

Sistema de Comunicaciones Basado en APRS y LoRa: Fundamentos, Legislación y Desarrollo de un Servidor de Monitoreo

Jose David Badilla Monge*, Josué Rojas González*
and Bernal Jesús Zamora Barrantes*

*Electronics Engineering School, Costa Rica Institute of Technology (ITCR), 30101 Cartago, Costa Rica,
{Josedbm201445,josuer1198, bxbz77 }@gmail.com

Abstract—Este documento presenta la fundamentación teórica, regulatoria y técnica necesaria para el desarrollo de un sistema de comunicaciones basado en APRS y LoRa, en el contexto de la legislación costarricense. Se analizan las características principales de ambas tecnologías, sus aplicaciones en sistemas de monitoreo y rastreo, y las normativas relevantes del espectro radioeléctrico nacional. Además, se describe el diseño e implementación de un servidor APRS utilizando la plataforma TrackDirect, integrando tecnologías como FastAPI, PostgreSQL, Docker y Leaflet.js para la visualización de datos en tiempo real.

Key words—APRS, LoRa, Comunicaciones Inalámbricas, IoT, Espectro Radioeléctrico, TrackDirect, FastAPI, Docker, PostgreSQL, Monitoreo en Tiempo Real

I. INTRODUCCIÓN

El avance de las tecnologías de comunicación inalámbrica ha permitido el desarrollo de soluciones eficientes para la transmisión de datos en tiempo real, incluso en entornos donde las infraestructuras tradicionales de telecomunicaciones presentan limitaciones. En este contexto, el sistema de comunicaciones Automatic Packet Reporting System (APRS) y la tecnología de modulación de largo alcance conocida como LoRa (Long Range) se han consolidado como alternativas robustas y versátiles para aplicaciones de monitoreo, rastreo y telemetría en redes distribuidas.

Este documento tiene como propósito ofrecer un análisis detallado sobre los principios técnicos que sustentan la operación de APRS y LoRa, así como su implementación bajo el marco normativo costarricense vigente. Para ello, se examinan las bases conceptuales de ambas tecnologías, su funcionamiento, los protocolos de comunicación utilizados y las bandas de frecuencia disponibles conforme al Cuadro Nacional de Atribución de Frecuencias (PNAF) y al Decreto N° 44010-MICITT.

Adicionalmente, se aborda el proceso de construcción de un sistema de monitoreo de datos APRS empleando herramientas contemporáneas de desarrollo de software. La propuesta se basa en la utilización de la plataforma de código abierto trackdirect, que integra tecnologías como Docker para la contenedorización de servicios, FastAPI para la creación de interfaces de programación de aplicaciones eficientes, y PostgreSQL como sistema de gestión de bases de datos. La combinación de estas herramientas no solo facilita la implementación

modular y escalable del servidor, sino que también promueve buenas prácticas de desarrollo y despliegue de infraestructuras resilientes.

La necesidad de comprender en profundidad tanto las bases técnicas como el entorno legal que rige las radiocomunicaciones en Costa Rica resulta fundamental para garantizar el cumplimiento de las regulaciones nacionales y la operatividad óptima de los sistemas diseñados. En este sentido, el presente trabajo pretende servir como guía integral para ingenieros, radioaficionados y desarrolladores interesados en la creación de redes de comunicación inalámbricas basadas en APRS y LoRa, contribuyendo así al fortalecimiento del ecosistema de tecnologías de información y comunicación en el país.

II. APRS

A. ¿Qué es APRS?

El **Automatic Packet Reporting System** (APRS) es un protocolo de comunicaciones digitales desarrollado por Bob Bruninga, WB4APR, en la década de 1990 [8]. Su propósito principal es el intercambio de información táctica en tiempo real a través de redes de radioaficionados. APRS integra técnicas de radio por paquetes para transmitir y recibir datos como:

- Coordenadas de localización geográfica (GPS).
- Información meteorológica.
- Mensajes de texto.
- Datos de telemetría de sistemas automatizados.

Una de las principales características de APRS es su arquitectura descentralizada: cada estación puede actuar como transmisor y receptor autónomo, generando redes resilientes ideales para situaciones de emergencia y monitoreo de variables en tiempo real.

B. Aplicaciones de APRS

APRS ha demostrado su utilidad en múltiples contextos, incluyendo:

- Rastreo de vehículos, embarcaciones y aeronaves.
- Monitoreo de estaciones meteorológicas remotas.
- Comunicación entre operadores de radioaficionados en tiempo real.

- Apoyo en operaciones de emergencia y rescate.
- Implementaciones educativas para el aprendizaje de comunicaciones digitales.
- Redes de sensores distribuidos en aplicaciones ambientales.

C. Protocolos de Comunicación Utilizados

El protocolo subyacente en APRS es **AX.25** [9], una adaptación del estándar X.25 orientado a redes de radioaficionados. AX.25 ofrece direccionamiento, detección de errores y control de flujo, optimizando la transmisión eficiente de pequeños paquetes de datos a través de medios de radio.

D. Bandas de Frecuencia Utilizadas

A nivel mundial, APRS opera principalmente en:

- 144.390 MHz (VHF) en América del Norte.
- 144.800 MHz (VHF) en Europa.
- 433.920 MHz (UHF) en algunas aplicaciones específicas.

En Costa Rica, las bandas autorizadas deben consultarse en el PNAF actualizado [1].

E. Componentes Clave de una Red APRS

La infraestructura típica de APRS incluye:

- **Transceptor de radio:** Para transmitir y recibir señales de RF.
- **TNC (Terminal Node Controller):** Modem que convierte tramas digitales a audio compatible.
- **Receptor GPS:** Fuente de datos de localización en tiempo real.
- **Software APRS:** Programas como APRSISCE/32 o Xastir.
- **I-Gate:** Puente que conecta APRS local a la red APRS-IS global.
- **Digipeaters:** Repetidores que extienden el alcance de las transmisiones.

III. LoRa

A. ¿Qué es LoRa?

LoRa (Long Range) es una tecnología de modulación de espectro ensanchado basada en *chirps* (CSS, Chirp Spread Spectrum), desarrollada por Semtech Corporation [10]. Está diseñada para permitir comunicaciones inalámbricas de largo alcance, bajo consumo de energía y gran penetración en ambientes urbanos y rurales.

Esta tecnología ha sido ampliamente adoptada en aplicaciones del Internet de las Cosas (IoT), donde es crucial transmitir pequeñas cantidades de datos a través de distancias extensas, en condiciones de batería limitada y con mínimas infraestructuras de soporte.

Las principales ventajas de LoRa son:

- Alcance de hasta 15 km en zonas rurales abiertas.
- Operación en bandas de espectro no licenciado (ISM).
- Excelente eficiencia energética, con dispositivos que pueden operar por años con baterías estándar.
- Alta resistencia a interferencias y robustez en entornos ruidosos.

B. Frecuencias Utilizadas

LoRa opera en bandas ISM, las cuales están reservadas internacionalmente para usos industriales, científicos y médicos sin necesidad de licencia previa. Las frecuencias específicas varían según la región:

- 433 MHz: Región Asia y partes de Europa.
- 868 MHz: Europa.
- 915 MHz: América (incluyendo Costa Rica, según el PNAF [1]).

Se debe verificar siempre el reglamento nacional vigente para asegurar el uso adecuado de las frecuencias.

C. Aplicaciones de LoRa

Gracias a sus características, LoRa es ideal para múltiples sectores:

- **Monitoreo ambiental:** Sensores distribuidos para temperatura, humedad, calidad del aire, etc.
- **Medición inteligente:** Contadores de agua, electricidad y gas conectados.
- **Rastreo de activos:** Seguimiento de vehículos, maquinaria o ganado.
- **Agricultura de precisión:** Control de variables agrícolas en tiempo real.
- **Seguridad:** Sistemas de alarma y monitoreo perimetral.
- **Domótica:** Automatización de viviendas y edificios inteligentes.

La flexibilidad de LoRa permite diseñar soluciones económicas y escalables en numerosos campos de aplicación.

IV. LEGISLACIÓN COSTARRICENSE

El Cuadro Nacional de Atribución de Frecuencias (PNAF) [1], actualizado mediante el Decreto N° 44010-MICITT, publicado en el alcance N°99 a la Gaceta N°95 del 30 de mayo de 2023 [2], establece las condiciones técnicas y regulatorias específicas para cada banda de frecuencia. Este documento define las bandas de frecuencia disponibles en el país, sus aplicaciones, y las restricciones técnicas como la potencia isotrópica radiada equivalente (PIRE) y las clases de emisión permitidas.

En el caso de las bandas ISM (Industrial, Scientific and Medical), como las frecuencias utilizadas por LoRa (433 MHz, 868 MHz y 915 MHz), el PNAF establece que estas bandas están designadas para aplicaciones de radiocomunicaciones de corto alcance y uso libre, siempre que se respeten las restricciones de potencia y tipo de modulación [1]. Para operar en estas frecuencias no se requiere una licencia previa, pero sí el cumplimiento de normativas técnicas para evitar interferencias con otros servicios.

El Decreto N° 44010-MICITT, en su actualización más reciente, subraya la necesidad de respetar las especificaciones técnicas relacionadas con la clase de emisión, identificada por códigos que describen la naturaleza de la señal modulada, el tipo de información transmitida y el método de modulación [3]. En el contexto de LoRa y APRS, las clases de emisión más utilizadas son F1D (datos digitales) y G1D (datos digitales).

de transmisión en paquetes) [4]. La PIRE permitida varía según la banda, siendo típicamente de 14 dBm a 27 dBm para dispositivos LoRa, dependiendo de la región y la normativa local [5].

El alcance N°99 a la Gaceta N°95 especifica los permisos requeridos para sistemas de comunicación como LoRa y APRS, determinando las bandas de frecuencia disponibles y las potencias máximas autorizadas [2]. La consulta de este documento es esencial para asegurar la operación legal y eficiente de estas redes.

Importante: Aunque el uso de bandas ISM no requiere licencia, los dispositivos deben respetar las especificaciones de clase de emisión, ancho de banda, potencia máxima y evitar generar interferencias perjudiciales

V. PLATAFORMAS EN LÍNEA PARA MONITOREO DE APRS

El Automatic Packet Reporting System (APRS) es un sistema de comunicación digital utilizado principalmente en radioaficionados para el rastreo de posiciones, mensajería, telemetría y otras aplicaciones de datos en tiempo real. Para visualizar la información transmitida por estaciones APRS en el mundo, existen diversas plataformas en línea que ofrecen mapas interactivos y herramientas avanzadas de análisis. A continuación, se describen tres plataformas principales utilizadas para este propósito.

A. APRS.fi

APRS.fi es una de las plataformas más populares y completas para el monitoreo global de estaciones APRS en tiempo real. Su dirección web es:

<https://aprs.fi/#!lat=9.9333&lng=-84.0845>

Entre sus características principales destacan:

- **Mapa Interactivo:** Permite visualizar la ubicación de estaciones APRS en todo el mundo mediante datos obtenidos de la red APRS-IS.
- **Historial de Tramas:** Permite revisar registros históricos de paquetes APRS transmitidos por estaciones específicas.
- **Información Extendida:** Muestra datos detallados de cada estación, incluyendo posición, velocidad, altitud y mensajes transmitidos.
- **Soporte para APRS-IS y RF:** Puede mostrar estaciones tanto de la red APRS-IS como de gateways de RF.
- **Filtros Avanzados:** Permite seleccionar estaciones por tipo (vehículos, estaciones meteorológicas, satélites, etc.).
- **Integración con Google Maps:** Utiliza mapas de Google para facilitar la navegación.
- **API para Desarrollo:** Ofrece una API para integrar información APRS en otras aplicaciones.

B. APRS Direct

APRS Direct es una plataforma alternativa para visualizar datos APRS en un mapa interactivo con una interfaz optimizada para dispositivos móviles y computadoras. Su enlace es:

<https://aprsdirect.de/?center=9.9205,-84.0598&zoom=11>

Sus principales características incluyen:

- **Interfaz Moderna:** Diseñada para facilitar el acceso a datos APRS con una navegación simplificada.
- **Visualización en Tiempo Real:** Muestra estaciones activas en la red con datos actualizados constantemente.
- **Opciones de Personalización:** Permite ajustar capas de visualización y aplicar filtros específicos.
- **Enlaces a QRZ.com:** Proporciona información de radioaficionados registrada en QRZ.com cuando está disponible.
- **Compatibilidad con Dispositivos Móviles:** Optimizado para uso en teléfonos inteligentes y tablets.

C. APRS Map

APRS Map ofrece una visualización alternativa de estaciones APRS con un enfoque en accesibilidad y velocidad de carga. Se accede mediante:

<https://aprs-map.info/?center=9.9205,-84.0598&zoom=11>

Entre sus características más relevantes se encuentran:

- **Mapa Rápido y Ligero:** Diseñado para una carga eficiente con bajo consumo de recursos.
- **Soporte para Diversos Mapas Base:** Permite elegir entre múltiples tipos de mapas como OpenStreetMap, Google Maps y otros.
- **Integración con APRS-IS:** Obtiene datos en tiempo real de la red APRS-IS.
- **Vista de Datos en Tabla:** Además del mapa, ofrece una lista de estaciones y sus últimas transmisiones en formato tabular.
- **Modo Nocturno:** Incluye una opción para mejorar la visibilidad en condiciones de poca luz.

D. Comparación y Usos

Cada una de estas plataformas ofrece diferentes ventajas dependiendo de las necesidades del usuario:

- APRS.fi es ideal para análisis detallado y acceso a datos históricos.
- APRS Direct es una opción moderna y fácil de usar, con integración a QRZ.com.
- APRS Map es una alternativa ligera y rápida con múltiples opciones de mapas.

El uso de estas herramientas permite a los radioaficionados y entusiastas del APRS monitorear redes de comunicación digital, rastrear estaciones móviles y fijas, y analizar datos meteorológicos y de telemetría en tiempo real.

VI. CONSTRUCCIÓN DE UN SERVIDOR APRS BASADO EN TRACKDIRECT

Con el objetivo de visualizar en tiempo real los datos recolectados por estaciones APRS, se implementó un servidor web basado en el repositorio de código abierto TrackDirect (<https://github.com/qvarforth/trackdirect>). Esta plataforma

integra tecnologías modernas como Python, FastAPI, PostgreSQL, Leaflet.js y Docker, facilitando un despliegue modular y escalable del sistema de monitoreo.

A. Requisitos Previos

La instalación del servidor requiere disponer de un sistema operativo basado en Linux (preferiblemente Ubuntu o Debian) o Windows con WSL2 habilitado. Adicionalmente, es indispensable contar con Docker, Docker Compose, Python versión 3.10 o superior, PostgreSQL para almacenamiento de datos, Visual Studio Code como entorno de desarrollo, y Git para la gestión de versiones.

B. Instalación de Dependencias

El primer paso consiste en la instalación de Docker y Docker Compose, ejecutando los siguientes comandos en terminal:

```
sudo apt update && sudo apt install ...  
-y docker.io docker-compose  
sudo systemctl enable --now docker
```

Posteriormente, se verifica la correcta instalación mediante:

```
> docker --version  
> docker-compose --version
```

El código fuente del proyecto se clona desde el repositorio oficial ejecutando:

```
git clone https://github.com/qvarforth...  
/trackdirect.git  
cd trackdirect
```

C. Estructura del Proyecto

TrackDirect presenta una arquitectura modular distribuida en varios componentes: el directorio `api/` contiene la lógica de la API FastAPI; `database/` define la estructura de la base de datos PostgreSQL; `static/` almacena archivos estáticos de la interfaz web; mientras que `docker-compose.yml` y `Dockerfile` configuran los servicios en contenedores.

D. Configuración de la Base de Datos

Para almacenar la información proveniente de las estaciones APRS, se configura una base de datos PostgreSQL ejecutando las siguientes instrucciones:

```
sudo -u postgres psql  
CREATE DATABASE trackdirect;  
CREATE USER trackuser WITH ...  
PASSWORD 'password';  
GRANT ALL PRIVILEGES ON DATABASE ...  
trackdirect TO trackuser;
```

E. Despliegue del Servidor

La puesta en marcha del servidor se realiza mediante Docker Compose. Desde el directorio raíz del proyecto se ejecuta:

```
docker-compose up --build -d
```

Esta acción levanta tres contenedores: uno con FastAPI para la gestión de la API APRS, otro con PostgreSQL para la persistencia de datos, y un tercero que expone una interfaz web utilizando Leaflet.js. El estado de los contenedores puede verificarse mediante:

```
docker ps
```

F. Entorno de Desarrollo en Visual Studio Code

Se recomienda utilizar Visual Studio Code debido a su integración nativa con herramientas de desarrollo de contenedores y su soporte para Python. Funcionalidades relevantes incluyen la extensión Docker para gestionar contenedores visualmente, el soporte para debugging de APIs en Python y la integración con Git para control de versiones. El proyecto puede abrirse con:

```
code trackdirect
```

G. Pruebas y Verificación

Una vez desplegados los servicios, se accede a la documentación automática de la API a través de:

<http://localhost:8000/docs>

La detención de todos los servicios activos puede realizarse ejecutando:

```
docker-compose down
```

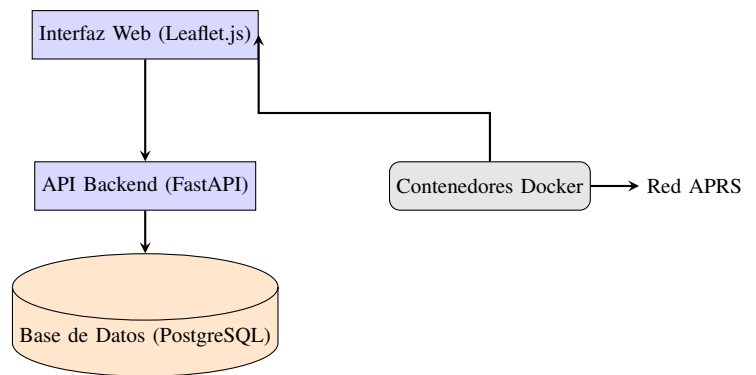


Figure 1: Arquitectura general del servidor APRS basado en TrackDirect.

VII. IMPLEMENTACIÓN DEL SERVIDOR APRS CON TRACKDIRECT

A. Arquitectura del Sistema

Este diagrama representa la estructura general del sistema TrackDirect. La arquitectura es modular, basada en contenedores Docker, lo que permite escalar, desplegar y mantener fácilmente cada componente (backend, base de datos, frontend y WebSocket). TrackDirect centraliza el procesamiento de tramas APRS, almacenamiento geoespacial y visualización en tiempo real a través de mapas interactivos.

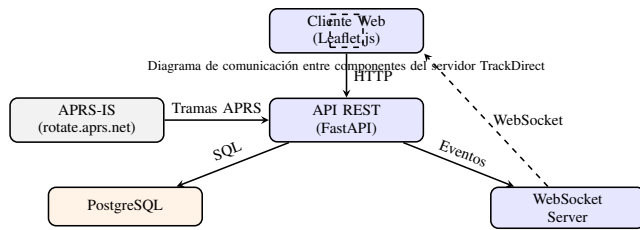


Figure 2: Arquitectura modular del sistema con integración Docker

B. Configuración y Solución de Problemas

El despliegue inicial del servidor TrackDirect se basa en una arquitectura contenedorizada con Docker, facilitando la portabilidad, escalabilidad y replicabilidad del entorno. A continuación, se presentan los comandos esenciales para la instalación:

Listing 1: Comandos clave para despliegue

```
# Clonar repositorio
git clone https://github.com/qvarforth/trackdirect.git
cd trackdirect

# Configurar variables de entorno
echo "POSTGRES_PASSWORD=clave_segura" > .env
echo "APRS_FILTER='r/9.5/-84.2/100'" >> .env

# Iniciar contenedores
docker compose up --build -d
```

- El repositorio oficial contiene la infraestructura de backend (FastAPI), base de datos (PostgreSQL), servidor WebSocket, frontend con Leaflet.js y configuraciones de Docker.
- El archivo `.env` se utiliza para gestionar variables sensibles y parámetros dinámicos como:
 - `POSTGRES_PASSWORD`: Define la contraseña del servicio de base de datos.
 - `APRS_FILTER`: Filtra las tramas APRS por región geográfica, en este caso, un radio de 100 km centrado en latitud 9.5, longitud -84.2. Esto reduce el tráfico innecesario, mejora el rendimiento y focaliza el monitoreo.
- El comando `docker compose up --build -d` crea y ejecuta los contenedores en segundo plano. El uso de `--build` garantiza que las imágenes estén actualizadas. Este enfoque se alinea con el paradigma de Infraestructura como Código (IaC), permitiendo la replicación exacta del entorno en diferentes máquinas.

El uso de variables de entorno mediante archivos como `.env` permite desacoplar los parámetros de configuración del código fuente de la aplicación. Esta práctica es ampliamente aceptada en entornos profesionales y se alinea con los principios propuestos en la metodología de desarrollo conocida como 12-Factor App. Al externalizar configuraciones sensibles, como credenciales de base de datos o filtros geográficos, se facilita la adaptabilidad del sistema entre distintos entornos, además de mejorar la seguridad y el mantenimiento.

Table I: Problemas comunes y soluciones

Síntoma	Solución
Mapa no carga	<p><code>leafletTilesUrl = 'https://{s}.tile.osm.org/{z}/{x}/{y}.png'</code></p> <ul style="list-style-type: none"> • Usar proveedor alternativo:
Datos no en tiempo real	<ul style="list-style-type: none"> • Verificar <code>aprsd.conf</code>: <pre>server rotate.aprs.net port 14580 filter \$APRS_FILTER</pre>
Alto uso de CPU en PostgreSQL	<ul style="list-style-type: none"> • Crear índice BRIN: <pre>CREATE INDEX idx_time ON positions USING BRIN(timestamp);</pre>
Fallos en WebSocket	<ul style="list-style-type: none"> • Verificar puertos en <code>docker-compose.yml</code>: <pre>- "8000:8000" (API) - "9001:9001" (WS)</pre>

En el contexto específico de TrackDirect, el parámetro `APRS_FILTER` permite establecer un área geográfica de interés mediante un filtro de tipo `r/lat/lon/radio`, conocido como círculo de contención o bounding circle. Esta técnica permite limitar el tráfico de tramas APRS recibidas, enfocando el procesamiento en una región específica. Al reducir el volumen de datos entrantes, se optimiza el uso de recursos computacionales y se mejora la eficiencia del almacenamiento, especialmente cuando se espera una alta densidad de estaciones.

El archivo `docker-compose.yml` permite declarar y orquestar todos los servicios necesarios para el funcionamiento del sistema, incluyendo el backend API, la base de datos PostgreSQL, el servidor WebSocket y el cliente web. Esta solución basada en Docker Compose representa una estrategia moderna y eficiente para la implementación de sistemas distribuidos, ya que permite configurar redes internas, volúmenes persistentes y enlaces entre contenedores con un solo comando. Este enfoque modular y reproducible es especialmente valioso en entornos académicos o de producción donde se requiere consistencia entre despliegues.

La visualización de los datos se realiza a través de Leaflet.js, una biblioteca JavaScript de código abierto que requiere la integración con un proveedor de tiles cartográficos. Algunos servicios como Mapbox o Esri requieren autenticación o suscripciones comerciales, lo que puede limitar su uso en entornos abiertos. Por esta razón, OpenStreetMap representa una alternativa viable y sostenible, ya que ofrece acceso libre bajo la licencia Open Database License (ODbL), compatible con proyectos comunitarios y educativos.

Desde el punto de vista de la transmisión de datos, APRS se basa en el protocolo AX.25, una adaptación de X.25 para entornos de radioaficionados. La conexión a la red APRS-IS mediante el servidor `rotate.aprs.net` en el puerto 14580 permite integrar datos en tiempo real desde una red global de estaciones. Este servidor actúa como balanceador

dinámico mediante DNS, redirigiendo las conexiones hacia nodos disponibles sin necesidad de configurar manualmente una dirección IP fija.

En cuanto al almacenamiento, PostgreSQL se utiliza como sistema gestor de base de datos relacional para registrar las tramas entrantes. A medida que el volumen de datos aumenta, especialmente con columnas temporales como `timestamp`, los índices convencionales tipo B-tree pueden volverse ineficientes. En este escenario, los índices BRIN (Block Range Index) ofrecen una solución más adecuada, ya que están optimizados para columnas que contienen datos secuenciales. Estos índices requieren menos espacio en disco, se construyen más rápidamente y, aunque menos precisos que los B-tree, resultan eficaces para consultas sobre series temporales.

Por último, la actualización en tiempo real de los datos hacia el cliente se logra mediante WebSocket, una tecnología que permite mantener una conexión persistente y bidireccional entre el navegador y el servidor. Esta conexión es fundamental para reflejar al instante la aparición o desplazamiento de estaciones en el mapa. Si los puertos necesarios no están correctamente expuestos en la configuración de Docker, el navegador no podrá establecer el canal de comunicación, interrumpiendo la funcionalidad dinámica del sistema.

Resumen operativo:

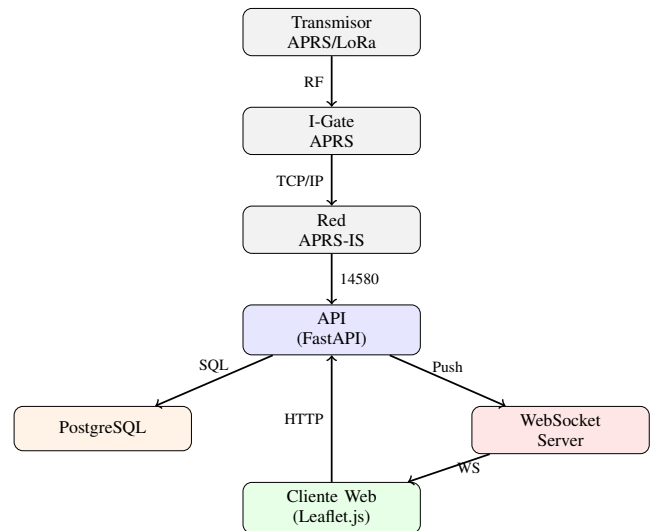
Cuando se presentan fallos en la visualización del mapa, la causa más probable es la restricción del proveedor de tiles utilizado. Una solución práctica es sustituir el proveedor por una fuente libre, como OpenStreetMap, que no impone límites de acceso ni requiere autenticación. En situaciones donde los datos no aparecen en tiempo real, es necesario verificar la conexión con la red APRS-IS y la correcta aplicación del filtro geográfico en el archivo `aprsd.conf`, ya que una configuración inadecuada puede bloquear la recepción de paquetes útiles.

En cuanto al rendimiento del sistema, si se detecta un uso elevado de CPU en PostgreSQL, la implementación de un índice BRIN sobre la columna de tiempo mejora de forma significativa la velocidad de las consultas sin comprometer la integridad de los datos. Este tipo de índice es especialmente recomendable en escenarios donde se almacenan grandes volúmenes de información con características temporales o secuenciales.

Finalmente, si el cliente web no recibe actualizaciones en tiempo real, es fundamental comprobar que los puertos necesarios para el funcionamiento de WebSocket se encuentren correctamente expuestos en el archivo de configuración de Docker Compose. Un error en esta etapa puede impedir el establecimiento de la conexión persistente, eliminando la capacidad del sistema para responder de forma inmediata a los eventos entrantes desde la red APRS.

C. Flujo de Datos APRS-IS

Este diagrama muestra cómo un paquete APRS fluye desde la red hasta su visualización: 1. Un transmisor APRS (por ejemplo, un tracker LoRa) emite una trama. 2. La trama es



Flujo completo de datos desde transmisor APRS hasta visualización web

Figure 3: Secuencia técnica del flujo de datos en TrackDirect

captada por servidores APRS-IS y enviada a nuestro backend. 3. FastAPI interpreta y almacena la posición en PostgreSQL. 4. A través de WebSocket, los datos son empujados al navegador. 5. Leaflet renderiza la información en tiempo real.

Personalización del Frontend

Listing 2: Personalización de mapas

```

// htdocs/js/map-config.js
const customLayers = {
  weatherStations: {
    name: "Est. Meteorológicas",
    filter: (station) =>
      station.type === 'weather',
    icon: 'icons/weather.png'
  },
  mobileStations: {
    name: "Vehículos en Movimiento",
    filter: (station) =>
      station.speed > 5,
    icon: 'icons/mobile.png'
  }
};
trackDirect.addCustomLayers(
  customLayers
);

```

TrackDirect permite capas personalizadas: - Puedes usar `'filter()'` para mostrar sólo ciertos tipos de estaciones (ej. meteorológicas). - Se pueden definir íconos propios para diferenciar tipos en el mapa. - La API JavaScript de TrackDirect permite extensiones como animaciones, trayectorias o agrupación.

Consideraciones de Rendimiento

El rendimiento de un sistema de monitoreo en tiempo real como TrackDirect depende de múltiples factores: eficiencia en la gestión del almacenamiento, configuración adecuada del motor de base de datos, y control del volumen de datos que ingresan al sistema desde fuentes externas. Cada uno de estos aspectos debe ser tratado con rigor técnico para garantizar la

escalabilidad y estabilidad del servidor bajo condiciones de uso prolongado y creciente carga operativa.

Desde el punto de vista del almacenamiento, una de las mejores prácticas en sistemas de adquisición de datos de alta frecuencia es establecer una política de retención temporal limitada. En este caso, se recomienda conservar únicamente las últimas 72 horas de información, eliminando automáticamente registros más antiguos mediante mecanismos programados (por ejemplo, tareas cron con sentencias `DELETE WHERE timestamp < NOW() - INTERVAL '72 hours'`). Esta estrategia se justifica tanto por razones de eficiencia como de pertinencia, ya que los datos APRS pierden valor operativo fuera de ventanas cortas de análisis. Además, limitar el tamaño de la base de datos evita la degradación progresiva del rendimiento de las consultas, especialmente en esquemas donde no se aplica particionamiento de tablas.

Respecto al ajuste de PostgreSQL, resulta fundamental dimensionar correctamente los parámetros de memoria y concurrencia, en función de la infraestructura disponible y la carga esperada. Una configuración inicial sugerida contempla los siguientes valores:

- **Almacenamiento:** Se recomienda limitar la retención de datos a 72 horas para evitar crecimiento descontrolado de la base.
- **Parámetros PostgreSQL:**

```
shared_buffers = 4GB
maintenance_work_mem = 1GB
work_mem = 64MB
```
- **Filtro geográfico:** Usar filtros de entrada APRS (como `'r/9.5/-84.2/100'`) limita el tráfico a un área definida y reduce carga.

El parámetro `shared_buffers` determina cuánta memoria se asigna para el caché interno de PostgreSQL. Un valor apropiado corresponde al 25–40% de la memoria RAM total del sistema, siendo 4 GB una referencia adecuada para servidores con 16 GB de RAM o más. Por su parte, `maintenance_work_mem` afecta el rendimiento de operaciones como creación de índices, `VACUUM` o `ANALYZE`, por lo que un valor de 1 GB permite acelerar tareas de mantenimiento en grandes volúmenes de datos. Finalmente, `work_mem` regula la cantidad de memoria disponible para operaciones intermedias como ordenamientos o agrupamientos durante la ejecución de consultas. Un valor de 64 MB representa un punto de equilibrio entre eficiencia y consumo de recursos, especialmente si el número de conexiones concurrentes no es elevado.

Otro aspecto crítico es la gestión del volumen de datos entrantes desde la red APRS-IS. A diferencia de un sistema que recibe datos exclusivamente desde dispositivos controlados, TrackDirect puede integrarse a una red global donde circulan cientos de miles de paquetes por hora. Para mitigar la carga innecesaria, se recomienda emplear filtros de entrada que restrinjan las tramas a una región geográfica de interés. Un ejemplo de este tipo de filtro es `r/9.5/-84.2/100`,

que indica al servidor que acepte únicamente paquetes cuya posición esté contenida dentro de un círculo de 100 kilómetros de radio, centrado en la latitud 9.5 y longitud -84.2. Este mecanismo, soportado por el protocolo APRS-IS, permite reducir drásticamente el ancho de banda utilizado, el número de inserciones en base de datos, y el esfuerzo computacional requerido para mantener actualizada la interfaz visual.

En conjunto, estas recomendaciones representan una estrategia integral de optimización. Limitar la persistencia temporal reduce la huella del almacenamiento, los parámetros afinados en PostgreSQL mejoran la respuesta del sistema bajo consultas concurrentes, y los filtros geográficos minimizan la carga procesada innecesariamente. La combinación de estas técnicas no solo favorece la sostenibilidad del sistema en el tiempo, sino que habilita su escalabilidad hacia despliegues más exigentes, tanto en volumen de estaciones como en duración de operación continua.

VIII. UTILIDAD DEL SISTEMA IMPLEMENTADO

En este caso, la utilidad que se le dará al servidor aplicado en el TEC para todas las sedes, es un sistema de trackeo de los buses (Fig. 4) y servicios de transporte asociados a la universidad. Este sistema será capaz de ver la ubicación de los servicios de transporte activos, su costo, vínculo para reservar o pagar su ticket de transporte así como horario según cada estación de parada.



Figure 4: Tracking bus

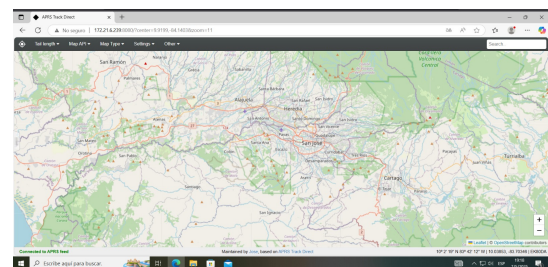


Figure 5: Mapa del servidor para visualización de dispositivos LoRa

A. Requisitos

Inicialmente, los usuarios podrán consultar esta información solamente conectado a las redes insitucionales a modo de prueba la visualización en computadora se verá tal y como se observa en la figura 5. El dimensionamiento para el servidor

debe ser capaz de dar atención simultánea a al menos 1000 personas, esto debido a que la población del TEC anualmente ronda los 10000 según los datos institucionales.

- 2020: 10120
- 2021: 10300
- 2022: 10300
- 2023: 10000
- 2024: 9750

Por lo tanto, a nivel de hardware y como caso extremo la configuración recomendada es de al menos 32 GB de RAM, un procesador de 8 núcleos y como mínimo un almacenamiento SSD de 256 GB. En cuanto al almacenamiento es importante tener en cuenta que se debe establecer un borrado de memoria al menos cada 3 horas para no saturar el espacio en memoria que se llenará debido a los datos descargados de los mapas consultados.

B. Costos

Para cumplir con estos requisitos investigados y definidos, se necesita presupuestar la adquisición de estos materiales. Es por esto, que se construye un BOM para el proyecto de modo que ayude a presupuestar el costo de los materiales, el cual es de \$1.536,89. Esta lista se encuentra en anexos 7.

Junto a esto, las horas invertidas en el desarrollo del proyecto se encuentran en anexos 6. Por lo que debido a las horas ingeniero invertidas junto al BOM, el costo total del proyecto correspondería a \$2130.44.

IX. RESULTADOS DE LA IMPLEMENTACIÓN DEL SERVIDOR APRS

La implementación del servidor APRS basado en TrackDirect se abordó en tres entornos distintos, con el objetivo de evaluar su despliegue y funcionalidad en diferentes configuraciones. A continuación, se detallan los resultados obtenidos para cada uno de estos escenarios.

A. Implementación en Máquina Virtual Remota (Servidor de la Escuela)

En el entorno de la máquina virtual proporcionada por la Escuela de Ingeniería Electrónica del TEC, se logró realizar la configuración y el montaje completo del servidor APRS (Fig.9). Se verificó que el sistema era capaz de establecer conexión con el servicio APRS-IS (Internet Service) y de recibir el flujo de datos global de APRS, lo que indica un correcto funcionamiento de la capa de comunicación y recepción de datos.

Sin embargo, a pesar de recibir el flujo de datos del APRS-IS, no fue posible observar la visualización de dispositivos o información en el mapa de la interfaz web de TrackDirect (Fig.5). Este inconveniente sugiere una posible falla en la interpretación, procesamiento o renderización de los datos recibidos por parte de la aplicación, o bien una configuración incompleta que impedía que los datos se tradujeran en representaciones gráficas en el frontend. Aunque el *feed* de datos era funcional, la ausencia de una representación visual efectiva limitó la utilidad del sistema en este entorno.

B. Implementación en Máquina Local (Docker Directo)

La implementación del servidor TrackDirect en la máquina local (equipo personal) se realizó utilizando Docker directamente sobre el sistema operativo anfitrión. En este escenario, se logró un despliegue exitoso de todos los contenedores Docker necesarios para el funcionamiento de la plataforma (base de datos, backend, frontend y el componente *aprsc*).

No obstante, se presentaron problemas de acceso y autenticación al servidor interno de *aprsc* (Fig.10). A pesar de que el contenedor *aprsc* se inició correctamente, la plataforma TrackDirect no pudo establecer una conexión o autenticarse adecuadamente con él, impidiendo el intercambio de datos entre el servidor APRS y los demás módulos de la aplicación. Este fallo se manifestó principalmente al intentar iniciar sesión o acceder a funcionalidades que dependían de esta conexión, resultando en un servidor parcialmente funcional donde el *frontend* y *backend* podían operar, pero sin la integración vital de los datos APRS.

C. Implementación en Máquina Virtual Local (VirtualBox)

El despliegue en una máquina virtual local configurada en VirtualBox resultó ser el más complejo y limitante (a pesar de que no debió ser así). Durante el proceso de ejecución de `docker-compose up`, la operación falló consistentemente (Fig.11) en la fase de montaje del contenedor asociado a *aprsc*.

Los errores reportados durante esta fase indicaban:

- Problemas de credenciales de acceso.
- Ausencia de respuesta del servidor interno de *aprsc*.
- Errores de clave GPG ('untrusted public key algorithm: dsa2048') al intentar configurar el repositorio de *aprsc* para la descarga de paquetes.
- Fallos de conectividad ('Connection refused') al intentar acceder al servidor remoto de repositorios de *aprsc* (`aprsc-dist.he.fi`), impidiendo la descarga de dependencias o claves esenciales para la construcción de la imagen *aprsc*.

Estos obstáculos combinados impidieron que el componente *aprsc* se construyera e iniciara correctamente dentro de su contenedor, lo que a su vez bloqueó el despliegue completo de la pila de servicios de TrackDirect. Consecuentemente, no se pudo establecer una funcionalidad operativa del servidor en este entorno.

D. Resumen de Desafíos

Los resultados obtenidos en las diferentes plataformas resaltan una serie de desafíos críticos: la sensibilidad de las configuraciones de *aprsc* y su integración con TrackDirect, las estrictas medidas de seguridad de las versiones modernas de Ubuntu con respecto a algoritmos de claves GPG obsoletos, y la dependencia de la disponibilidad de servidores de repositorios externos. Estos factores impidieron una implementación fully funcional en todos los escenarios evaluados.

X. ANÁLISIS DE RESULTADOS

El análisis de los resultados obtenidos en las diferentes fases de implementación del servidor APRS basado en TrackDirect revela patrones y causas subyacentes a los desafíos encontrados, ofreciendo lecciones valiosas sobre el despliegue de sistemas complejos y la gestión de dependencias.

A. Análisis de la Implementación en Máquina Virtual Remota

La capacidad de conectar con APRS-IS y recibir el flujo de datos global en el entorno de la VM de la escuela, a pesar de la falta de visualización en el mapa, sugiere que la comunicación de bajo nivel del componente `aprsc` era funcional. Los problemas de visualización en la interfaz de TrackDirect podrían atribuirse a diversas causas:

- **Falla en la Integración Backend-Frontend:** Es posible que los datos APRS se recibieran y almacenaran correctamente en la base de datos, pero el `*backend*` de TrackDirect no los estuviera procesando o sirviendo adecuadamente al `*frontend*` para su representación en el mapa. Esto podría deberse a configuraciones erróneas en las rutas de la API, errores en el código del `*backend*` al interactuar con la base de datos o fallas en la lógica de procesamiento de los paquetes APRS para extraer coordenadas válidas.
- **Problemas en el Renderizado del Frontend:** Alternativamente, el `*frontend*` podría haber tenido dificultades para interpretar los datos recibidos del `*backend*` o para interactuar con la API del mapa (ej., OpenStreetMap). Errores en JavaScript en el navegador, limitaciones de la red institucional para cargar recursos externos del mapa, o configuraciones incorrectas de las capas de visualización podrían ser los factores.
- **Filtrado de Datos:** Una configuración inadvertida en `aprsc` o TrackDirect podría estar filtrando los datos de posición o no enviándolos con el formato esperado para la visualización en el mapa.

Este escenario demostró una conectividad básica exitosa, pero puso de manifiesto la complejidad de la integración entre los distintos módulos de la aplicación para lograr una funcionalidad completa.

B. Análisis de la Implementación en Máquina Local

Los problemas de acceso y autenticación al servidor interno de `aprsc` en el despliegue local de Docker apuntan directamente a una configuración incorrecta de las credenciales o los parámetros de conexión entre los contenedores. TrackDirect requiere que sus componentes de `*backend*` y `*frontend*` se comuniquen con el servidor `aprsc` a través de una interfaz de red interna de Docker. Las causas más probables son:

- **Credenciales de Autenticación:** El servidor `aprsc` requiere un `*passcode*` y, posiblemente, otros parámetros de autenticación para permitir conexiones. Si estos no estaban configurados de manera idéntica en `aprsc.conf` y en las variables de entorno o archivos de configuración de los contenedores de TrackDirect, la conexión sería rechazada.

- **Nombre del Servicio o Puerto Incorrecto:** Aunque `docker-compose` facilita la comunicación entre servicios por sus nombres, un error tipográfico en la URL de conexión (ej., `http://aprsc:puerto`) o un puerto incorrecto en la configuración del `*backend*` impedirían el acceso.
- **Configuración de Red Docker:** Si bien `docker-compose` crea una red interna por defecto, configuraciones de red personalizadas o restricciones de firewall a nivel del sistema operativo anfitrión podrían haber interferido, aunque esto es menos probable si todos los contenedores estaban en la misma red de `docker-compose`.

Este resultado subraya la criticidad de la coherencia en la configuración entre contenedores en un entorno orquestado con Docker Compose.

C. Análisis de la Implementación en Máquina Virtual Local

El fracaso en el despliegue completo en la máquina virtual local fue el más instructivo, ya que reveló una combinación de problemas de dependencia externa y políticas de seguridad del sistema operativo:

- **Problema de la Clave GPG ('dsa2048'):** La incapacidad de `'apt'` para verificar la firma del repositorio de `aprsc` debido al uso del algoritmo DSA2048 (`'untrusted public key algorithm'`) en su clave GPG es un problema inherente a las versiones modernas de Ubuntu (22.04 LTS y 24.04 LTS). Estas versiones han endurecido sus políticas de seguridad, desaprobando algoritmos considerados obsoletos. Este obstáculo impidió la instalación del paquete `aprsc` a través de `'apt'`, afectando directamente la capacidad de construir la imagen Docker correspondiente. Esta situación resalta la importancia de la compatibilidad de versiones entre el sistema operativo base y las dependencias externas.
- **Problema de Conectividad ('Connection refused'):** La recurrencia del error `'Connection refused'` al intentar acceder al repositorio `aprsc-dist.he.fi` indica una interrupción del servicio o un bloqueo a nivel de red externa (posiblemente un firewall en el lado del servidor de `aprsc` o en la red del usuario). Este factor, siendo incontrolable desde el entorno de la VM, fue una barrera fundamental que impidió incluso la descarga de la clave GPG y, por ende, el acceso a los paquetes.
- **Dependencia en Cascada:** Los problemas de GPG y conectividad crearon una dependencia en cascada. Sin poder instalar el paquete `aprsc` o descargar sus recursos, la construcción de la imagen Docker falló, lo que imposibilitó el inicio de los servicios orquestados por `docker-compose`.

Este escenario particular demostró que, si bien Docker proporciona portabilidad y aislamiento, no exime de las dependencias externas (servidores de paquetes) ni de las políticas de seguridad subyacentes del sistema operativo anfitrión o de la imagen base del contenedor. La resolución de estos problemas

a menudo requiere enfoques alternativos, como la compilación manual del software.

XI. CONCLUSIONES

El desarrollo de sistemas de comunicaciones LoRa/APRS en Costa Rica, particularmente para aplicaciones de geolocalización y monitoreo de transporte como la propuesta para el TEC, representa una plataforma de gran potencial para la transmisión eficiente de datos en tiempo real. La comprensión integral de esta tecnología, sus aplicaciones específicas y el marco regulatorio nacional es crucial para asegurar no solo la eficiencia operativa, sino también el cumplimiento normativo. La sinergia entre LoRa y APRS permite, conceptualmente, la creación de redes híbridas con capacidades avanzadas de seguimiento y control remoto.

Sin embargo, la implementación práctica de un servidor APRS basado en TrackDirect, aunque conceptualmente prometedora, reveló una serie de desafíos significativos que impidieron un despliegue completamente funcional en los entornos evaluados. La infraestructura moderna basada en contenedores con Docker y la arquitectura modular que ofrece TrackDirect con, por ejemplo, FastAPI, facilita la portabilidad y la gestión. No obstante, la experiencia en este proyecto demostró que la efectividad de estas herramientas está intrínsecamente ligada a la compatibilidad de las dependencias externas y la robustez de las configuraciones específicas.

Los principales hallazgos y lecciones aprendidas de esta implementación fueron:

- **Desafíos en la Gestión de Repositorios y Claves GPG:** La integración del componente `aprsd` se vio obstaculizada por problemas recurrentes con su repositorio de paquetes, específicamente la incompatibilidad del algoritmo de la clave GPG ('dsa2048') con las políticas de seguridad más estrictas de Ubuntu 24.04 LTS. Esto requirió intentos de soluciones alternativas para la importación de claves y, en última instancia, sugirió la necesidad de considerar la instalación manual del software o versiones anteriores del sistema operativo.
- **Sensibilidad de la Conectividad Externa:** La imposibilidad de conectar con el servidor de repositorios de `aprsd` ('aprsd-dist.he.fi') debido a errores de "Connection refused" subrayó la criticidad de la disponibilidad de servicios externos y la necesidad de una gestión de dependencias más resiliente en entornos de despliegue.
- **Complejidad de Integración y Configuración Inter-Contenedores:** Los problemas de autenticación y visualización en el mapa, observados incluso en entornos donde el despliegue de contenedores fue exitoso, resaltan la minuciosidad requerida en la configuración de las credenciales, los puertos y el flujo de datos entre los distintos servicios (backend, frontend, `aprsd`) dentro de la arquitectura Docker Compose.
- **Importancia del Entorno de Desarrollo y Depuración:** La experiencia con Visual Studio Code, aunque no exenta de frustraciones por los fallos de despliegue, demostró su utilidad como entorno robusto para la edición de código,

la gestión de contenedores y la depuración de problemas complejos.

En síntesis, si bien la visión de un sistema de seguimiento de transporte para el TEC mediante LoRa/APRS es viable y prometedora, su materialización efectiva requiere una atención rigurosa a las dependencias de software, la compatibilidad de versiones y una estrategia de despliegue que anticipe y mitigue los problemas de seguridad y conectividad con fuentes externas. El proyecto, a pesar de no lograr una implementación completamente funcional en todos los escenarios, proporcionó una valiosa experiencia en la identificación y análisis de problemas técnicos en un sistema de comunicaciones moderno y distribuido.

XII. RECOMENDACIONES

Basado en los resultados y el análisis de la implementación del servidor APRS con TrackDirect, se proponen las siguientes recomendaciones para futuras iteraciones del proyecto o para el despliegue exitoso de sistemas similares:

A. Enfoque en la Instalación del Componente APRSD

El componente `aprsd` fue un punto crítico de falla en múltiples entornos. Para abordarlo, se recomienda:

- **Priorizar la Compilación desde Código Fuente:** Ante los persistentes problemas con el repositorio APT de `aprsd` (especialmente la incompatibilidad de la clave GPG `dsa2048` con las versiones modernas de Ubuntu y los fallos de conectividad al servidor remoto), la estrategia más robusta y fiable sería la compilación manual e instalación de `aprsd` desde su código fuente. Esto elimina la dependencia del gestor de paquetes APT y sus validaciones de claves, ofreciendo mayor control sobre el proceso.
- **Contactar a los Mantenedores del Repositorio:** Se sugiere establecer comunicación con los desarrolladores de `aprsd` para informarles sobre los problemas de compatibilidad de su clave GPG con las versiones recientes de Ubuntu. Una actualización a un algoritmo de clave más moderno (ej., RSA4096) en su repositorio APT beneficiaría a futuros usuarios.

B. Refinamiento de la Configuración y Depuración en Contenedores

La integración de `aprsd` con los demás servicios de TrackDirect dentro de Docker Compose presentó desafíos de comunicación y autenticación. Se recomienda:

- **Verificación Rigurosa de Credenciales y Parámetros de Conexión:** Realizar una auditoría exhaustiva de todos los archivos de configuración (`aprsd.conf`, variables de entorno de Docker, configuraciones de la aplicación TrackDirect) para asegurar la consistencia absoluta en puertos, nombres de host internos y credenciales de autenticación entre los servicios (`aprsd`, backend, frontend).

- **Utilización Avanzada de Logs y Monitoreo de Contenedores:** Implementar una estrategia de monitoreo detallada de los logs de Docker ('docker logs') y de los logs internos de cada aplicación para identificar de forma precisa los puntos de fallo en la comunicación inter-contenedor. Herramientas de orquestación o visualización de logs (ej., ELK Stack simplificado o Grafana Loki) podrían ser beneficiosas.

C. Estrategia de Despliegue y Pruebas

Para futuros proyectos o iteraciones, se aconseja adoptar un enfoque más incremental en el despliegue y las pruebas:

- **Despliegue por Etapas:** En lugar de intentar un `docker-compose up` completo de inmediato, desplegar y validar cada componente de forma individual (ej., primero la base de datos, luego `aprsd` de forma aislada, después el backend conectado solo a la base de datos, y finalmente el frontend). Esto permite aislar los problemas a un componente específico.
- **Pruebas de Conectividad Interna:** Una vez que los contenedores estén levantados, realizar pruebas de conectividad y autenticación entre ellos utilizando herramientas como 'curl' o 'netcat' desde dentro de los propios contenedores para diagnosticar problemas de red o configuración.

D. Consideraciones de Infraestructura y Seguridad

Los problemas de conectividad externa y la estricta política de seguridad de Ubuntu 24.04 LTS resaltan la importancia de:

- **Resiliencia ante Dependencias Externas:** Evaluar la posibilidad de mantener copias locales o cachés de repositorios críticos para minimizar la dependencia de la disponibilidad de servidores externos.
- **Compatibilidad de Versiones:** Investigar y seleccionar cuidadosamente las versiones del sistema operativo y sus librerías para asegurar la compatibilidad con todas las dependencias del proyecto, especialmente aquellas que puedan emplear tecnologías legadas (como algoritmos GPG más antiguos).

La aplicación de estas recomendaciones puede mitigar significativamente los desafíos encontrados, facilitando un despliegue exitoso y una operación estable del servidor APRS en futuras implementaciones.

REFERENCES

- [1] Ministerio de Ciencia, Innovación, Tecnología y Telecomunicaciones (MICITT). Cuadro Nacional de Atribución de Frecuencias (PNAF). Disponible en: <https://www.micitt.go.cr/pnaf>
- [2] La Gaceta N°95, Alcance N°99. Publicación oficial del 30 de mayo de 2023. Disponible en: <https://www.imprentanacional.go.cr/gaceta>
- [3] Decreto N°44010-MICITT. Reglamento de uso del espectro radioeléctrico en Costa Rica. Disponible en: <https://www.micitt.go.cr>
- [4] Unión Internacional de Telecomunicaciones (ITU). Clasificación de emisiones de radio. Disponible en: <https://www.itu.int>
- [5] Semtech Corporation. LoRaWAN Regional Parameters 2022. Disponible en: <https://lora-alliance.org>
- [6] Qvarforth, P. trackdirect. Disponible en: <https://github.com/qvarforth/trackdirect>

- [7] Políticas de OpenStreetMap. Disponible en: <https://operations.osmfoundation.org/policies/tiles/>
- [8] Bruninga, B. *APRS Protocol Specification*, 1996. Disponible en: <http://www.aprs.org/doc/APRS101.PDF>
- [9] American Radio Relay League. *AX.25 Amateur Packet-Radio Link-Layer Protocol*, 1984. Disponible en: <http://www.tapr.org/pdf/AX25.2.2.pdf>
- [10] Semtech Corporation. *LoRa Modulation Basics*, 2015. Disponible en: <https://www.semtech.com/uploads/documents/an1200.22.pdf>
- [11] Kikkert, J. *RF Electronics Design and Simulation*. James Cook University, Townsville, Australia, 2022. Disponible en: <https://resources.system-analysis.cadence.com/i/1325428-rf-electronics-design-and-simulation/2?>

XIII. ANEXOS

A. Cronograma de trabajo y horas dedicadas

Proyecto - Taller integrador 1S25

Grupo 8

Badilla Monge José David

Rojas Gonzalez Josué

Zamora Barrantes Bernal Jesús

Inicio del proyecto:

ju, 3/13/2025

Semana para mostrar:

3

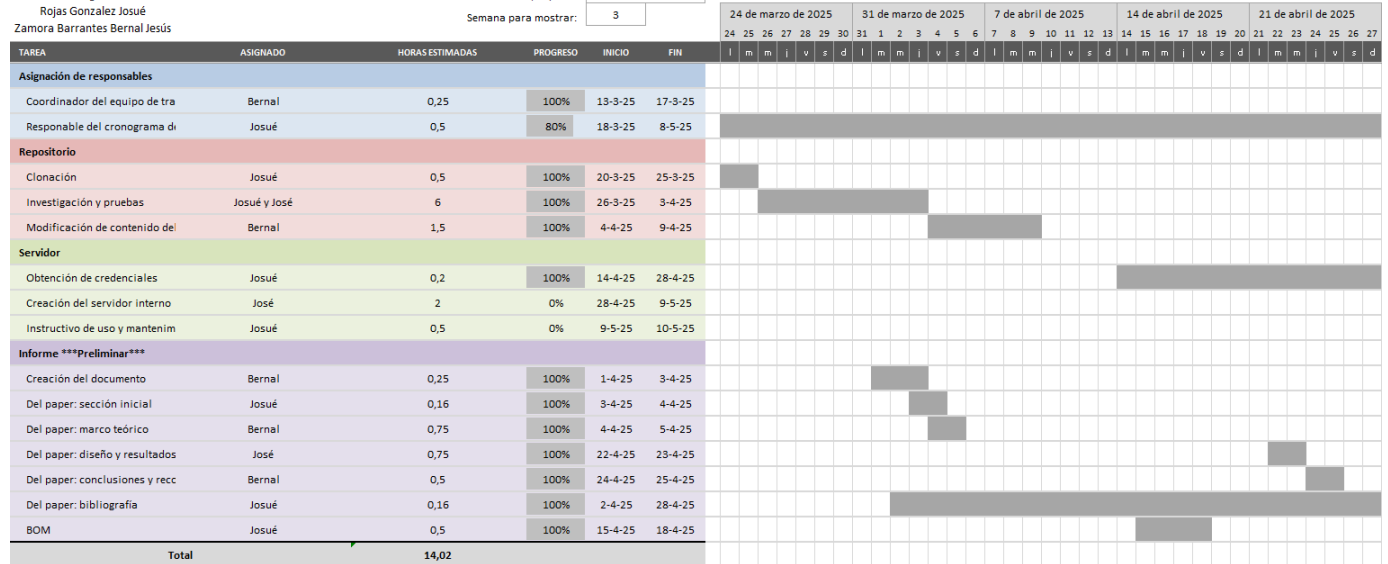


Figure 6: Cronograma

B. Build of Materials

BOM: Servidor		
Cantidad	Equipo	Costo
1	Monitor	\$ 69,99
1	Periféricos	\$ 18,99
1	Chasis de servidor 4U	\$ 89,99
1	AMD EPYC™ 7282	\$ 169,95
1	SIENAD8UD-2L2Q	\$ 624,99
2	Samsung 32GB DDR5 4800MHz	\$ 329,99
1	Fuente de alimentación Corsair HX1000i	\$ 234,99
Total		\$ 1.538,89

Figure 7: BOM

C. Repositorio

El repositorio con el que trabajamos se encuentra en: <https://github.com/jorago7/ServidorAPRS>

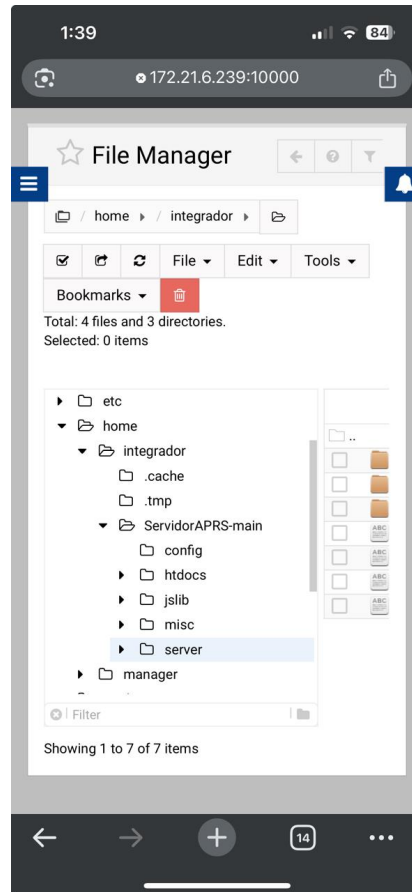
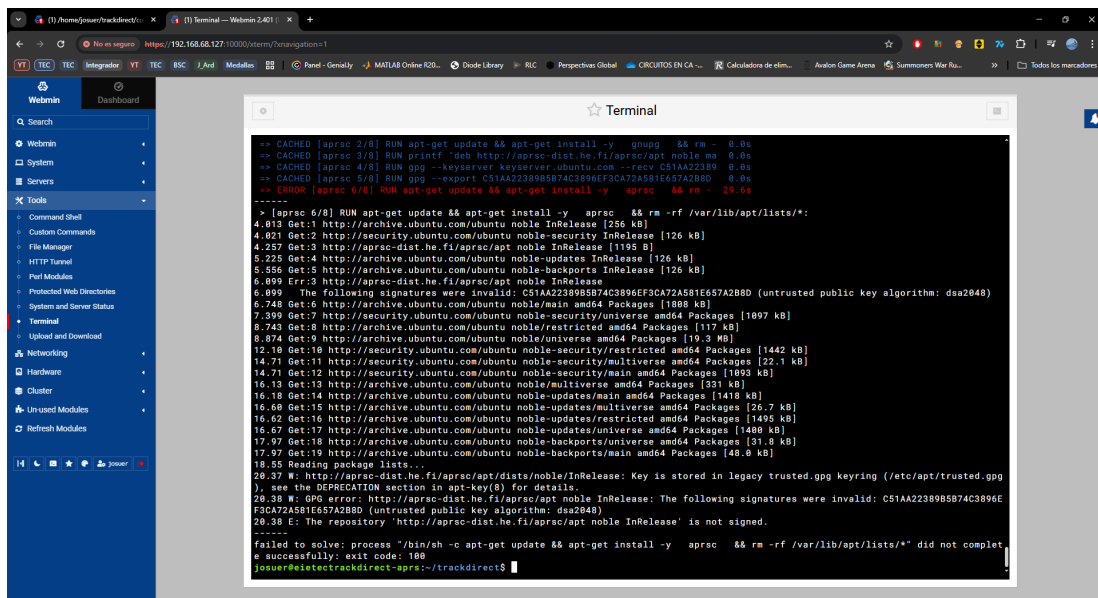


Figure 8: Webmin del servidor en el servidor de la Escuela



```
➤ [aprs 6/8] RUN apt-get update && apt-get install -y gnupg && rm -f /var/lib/apt/lists/*:  
4.013 Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]  
4.021 Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]  
4.257 Get:3 http://aprs-dist.he.fi/aprs/apt noble InRelease [1195 B]  
5.225 Get:4 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]  
5.256 Get:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]  
6.899 Err:3 http://aprs-dist.he.fi/aprs/apt noble InRelease  
6.899 The following signatures were invalid: C51AA22389B5874C3896EF3CA72A581E657A28BD (untrusted public key algorithm: dsa2048)  
6.748 Get:6 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1888 kB]  
7.399 Get:7 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1097 kB]  
8.743 Get:8 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]  
8.874 Get:9 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]  
12.18 Get:10 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [1442 kB]  
14.71 Get:11 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [22.1 kB]  
14.71 Get:12 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1893 kB]  
16.13 Get:13 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]  
16.18 Get:14 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1418 kB]  
16.68 Get:15 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [26.7 kB]  
16.62 Get:16 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [1495 kB]  
16.67 Get:17 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1400 kB]  
17.97 Get:18 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [31.8 kB]  
17.97 Get:19 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [48.0 kB]  
18.55 Reading package lists...  
20.37 W: http://aprs-dist.he.fi/aprs/apt/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.  
20.38 W: GPG error: http://aprs-dist.he.fi/aprs/apt noble InRelease: The following signatures were invalid: C51AA22389B5874C3896EF3CA72A581E657A28BD (untrusted public key algorithm: dsa2048)  
20.38 E: The repository 'http://aprs-dist.he.fi/aprs/apt noble InRelease' is not signed.  
-----  
failed to solve: process "/bin/sh -c apt-get update && apt-get install -y gnupg && rm -f /var/lib/apt/lists/*" did not complete successfully: exit code: 100  
josu@electrackdirect-aprs:~/trackdirect$
```

Figure 11: Construcción fallida en máquina virtual debido a fallo del servidor interno de APRS