicionando la referencia al archivo en el fichero de configuración *app.json* como se muestra en la Figura 32.

.

Figura 32

App.json referencia a google-services.json en su estructura

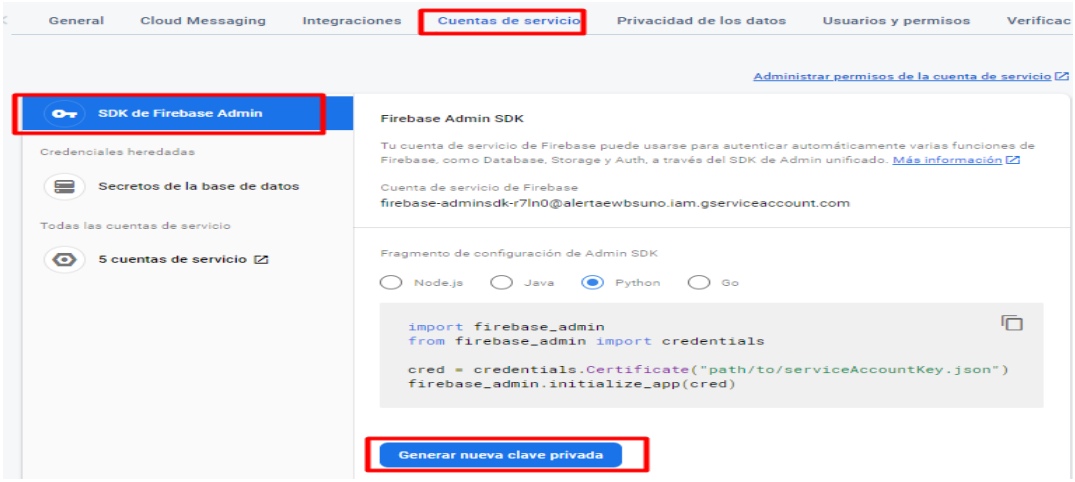


Nota. La figura muestra la configuración necesaria en el archivo app.json para apuntar correctamente a los servicios de FCM.

Una vez realizado el procedimiento anterior la App está lista para recibir notificaciones, ahora se verá la configuración del SDK (en inglés, *Software Development Kit*), para poder enviar notificaciones, se necesita una llave privada que se descarga de la consola de Google como se muestra en la Figura 33 y el código el proporcionado por la consola de Google como se muestra en la Figura 34.

Figura 33

Google console - cuentas de servicio - SDK de firebase admin



Nota. La figura muestra la generación de la llave privada para el SDK de FCM para Python y poder enviar mensajes a los terminales registrados.

A continuación, se muestra el código para enviar notificaciones para clientes Android.

Figura 34

Código que enviar notificaciones para usuarios Android

```
import firebase_admin
from firebase_admin import credentials, messaging
#credenciales para firebase admin
cred = credentials.Certificate(
 "./alertaewbsuno-firebase-adminsdk-r7ln0-74f0d94c18.json")
firebase_admin.initialize_app(cred)

#envia notificaciones
#title Titulo del mensaje
#msg Cuerpo del mensaje
#registration_token Arreglo de String con los token
def sendPush(title, msg, registration_token, dataObject=None):

 message = messaging.MulticastMessage(
 notification=messaging.Notification(
 title=title,
 body=msg
 ),
 data=dataObject,
 tokens=registration_token,
 )


 response = messaging.send_multicast(message)
 print('Successfully sent message:', response)
```

Nota. La figura muestra el fragmento de código proporcionado por FCM con un SDK para Python y el envío de notificaciones push.

Se intentó realizar con una tarea en segundo plano sin tener éxito sobre el control de cada cuanto segundo se ejecutaba dicha tarea, en iOS no se ejecutaba al estar en segundo plano y en Android se ejecutaba esporádicamente, según la documentación de Android el intervalo tiempo mínimo es de 15 minutos.

A continuación, se abordará el cómputo en la nube, y como desplegar la aplicación en un Droplet de Digital Ocean.

Cloud Computing

Digital Ocean

Servicio de computación en la nube, popular en los últimos años debido a la facilidad de implementación de prototipos de servidor de bajo costo y con alcance global.

Además de contar con variedad de *Droplets*, sistemas operativos, servicios de aplicaciones (en inglés, A*pp Services*) y precios *Digital Ocean* ofrece un saldo inicial de $100 para desarrolladores con el registro de una tarjeta de crédito, se entiende que hay otras nubes de cómputo como Google Cloud, AWS, Azure, pero debido a costos y para un periodo de pruebas se escogió *Digital Ocean*.

Droplet

Máquina virtual con IP pública ver Tabla 2 formateada que contiene sistema operativo Ubuntu 20.04 LTS para usarla como servidor, aquí si instalaron librerías, programas y paquetes utilizados para el funcionamiento del Gateway Server en la Tabla 3, se puede observar las características del *Droplet*.

Tabla 2

Dominio e IP pública del servidor

Dominio	IP
alertaewbs.site	143.244.157.58

Nota. Esta tabla muestra el dominio y la IP pública del Gateway Server.

Tabla 3

Características del Droplet

SO	vCore [Unidad]	RAM [GB]	SSD [GB]	Peticiones [Unidad]
Ubuntu 20.04 LTS	1	2 GB	200	500

Nota. Esta tabla muestra las características de la máquina virtual de Digital Ocean.

Configuración de Dominio

Agregar un Registro DNS según su servidor de DNS, de tipo A o AAAA dependiendo de la IP, si es IPv4 es de tipo A, si es de tipo IPv6 es tipo AAAA. Como se muestra en la Figura 35.

Figura 35

Interfaz de GoDaddy para la configuración de la IP

Nota. En la figura se muestra cómo se puede configurar la IP pública al dominio, se muestra la interfaz de GoDaddy para la configuración.

Puesta en marcha del servidor

En la siguiente Tabla 4 se muestra las versiones utilizadas en el despliegue del servidor

Tabla 4

Herramientas y versiones para el despliegue

Herramienta	Versión
Python	3.8.10
Django RF	3.13.1
MySQL	8.0.31
Nginx	1.18.0
Gunicorn	20.1.0

Nota. Esta tabla muestra las diferentes herramientas de software utilizadas para el despliegue del Gateway Server.

El código QR que se muestra en la Figura 36 redirige a un video explicativo del paso a paso desde crear un Droplet hasta hacer el despliegue de la aplicación, los comandos utilizados se encuentran en la Tabla 5

Figura 36

QR de la puesta en marcha del servidor

Nota. La figura muestra un código QR, el cual redirecciona a un video demostrativo del paso a paso de la puesta en marcha del Gateway Server.

Ingresar al servidor por un cliente SSH, el usuario por lo general es root en Digital Ocean y la contraseña dependerá de la forma de autenticación, en este caso es por una contraseña, pero también se puede usar una clave SSH.

Tabla 5

Comandos para instanciar el servidor

Comando	Acción
Instancia de base de datos Mysql	
# sudo apt update	Actualiza la lista de paquetes
# sudo apt install mysql-server	Instala MySQL
# sudo mysql_secure_installation	Configuración MySQL
# sudo mysql	Ingresar a la consola de MySQL
$ ALTER USER 'root'@'localhost' IDENTIFIED WITH caching_sha2_password BY 'password';	Cambiar el método de autenticación de MySQL
$ FLUSH PRIVILEGES;	Las tablas de privilegios vuelven a leer
$ create schema mensajes;	Crear un schema
$exit;	Salir de MySQL
Instalar Python 3.8	
# sudo apt update	Actualiza paquetes
# sudo apt install python3.8	Instala Python
# sudo apt install software-properties-common	Dependencias adicionales
# sudo add-apt-repository ppa:deadsnakes/ppa	Repositorio adicional
Instalar DjangoRestFramework	
# sudo apt update	Actualiza paquetes
# sudo apt install python3-django	Instalación de Django
# sudo apt install python3-pip	Instalación del gestor de dependencias de python
# sudo apt install python3-venv	Instalación del ambiente virtual
Configuración del Proyecto	
# cd /root	Ingresar a la ruta /root
# git clone https://github.com/jsalasx/gateway.git	Clonamos del repositorio el proyecto

Comando	Acción
# sudo python3 -m venv myenv	Creamos un ambiente virtual
# source myenv/bin/actívate	Activamos el ambiente virtual
# (myenv) pip install Django	Instalar Django
# (myenv) pip install djangorestframework	Instalar Django Rest Framework
# (myenv) pip install mysqlclient	Instalamos la librería para la conexión con MySQL
# (myenv) deactivate	Salir del ambiente virtual
Migración de Modelos	
python3 manage.py migrate	Migración de modelos a la base de datos
python3 manage.py createsuperuser	Creación de super usuario, utilizado para autenticarse en la aplicación
python3 manage.py runserver 0:8000	Poner en marcha el servidor en ambiente de pruebas
Despliegue en Producción	
# sudo apt update	Actualiza paquetes
# sudo apt install python-dev libpq-dev nginx	Instala dependencias y Nginx el Servidor Web
# source gateway/myenv/bin/actívate	Ingresar al ambiente virtual
(myenv) gateway/manage.py collectstatic	Generar archivos estaticos
(myenv) pip install django gunicorn	Instalar Gunicorn Servidor Web Especializado para Django
gunicorn --bind 0:8000 gateway.wsgi	Inicializacion Gunicorn
# sudo nano /etc/systemd/system/gunicorn.service	Creación del servicio de Linux

Comando	Acción
[Unit] Description=gunicorn daemon After=network.target [Service] User=root Group=root WorkingDirectory=/root/gateway/gateway/ ExecStart=/root/gateway/myenv/bin/gunicorn --access-logfile - --workers 3 --bind unix:/root/gateway/gateway/gateway.sock gateway.wsgi:application [Install] WantedBy=multi-user.target	Contenido del archivo gunicorn.service
	Gunicorn
# sudo systemctl start gunicorn	Iniciar el servicio Gunicorn
# sudo systemctl enable gunicorn	Habilitar el servicio de Gunicorn
# sudo nano /etc/nginx/sites-available/alertaewbs.site	Creación del archivo de configuración para Nginx

Comando	Acción

server {
listen 80;
server_name alertaewbs.site;
 	
location = /favicon.ico { access_log off;
log_not_found off; }
location /static/ {
root /root/gateway/gateway
}
 	
location / {
include proxy_params;
proxy_pass http://unix:/run/gunicorn.sock;}}
	Contenido del archivo alertaewbs.site

# sudo ln -s /etc/nginx/sites-available/alertaewbs /etc/nginx/sites-enabled	Creación el link simbólico
# sudo nginx -t	Verifica la configuración de Nginx
# sudo systemctl restart nginx	Reinicia el servicio de Nginx
# sudo ufw allow 80	Habilita el puerto 80 al firewall
# sudo ufw allow 443	Habilita el puerto 443 al firewall
Instalación del certificado SSL	
# sudo apt install certbot python3-certbot-nginx	Installation de certbot (Lets Encrypt)
# sudo certbot --nginx -d alertaewbs.site	Creación del certificado para el dominio alertaewbs.site

Nota. Esta tabla muestra el paso a paso de los comandos utilizados para instalar las dependencias y utilitarios para desplegar el servidor por consola en un Droplet de Digital Ocean

En la aplicación móvil y en la aplicación de escritorio existe un parámetro que es la URL de consulta que apunta a la IP o dominio público en este caso al dominio https://alertaewbs.site/.

Capítulo IV

Resultados

Escenario de Prueba

Se realizaron las pruebas en el Laboratorio de Televisión Digital de la Universidad de las Fuerzas Armadas – ESPE como se muestra en la Figura 38, con un resultado satisfactorio (ver Figura 37), el código QR anexo muestra el funcionamiento en el laboratorio.

Figura 37

Video demostrativo ambiente real en el Laboratorio de Televisión Digital UFA - ESPE

Nota. La figura muestra un código QR que redirecciona a un video demostrativo del funcionamiento del Sistema de Televisión Digital, con el envío de mensajes de alerta de emergencia.

Se recibieron correctamente las notificaciones en los ambientes de escritorio y móvil se detalla a continuación la lista de dispositivos usados en el escenario de prueba:

Transmisor

- Computador: Tiene una versión del software StreamXpress que soporta la carga de archivos con los datos de video, audio y el mensaje de alerta codificado TS (en inglés, *Transport Stream*).

- Modulador: Tarjeta DeKtec DTA-211 es un periférico USB del ordenador que le permite a través del software StreamXpress transmitir la señal al aire, trabaja en el rango de VHF a UHF, soportando estándares ISDB-Tb.
- Antena: Utilizada para transmitir la señal de salida.

Receptor

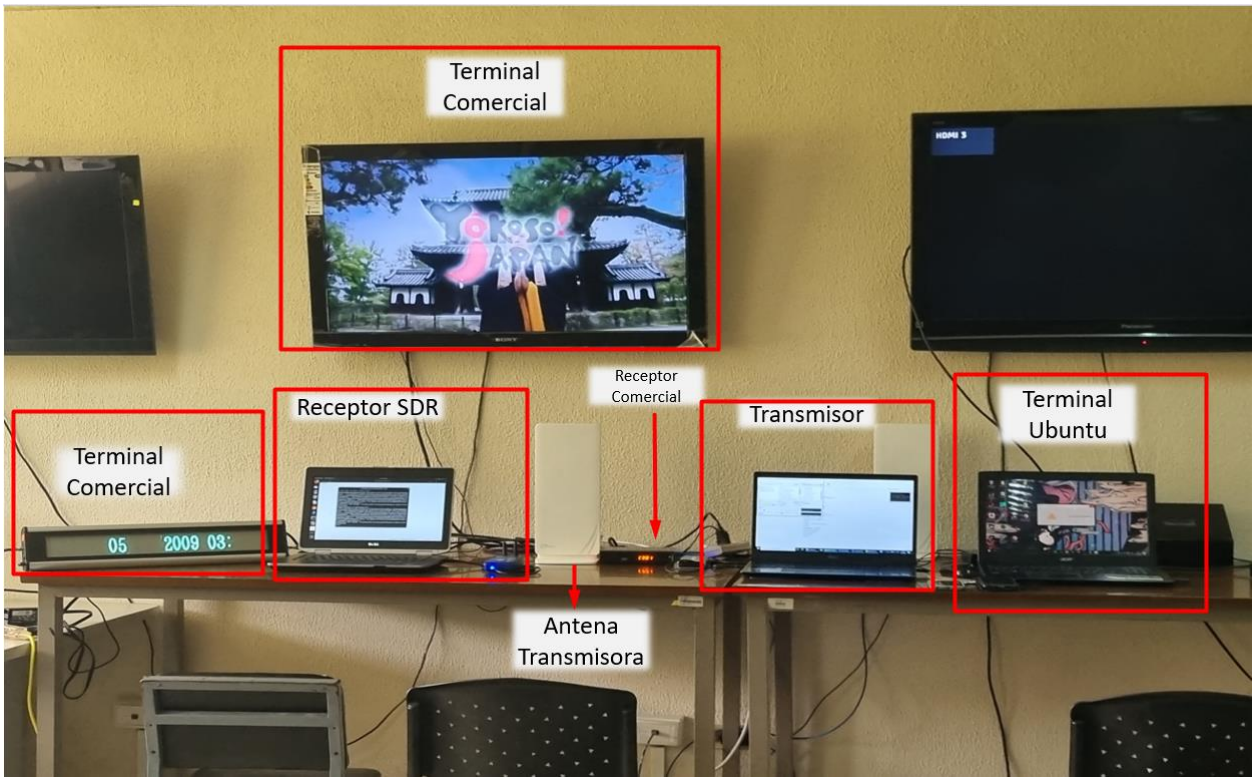
- Ordenador: Encargado del procesamiento de la señal, utilizando algoritmos de programación obtiene el mensaje y lo guarda en una ruta específica.
- SDR Adalm Pluto: Un dispositivo que permite procesar señales en el rango de UHF compatible con el estándar ISDB-Tb.

Clientes finales

- Windows: Ordenador Dell i7, 16 de ram, con Sistema operativo nativo Windows 10 arquitectura 64 bits.
- Linux: Ordenador Dell i7 16 de ram, máquina virtual con sistema operativo Ubuntu 20.04.
- iOS: iPhone 6S con iOS 14.8.1.
- Android: Samsung S21 FE con Android 12.

Figura 38

Escenario de prueba real en el laboratorio de Televisión Digital de la Universidad de las Fuerzas Armadas – ESPE, con sus respectivos elementos



Nota. La figura muestra los elementos que conforman el Sistema de Televisión Digital del laborío de la Universidad de las Fuerzas Armadas – ESPE, en color rojo están sus respectivos identificadores.

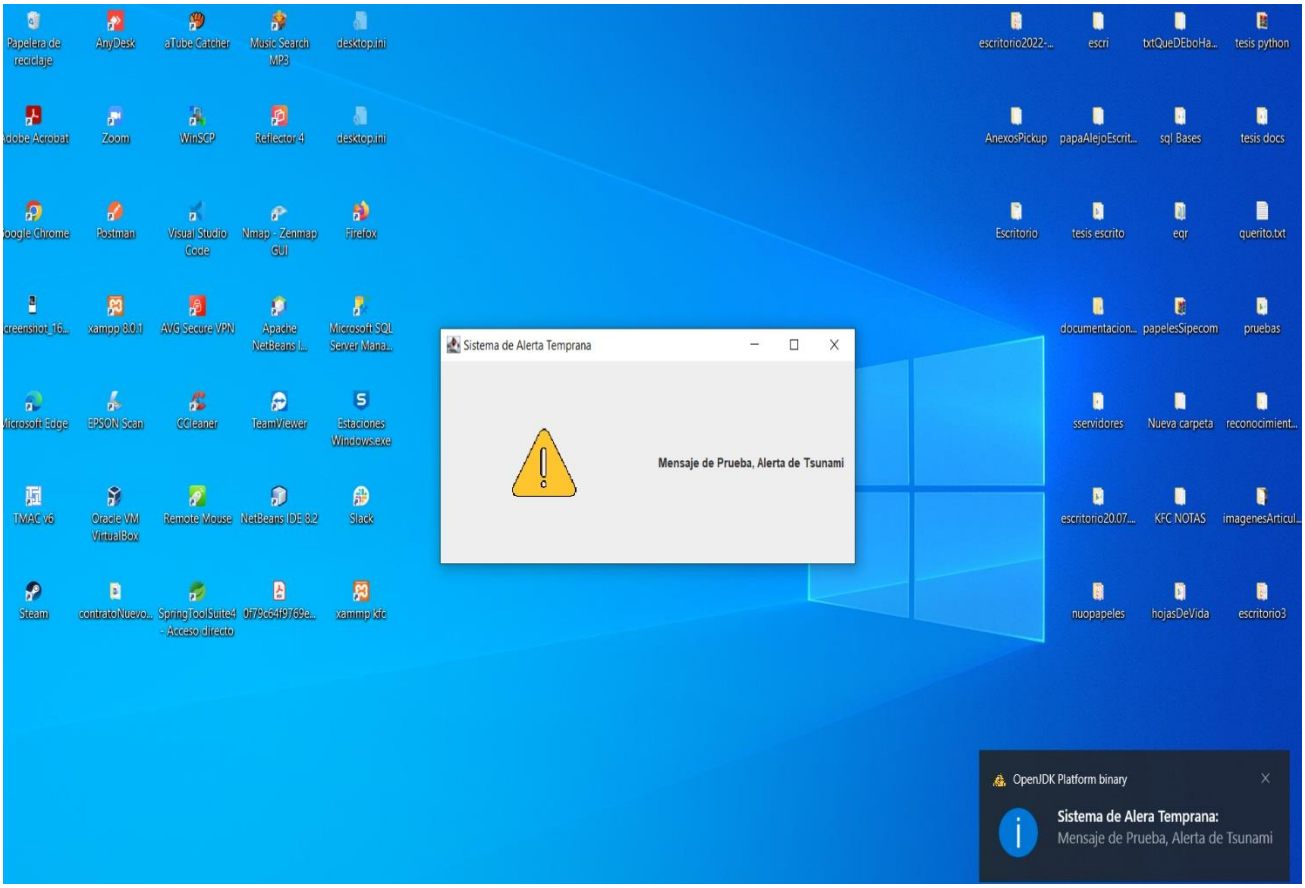
A continuación, se profundizará en el diseño de las notificaciones y las interfaces de usuario para los diferentes sistemas operativos.

Ambiente de Escritorio

Se utilizó una ventana para alertar del mensaje llamada JFrame de Java en el centro de la pantalla ordenador de Windows y el sistema de notificaciones de Windows (ver Figura 39), se adjunta el manual de instalación en los anexos, una vez la aplicación es instalada se deberá reiniciar el ordenador, cada que el ordenador inicie la aplicación se ejecutará en segundo plano en el puerto 28080.

Figura 39

Notificación en un ambiente de escritorio Windows 10



Nota. La figura muestra el funcionamiento y la recepción de un mensaje de alerta temprana en ambiente Windows.

Funcionamiento Escritorio

Escaneando el siguiente código QR (ver Figura 40), se puede observar el funcionamiento de la aplicación de escritorio en Windows y en la Figura 41 se puede observar el funcionamiento en Linux (Ubuntu 20.04 LTS).

Figura 40

Video demostrativo ambiente Windows

Nota. La figura muestra un código QR que redirecciona a un video demostrativo del funcionamiento en ambiente Windows.

Figura 41

Video demostrativo ambiente Linux

Nota. La figura muestra un código QR que redirecciona a un video demostrativo del funcionamiento en ambiente Linux.

Ambiente Móvil

Se presentarán las interfaces móviles que fueron desarrolladas para que el usuario final pueda interactuar con las aplicaciones.

Android

Se desarrollaron dos Tabs Navigation que son dos pantallas de navegación, la primera muestra una lista de mensaje de alerta temprana en una vista tipo carrusel y la segunda contiene información sobre el proyecto (ver Figura 42), y además la opción de descargar el audio de la alarma para personalizarlo.

Figura 42

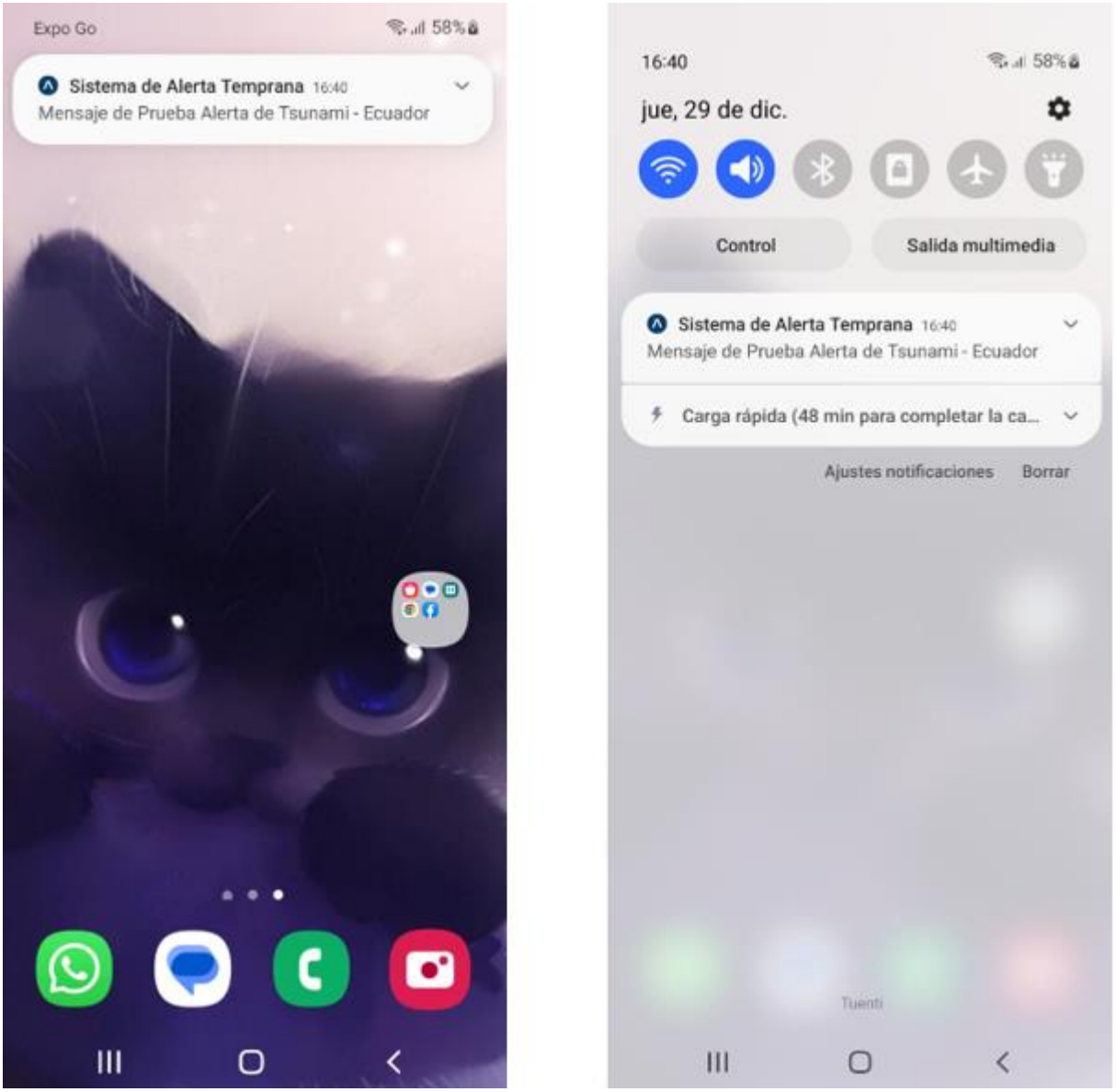
Interfaz Android

Nota. La figura muestra la pantalla principal llamada mensajes que contiene la información de la alerta renderizada en un componente carrusel y la pantalla secundaria se muestra información del proyecto.

Se puede observar en la Figura 43 las notificaciones, cuando un Mensaje de Alerta Temprana es recibido en los clientes móviles y muestra la notificación en la pantalla principal del cliente en el centro de notificaciones de Android.

Figura 43

Notificaciones Android



Nota. La figura muestra el funcionamiento de la recepción de mensajes de alerta temprana en ambiente Android.

Funcionamiento de Aplicación Android

El siguiente código QR representado por la Figura 44 muestra un video de cómo funciona el ambiente móvil en la recepción de mensajes de alerta.

Figura 44

Video demostrativo ambiente móvil Android

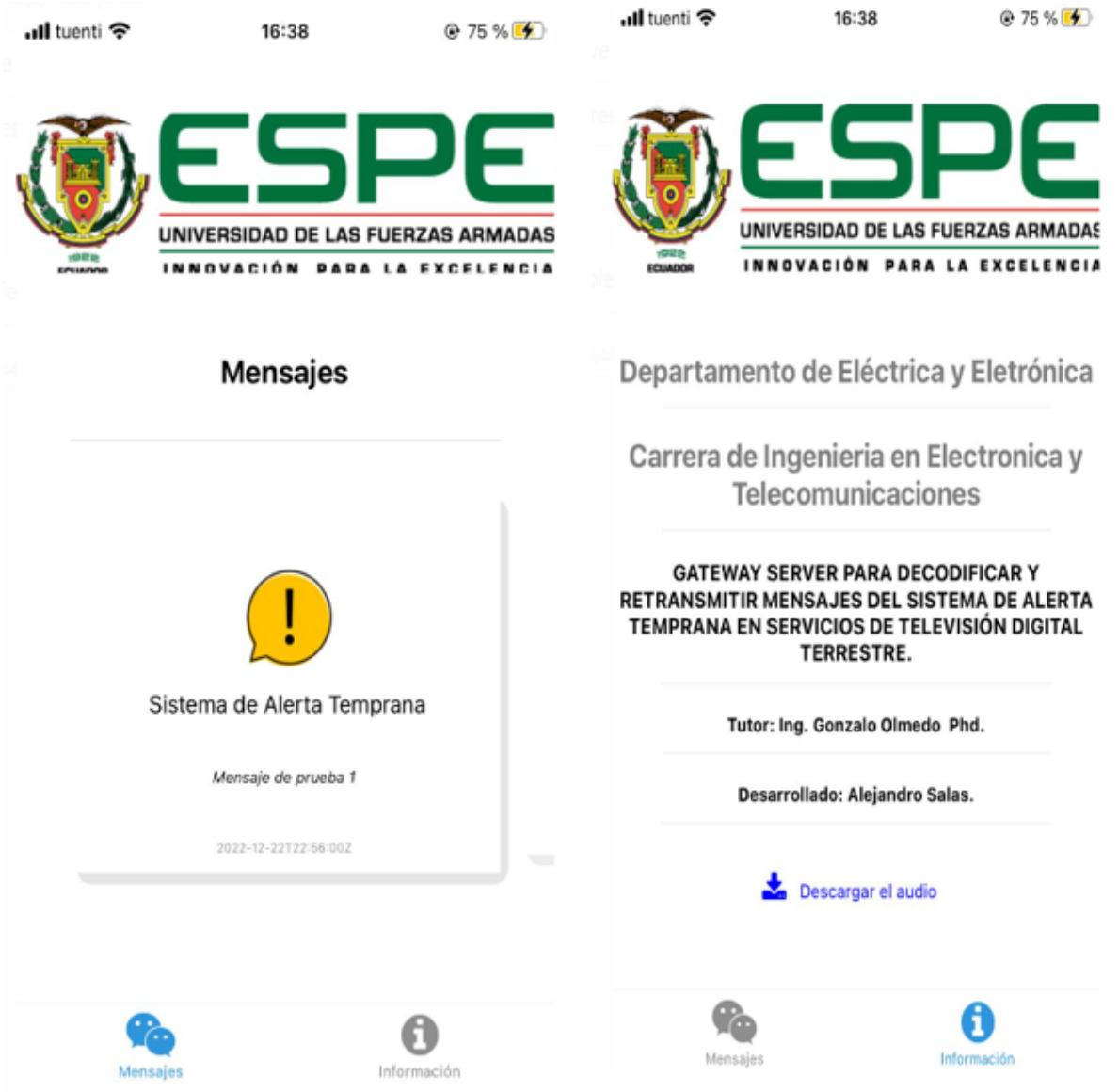
Nota. La figura muestra un código QR que redirecciona a un video demostrativo del funcionamiento en ambiente Android.

iOS

De igual manera que en Android y por la propiedad de React Native con expo de ser multiplataforma se muestran interfaces muy parecidas, dos Tabs Navigation, el primero de Mensajes que contiene la lista de Mensajes de Alerta Temprana y la segunda que muestra información del proyecto (ver Figura 45).

Figura 45

Interfaz iOS

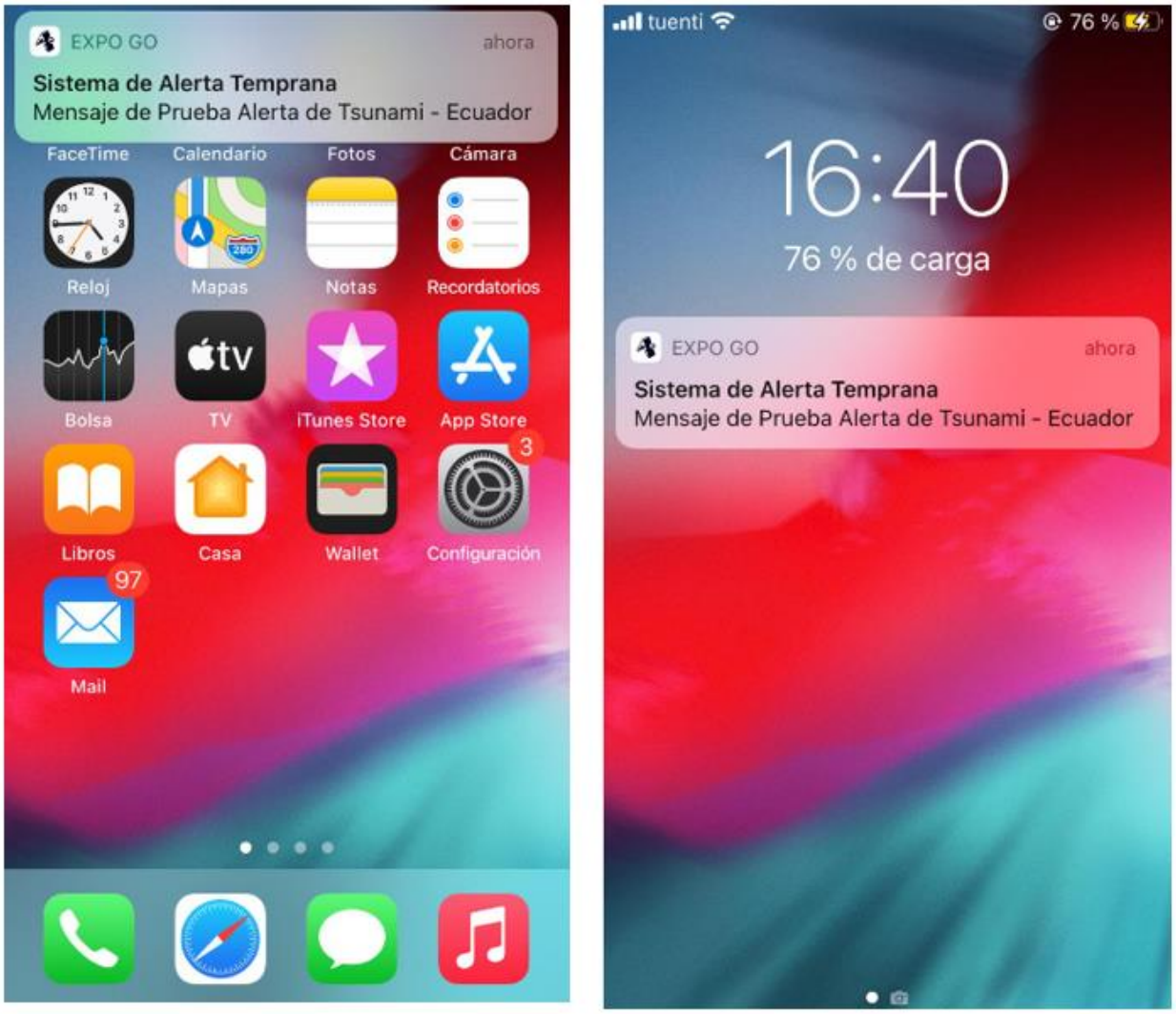


Nota. La figura muestra la interfaz de la aplicación en ambiente iOS.

Las notificaciones se muestran a continuación (ver Figura 46), como recibe un cliente iOS desde la pantalla principal del teléfono y de igual manera con el centro de notificaciones.

Figura 46

Notificaciones iOS



Nota. La figura muestra la recepción de mensajes de alerta temprana en ambiente iOS.

Funcionamiento en iOS

La siguiente Figura 47 con código QR muestra un video del funcionamiento de la aplicación móvil en iOS, y la notificación de mensajes.

Figura 47

Video demostrativo ambiente iOS

Nota. La figura muestra un código QR que redirecciona a un video demostrativo del funcionamiento en ambiente iOS.

Pruebas de Estrés

Para pruebas de Estrés del sistema y el servidor se utilizó la herramienta *Apache JMeter* con la cual se realizó varias peticiones *HTTP* de tipo *POST* y se observó el comportamiento del servidor ante estos escenarios se muestra en la tabla los resultados obtenidos, se realizaron pruebas entre 100 hasta 1000 peticiones, se realizaron 10 pruebas en cada escenario y se obtuvieron promedios con la finalidad de entregar datos lo más cercano al comportamiento real del servidor ver Tabla 6.

Este escenario es válido para clientes de escritorio que se conectan directamente al servidor, los clientes móviles tercerizan el servicio de Notificaciones utilizando *APNs* o *FCM Google* para iOS y Android respectivamente, con lo cual el consumo de recursos del servidor es mínimo para clientes móviles, con respecto a la documentación de *FCM* se pueden enviar lotes ilimitados con una longitud de 1000 dispositivos a la vez, mientras que *APNs* por su particularidad de enviar una a una. Las aplicaciones dependen de la velocidad de procesamiento del computador donde está alojado el script y de la velocidad de conexión a Internet.

Tabla 6

Pruebas de estrés con Apache JMeter

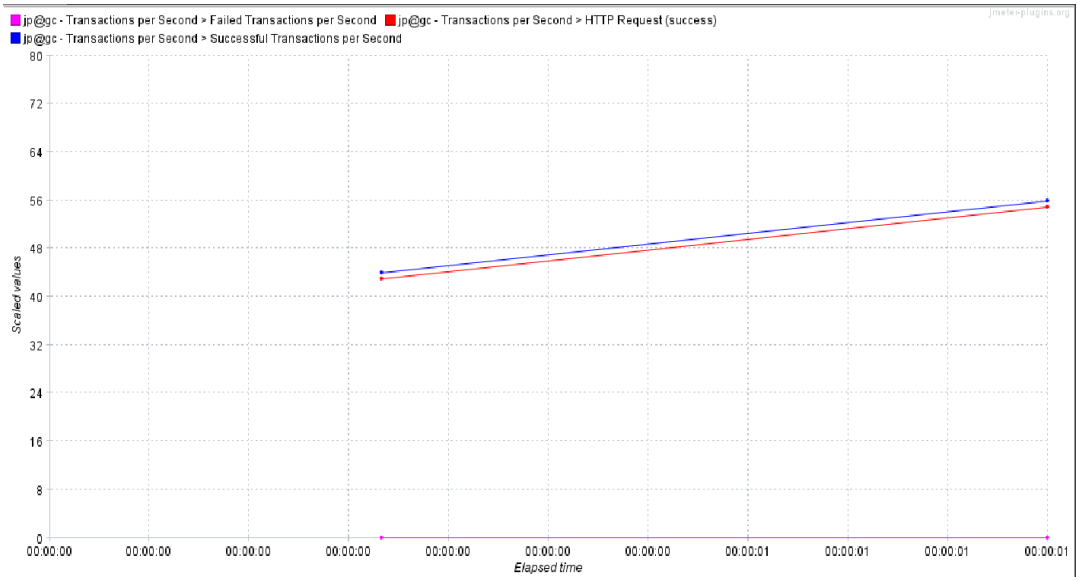
Número de peticiones	Porcentaje de error [%]	Tiempo [s]
100	0	1
200	0	3
300	0	4
400	0	5
500	3.2	7
600	10.67	9
700	35,86	8
800	44,75	8
900	46,44	7
1000	45,4	11

Nota. Esta tabla muestra el resumen de las pruebas de estrés del Gateway Server con la herramienta Apache JMeter.

Se puede observar en la siguiente Figura 48 de 100 Muestras que no existen errores todas las transacciones se completaron correctamente.

Figura 48

Comportamiento del servidor Apache JMeter con 100 muestras

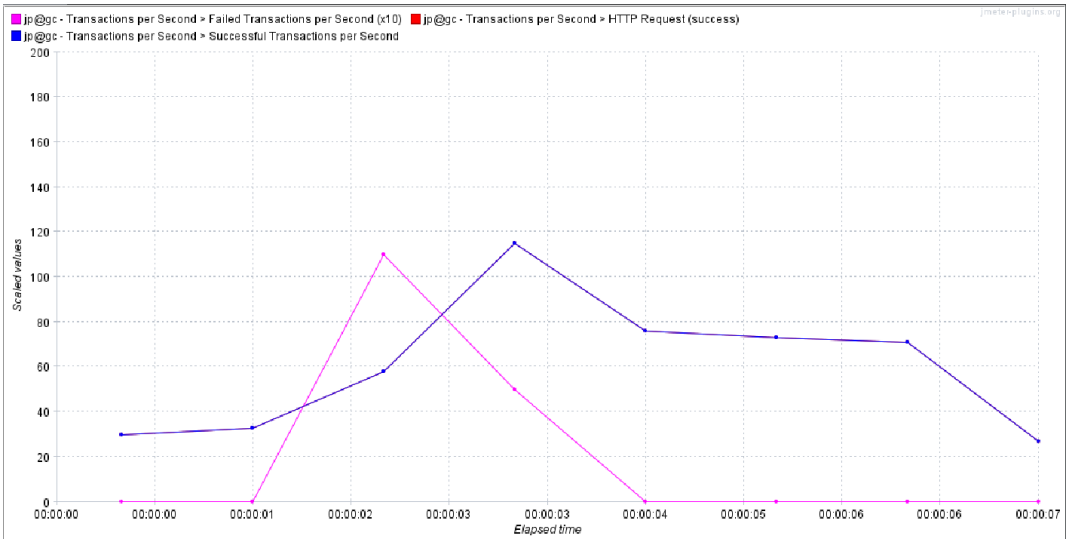


Nota. La figura muestra el comportamiento del servidor con la inyección de 100 peticiones.

Se puede observar en la siguiente Figura 49 con 500 Muestras que existen errores entre el segundo 1 y el segundo 4 de la inyección existen alrededor de 120 peticiones fallidas.

Figura 49

Comportamiento del servidor Apache JMeter con 500 muestras

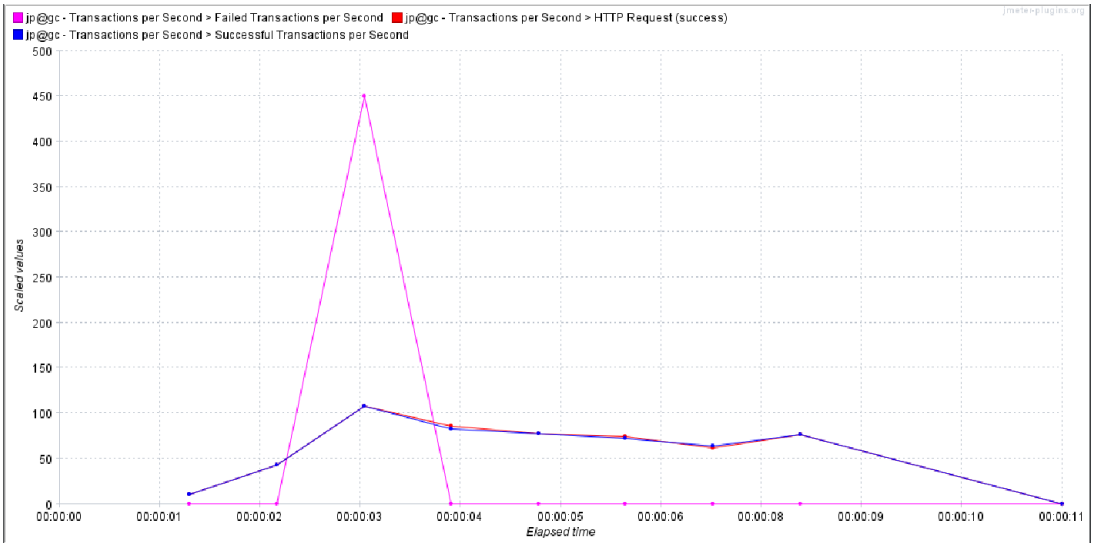


Nota. La figura muestra el comportamiento del servidor con la inyección de 500 peticiones.

Se puede observar en la siguiente Figura 50 con 1000 muestras que existen errores entre el segundo 2 y el segundo 4 de la inyección existen aproximadamente de 450 peticiones fallidas.

Figura 50

Comportamiento del servidor Apache JMeter con 1000 muestras



Nota. La figura muestra el comportamiento del servidor con la inyección de 1000 peticiones.

Conclusiones

Se desarrolló el estado del arte sobre el funcionamiento un Gateway Server para poder retransmitir a dispositivos móviles y computadores el mensaje de alerta temprana.

Se desarrollo y desplego un servidor en la nube como Gateway Server, se desarrolló una aplicación de escritorio multiplataforma (Windows, Linux) en el lenguaje de programación Java y una aplicación móvil multiplataforma (iOS, Android) para recibir los mensajes de alerta temprana, garantizando que se reciba la notificación y el usuario se percate de inmediato de la alerta siempre y cuando el cliente esté conectado a Internet, si no lo está recibirá la notificación unos segundos después de tener la conexión.

Se utilizaron las librerías de desarrollo de JavaScript para interfaces React Native y E*xpo Cli* debido la facilidad de crear con el mismo código aplicaciones para ambas plataformas o sistemas operativos como iOS (iPhone) y Android entre otros, además de la ventaja de no necesitar un ordenador MAC para codificar la aplicación en iOS.

El rendimiento y uso de recursos de una aplicación es mayor cuando se codificada bajo su lenguaje nativo Android - Java o iOS – SWIFT, el peso de la aplicación disminuye considerablemente cuando es codificada en lenguajes nativos, sin embargo, mediante el uso de librerías que se compilan nativamente es posible desarrollar más rápida y eficientemente la interfaces y funcionalidad de una aplicación.

Se logró integrar el sistema de transmisión de televisión digital terrestre y difundir satisfactoriamente el mensaje de alerta temprana, mediante las pruebas de estrés se garantiza 500 usuarios de escritorio concurrentes con un servidor muy modesto y haciendo referencia al número de usuarios móviles no existe un límite de terminales en iOS y en Android se pueden enviar 1000 notificaciones por bloque de manera ilimitada.

Recomendaciones

Se recomienda el uso de React Native con explo cli y otras librerías y/o frameworks de programación si su aplicación no requiere demasiado configuración nativa o módulos nativos propios de iOS y Android, ya que aceleran mucho la programación de interfaces con muchos plugins o dependencias listas para el uso y por el gran soporte que tienen estas tecnologías por los usuarios.

Se recomienda el uso de Cloud Computing debido al precio y al alcance global, que, de otra manera sería más costoso, además de ser amigable y tener documentación de cómo hacer los despliegues en la propia plataforma, facilitando así el poner en producción las aplicaciones Web.

Se recomienda seguir buenas prácticas, código limpio (en inglés, C*lean Code*), y utilizar un gestor de versiones para que el código sea reutilizable cualquier persona pueda entenderlo, modificarlo y añadir nuevas funciones, además de siempre asegurar la aplicación a nivel de protocolo con un certificado SSL y a nivel de autenticación con JWT o algún método de autenticación moderno que garantice la veracidad de los datos.

Trabajos Futuros

Para trabajos futuros se espera el uso de Docker y Kubernetes para montar sobre un contenedor el servicio de aplicación, aplicar una balanceador de carga, utilizar base de datos para caché y orquestar la aplicación para soportar mayor cantidad de usuarios y optimizar recursos.

Se espera la obtener la geolocalización del usuario para poder notificar solo a personas dentro del área de afectación del desastre natural, además de la integración con otros proyectos de grado para obtener rutas seguras de evacuación y una guía de que hacer cuando sucede determinado fenómeno.

Bibliografía

Alonso, I. (2019). *¿Qué es un WSGI?.* Medium. https://medium.com/@nachoad/que-es-wsgi-be7359c6e001

Analog Devices. (2017). *Adalm Pluto SDR Active Learning Modulo* https://www.analog.com/media/en/news-marketing-collateral/product-highlight/ADALM-PLUTO-Product-Highlight.pdf

AndroidStudio. (2023). *Introducción a Android Studio* https://developer.android.com/studio/intro?hl=es-419

Castillo, R. y Olmedo, G. (2021). *Implementación de un sistema de recepción de una señal de emergencia EWBS utilizando la Plataforma de Radio Definida por Software para el estándar ISDB-Tb* [Tesis de Ingeniería, Universidad de las Fuerzas Armadas - ESPE]. Repositorio Institucional de la Universidad de las Fuerzas Armadas ESPE. http://repositorio.espe.edu.ec/bitstream/21000/25711/1/T-ESPE-044717.pdf

de Souza, I. (2020, 17 de marzo). *API rest: conoce la importancia de ese recurso para el desempeño de una página web*. Rockcontent: https://rockcontent.com/es/blog/api-rest/

Ecu911. (2018). *¿Cómo funciona el Sistema de Alerta Temprana ante un Tsunami?* https://www.ecu911.gob.ec/sat-tsunami/

Fernández, Y. (2019, 30 de octubre). *Qué es Github y qué es lo que le ofrece a los desarrolladores.* Xataca Basics. https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores

Flores, M., Castillo, L., Martínez, F., y Cherez, S. (2018). Análisis Técnico del Sistema de Transmisión de Televisión*,* 1-5. https://dx.doi.org/10.31095/investigatio.2019.12.5

Fuentes, M., Cardena, N., y Gómez, D. (2012). *Evaluación de prestaciones (rendimiento e interferencias) del estándar de Televisión Digital Terrestre ISDB-Tb, mediante simulaciones y mediciones.* [Tesis de Ingeniería Universidad Politécnica de Valencia]. RuiNet Repositorio.

https://riunet.upv.es/bitstream/handle/10251/33245/Fuentes%20Muela.pdf?sequence=1&isAllowed=y

Granja, N. y Olmedo, G. (2011). *Análisis del Transport Stream para el estándar de televisión digital ISDB-TB* [Tesis de Ingeniería Escuela Politécnica del Ejército - ESPE]. Repositorio Institucional de la Universidad de las Fuerzas Armadas ESPE. http://repositorio.espe.edu.ec/xmlui/bitstream/handle/21000/4553/T-ESPE-032690.pdf?sequence=3&isAllowed=y

Gustavo, B. (2018, 22 de junio). *Dominios explicados para principiantes.* Hostinger. https://www.hostinger.es/tutoriales/que-es-un-dominio-web#%C2%BFQue_es_un_dominio

Gustavo, B. (2022, 22 de noviembre). *¿Qué es SSL, TLS y HTTPS?.* Hostinger. https://www.hostinger.es/tutoriales/ssl-tls-https

INEC, I. N. (2019). *Tecnologías de la Información y Comunicación.* https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Sociales/TIC/2019/201912_Principales_resultados_Multiproposito_TIC.pdf

inLab (2016, 7 de julio). *Django API REST*. https://inlab.fib.upc.edu/es/blog/django-api-rest

Ionos. (2022, 11 de noviembre). *Servidor DNS*. https://www.ionos.es/digitalguide/servidores/know-how/que-es-el-servidor-dns-y-como-funciona/

Java. (2022). *¿Qué es Java y por qué lo necesito?* https://www.java.com/es/download/help/whatis_java.html#:~:text=Java%20es%20una%20plataforma%20inform%C3%A1tica,crean%20muchos%20servicios%20y%20aplicaciones

Jiménez, A. (2019, 18 de junio). *React Native*. OpenWebinars https://openwebinars.net/blog/react-native-que-es-para-que-sirve/

Kinsta. (2022, 21 de febrero). *¿Qué Es Nginx y Cómo Funciona?.* https://kinsta.com/es/base-de-conocimiento/que-es-nginx/

López, L.(2020, 17 de enero). *Qué es Json Web Token y cómo funciona.* OpenWebinars. https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/

Matheus, C. (2020, 22 de abril). *Digital Ocean: qué es, cómo usar, ventajas y desventajas.* Rockcontent https://rockcontent.com/es/blog/digital-ocean/

Medina, R. (2022, 6 de septiembre). *Estadísticas de la situación Digital en Ecuador 2021-2022.* Branch. https://branch.com.co/marketing-digital/estadisticas-de-la-situacion-digital-en-ecuador-2021-2022/#:~:text=Ecuador%20es%20un%20pa%C3%ADs%20con,usuarios%20conectados%20diariamente%20a%20Internet.

Mozilla Developers. (2022). *¿Qué es JavaScript?* https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

Olmedo, G., Mena, R., Acosta, F. y Paredes, N. (2015). Transport Stream Generator and Player for Digital Terrestrial Television ISDB-TB*. ResearchGate*, 1-6. http://repositorio.espe.edu.ec/handle/21000/19257

Oñate, L., Gómez, J. y Lapo, H. (2018). Análisis y comparación de ISDB-T utilizando modulaciones OFDM y CC-OFDCM. *Ingenius,* 1-7. https://doi.org/https://doi.org/10.17163/ings.n21.2019.07

Recalde, C. y Olmedo, G.(2016). *Análisis y Generación del Flujo de Transporte con Sobreimposición de Texto para Alerta Temprana en Servicios de Televisión Digital Terrestre* [Tesis de Ingeniería Universidad de las Fuerzas Armadas - ESPE]. Repositorio Institucional de la Universidad de las Fuerzas Armadas ESPE. http://repositorio.espe.edu.ec/bitstream/21000/12248/1/T-ESPE-053500.pdf

RedHat. (2016). *Que son los servicios WEB.* https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Robledano, A. (2019, 24 de septiembre). *Qué es MySQL: Características y ventajas.* OpenWebinars. https://openwebinars.net/blog/que-es-mysql/

Robledeano, Á. (2019, 23 de septiembre). *Qué es Python: Características, evolución y futuro*. OpemWebinars: https://openwebinars.net/blog/que-es-python/

Romero, J. (2023). *Introducción a Django Rest Framework.* Axiacore https://axiacore.com/blog/introduccion-a-django-rest-framework-460/

Sambrano, Y. y Acosta, F.(2019). *Diseño e Implementación de una Interfaz que Adapte una Señal de Emergencia de Televisión Digital a la red de telefonía IP de la Universidad de las Fuerzas Armadas - ESPE* [Tesis de Ingeniería, Universidad de las Fuerzas Armadas - ESPE]. Repositorio Institucional de la Universidad de las Fuerzas Armadas ESPE. http://repositorio.espe.edu.ec/bitstream/21000/21274/1/T-ESPE-040865.pdf

Segura, A., Olmedo, G., Acosta, F. y Santillán, M. (2016). Designing a System for Monitoring and Broadcasting Early Warning Signs of Natural Disasters for Digital Terrestrial Television. */th IEEE Latin-American Conference on Communications (LATINCOM),* 1-6. https://doi.org/http://dx.doi.org/10.1109/LATINCOM.2015.7430119

Shogen, K., Ito, Y., Hamazumi, H., y Taguchi, M. (2006). Implementation of Emergency Warning Broadcasting System in the Asia Pacific Region. *ITU*, 8-10. https://www.itu.int/en/ITU-D/Emergency-Telecommunications/Documents/Thailand_2006/final1/Session%209/SESSION%209%20[NHK]%20Dr%20Kazuyoshi%20SHOGEN.pdf

Stackscale. (2020, 30 de abril). *Ubuntu 20.04 LTS: seguridad y rendimiento mejorados.* https://www.stackscale.com/es/blog/ubuntu-20-04-lts/

van der Meer, J. (1995). Digital Television Receiver Technology. *The Institution of Engineering and Technology,* 1-4. https://doi.org/10.1049/ic:19950883

Villamarin, D., Illescas, M., y Olmedo, G. (2011). *Implementación de un transmisor de pruebas de TV Digital terrestre ISDB-Tb para la emisión de aplicaciones interactivas* [Tesis de

Ingeniería, Escuela Politécnica del Ejercito - ESPE]. Repositorio Institucional de la Universidad de las Fuerzas Armadas ESPE. http://repositorio.espe.edu.ec/bitstream/21000/4289/1/T-ESPE-032586.pdf

Apéndices