

Índice general

Índice de figuras	III
Índice de tablas	IV
1. Introducción	1
1.1. ¿Qué es APRS?	1
1.2. Características Principales de APRS	1
1.3. Usos Principales de APRS	1
1.4. Historia y Desarrollo de APRS	2
2. Contexto del problema y estado del arte	4
2.1. Contexto del problema	4
2.1.1. Barreras de entrada	4
2.1.2. Hardware	4
2.1.3. OSINT y APRS	4
2.2. Estado del Arte	5
2.2.1. aprs.fi	5
2.2.2. aprs.to	5
2.2.3. Comparación	6
3. Propuesta de la Idea	7
3.1. Requisitos	7
3.2. La Idea	7
3.2.1. Enfoque	7
3.2.2. Beneficios	8
3.3. APRSINT	8
4. Arquitectura e Implementacion	9
4.1. Arquitectura	9
4.2. Tecnologías utilizadas (Marco Teórico)	9
4.2.1. Raspberry Pi	9
4.2.2. Disco duro ssd	10
4.2.3. Python	10
4.2.4. Dash	10
4.2.5. Cosmograph JS	10
4.2.6. PostgreSQL	11
4.2.7. Sqlalchemy	11
4.2.8. aprslib	12
4.2.9. AWS	12
4.2.10. Supervisord	12
4.2.11. Pandas	13
4.2.12. Apache Airflow	13

4.3.	Adquisición de datos	13
4.3.1.	RTL-SDR	13
4.3.2.	APRS-IS	14
4.3.3.	Recepción de paquetes APRS	14
4.3.4.	Procesado de paquetes APRS	15
4.3.5.	Almacenamiento de paquetes APRS en la base de datos	16
4.3.6.	Base de datos	17
4.4.	Visualización y Presentación de datos	17
4.4.1.	Dash	18
4.4.2.	Diseño	18
4.4.3.	Home - Primera pantalla	19
4.4.4.	Station - Segunda pantalla	20
4.4.5.	Graph - Tercera pantalla	21
5.	Propuesta de la Idea	24
5.1.	Implementacion y arquitectura	24
6.	Conclusiones y trabajo futuro	25

Índice de figuras

1.1.	Logo de APRS.	1
1.2.	Mapa de frecuencias APRS en el mundo.	2
1.3.	Bob Bruninga.	3
2.1.	Ejemplo de un RTL-SDR de tipo usb.	5
2.2.	Interfaz de aprs.fi	6
2.3.	Interfaz de aprs.to	6
3.1.	Estructura base de la aplicación.	8
4.1.	Infrasestructura de la red APRS - APRS-IS.	15
4.2.	Estructura de la base de datos.	17
4.3.	Primer diseño de la aplicación web en Figma.	18
4.4.	Reultados de la búsqueda de «emergency hospital».	20
4.5.	Grafo de estaciones APRS.	22

Índice de tablas

Capítulo 1

Introducción

1.1. ¿Qué es APRS?

El APRS o Sistema Automático de Reporte de Paquetes (APRS, por sus siglas en inglés: Automatic Packet Reporting System) es un sistema de comunicaciones digitales en tiempo real que permite el intercambio de información entre estaciones de radioaficionados.

De manera general, APRS se utiliza para el rastreo de vehículos, la transmisión de mensajes de texto, la diseminación de información meteorológica y la comunicación en situaciones de emergencia aunque debido a la flexibilidad del protocolo puede ser usado en cualquier otra situación.



Figura 1.1: Logo de APRS.

1.2. Características Principales de APRS

APRS posee varias características que lo hacen útil y versátil en el ámbito de la comunicación de radioaficionados, entre las cuales se encuentran las siguientes:

- **Frecuencia de Operación:** APRS opera en la banda de VHF, específicamente en la frecuencia de 144.80 MHz en Europa, aunque en otras regiones del mundo se utilizan frecuencias diferentes tal como se muestra en la figura Figura 1.2.
- **Modo de Transmisión:** APRS utiliza como modo de transmisión paquetes individuales que han de seguir un formato establecido, esto facilita enormemente la adopción e integración de esta tecnología.
- **Actualización de Datos en APRS:** En el sistema APRS, un paquete de información se transmite múltiples veces, disminuyendo gradualmente la frecuencia de envío de estas transmisiones a medida que el tiempo avanza. Este método tiene como objetivo maximizar la tasa de recepción de los paquetes.

1.3. Usos Principales de APRS

- **Reportes GPS:** APRS permite a los usuarios enviar su ubicación geográfica en forma de coordenadas, obtenida a través de sistemas GPS, lo que facilita el seguimiento de vehículos o personas en tiempo real.
- **Datos Meteorológicos:** Las estaciones meteorológicas suelen utilizar mensajes APRS para reportar diferentes datos como temperatura, humedad o presión barométrica, incluyendo información actualizada y útil para diversas aplicaciones.

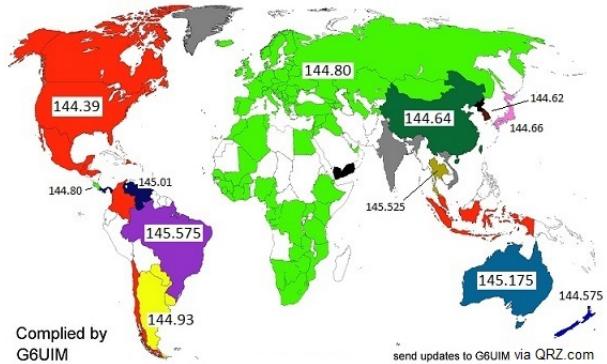


Figura 1.2: Mapa de frecuencias APRS en el mundo.

- **Integración con Internet:** Mediante el sistema APRS-IS, los mensajes APRS son accesibles por internet a través de distintos nodos, ampliando el alcance y la extensión de este sistema.
- **Uso en Emergencias:** En situaciones de emergencia, APRS es una herramienta vital para la comunicación de broadcast y el seguimiento de recursos y personal.

1.4. Historia y Desarrollo de APRS

El sistema APRS nació en la década de los 80 de la mano de Bob Bruninga, un ingeniero que trabajaba como en la Academia Naval de los Estados Unidos. Bruninga creó la primera implementación de APRS en un ordenador Apple II en 1982 con el objetivo de mapear informes de posición de la Marina en alta frecuencia.¹

El primer uso real del APRS fue en 1984, cuando Bruninga desarrolló una versión más avanzada en un VIC-20 para reportar la posición y el estado de los caballos de una carrera de resistencia.

Durante los siguientes años, Bruninga continuó perfeccionando el sistema, al que posteriormente bautizó como Sistema de Tráfico de Emergencia Sin Conexión (CETS, por sus siglas en inglés).

Tras una serie de ejercicios de la Agencia Federal de gestión de Emergencias (FEMA) usando CETS, el sistema fue trasladado al IGM pC. Durante la década de los 90, CETS (ya conocido como el Sistema de Reporte Automático de Posición) continuó evolucionando.

A medida que la tecnología GPS se volvía más ampliamente disponible, el término "Posición" fue reemplazado por "Paquete" para encapsular mejor las capacidades más genéricas del sistema y enfatizar sus usos más allá del mero reporte de posición.

¹<http://www.aprs.org/APRS-docs/ARTICLES.TXT>

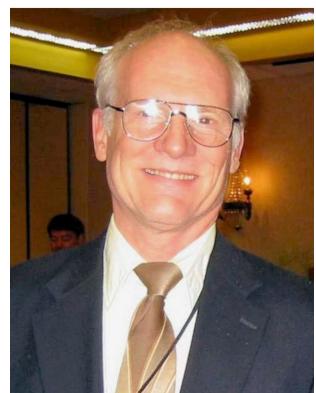


Figura 1.3: Bob Brunninga.

Capítulo 2

Contexto del problema y estado del arte

2.1. Contexto del problema

El *Automatic Packet Reporting System* (APRS) como sistema de comunicación digital se basa en la gran comunidad de radioaficionados e incluso tiene aplicaciones industriales como la transmisión de información en tiempo real de vehículos logísticos o estaciones de meteorología. A pesar de su versatilidad y flexibilidad, el APRS presenta algunas barreras de entrada para usuarios que no pertenecen a la comunidad radioaficionada o tienen sólidos conocimientos de este ámbito. En este capítulo, se analizarán estas barreras y se discutirán los requisitos necesarios.

2.1.1. Barreras de entrada

El APRS requiere conocimientos básicos sobre radiocomunicación y la obtención de una licencia de radioaficionado en caso de querer emitir. La obtención de la licencia a pesar de no implicar un desembolso económico importante, si que requiere un estudio y comprensión de los sistemas y legislación pertinente.

Adicionalmente, el APRS utiliza una estructura de mensaje específico y un conjunto de protocolos que pueden ser difíciles de entender para usuarios no familiarizados con este tipo de tecnología. La falta de documentación y antigüedad de esta hacen que desarrollar soluciones para este sistema sea complicado.

2.1.2. Hardware

Para utilizar APRS, se necesita hardware especializado que incluya un transceptor de radio, un *Terminal Node Controller* (TNC) y un dispositivo GPS (opcionalmente), existen otras opciones más baratas como los dispositivos *software defined radio* o RTL-SDR Figura 2.1 que se conectan directamente a un ordenador. Los TNC son dispositivos que convierten las señales digitales emitidas por un ordenador en señales de radio y viceversa. El dispositivo GPS proporciona información de ubicación que se puede transmitir junto con otros datos en el cuerpo del mensaje.

El precio del hardware APRS puede variar dependiendo de la calidad y las características del equipo. Sin embargo, a pesar de no contar con precios prohibitivamente altos, la inversión que supone conlleva el hecho de que solamente los usuarios con un interés previo en la radioafición o la tecnología en general estén dispuestos a adquirirlo.

2.1.3. OSINT y APRS

El APRS sirve como una herramienta de comunicación para los radioaficionados y de reporte de telemetría para las industrias. Pero se puede convertir en una valiosa fuente de información para la obtención de inteligencia (OSINT). Esto se debe a su capacidad para proporcionar una gran cantidad de datos en tiempo real sobre ubicación y estado de los activos así como una amplia gama de información adicional.

En el ámbito del OSINT, el APRS ofrece una serie de aplicaciones prácticas. Por ejemplo, los datos APRS son actualmente utilizados para el rastreo de vehículos en tiempo real, lo que resulta



Figura 2.1: Ejemplo de un RTL-SDR de tipo usb.

muy útil para empresas de logística y transporte. Además, la red APRS se utiliza para monitorear la actividad de estaciones meteorológicas, proporcionando datos en tiempo real sobre condiciones climáticas locales. También se utiliza en situaciones de emergencia, como pueden ser desastres naturales o incidentes críticos, el APRS desempeña un papel crucial al permitir la transmisión rápida de información sobre ubicaciones de refugios, recursos disponibles y necesidades de ayuda ya que no depende necesariamente de infraestructura de una entidad como si lo hace la red telefónica.

Aunque el APRS tiene un gran potencial para aplicaciones en el ámbito del OSINT, su uso en esta área es prácticamente inexistente. Este hecho puede explicarse en parte por las barreras de entrada mencionadas anteriormente, que incluyen la necesidad de poseer conocimientos especializados en radiocomunicación, obtener licencias de radioaficionado y adquirir hardware especializado.

2.2. Estado del Arte

Existen varios sitios web que ofrecen servicios relacionados con APRS, entre los cuales destacan aprs.fi y aprs.to probablemente las dos webs más grandes.

2.2.1. aprs.fi

aprs.fi es una de las webs más utilizadas para visualizar datos APRS en tiempo real y acceder a un histórico extenso de información. Sin embargo, su interfaz puede considerarse un tanto desactualizada (ver Figura 2.2), lo que puede dificultar la navegación y la búsqueda de información específica. Aunque ofrece una gran cantidad de datos y un histórico significativo, la plataforma carece de capacidades avanzadas de filtrado de estaciones y mensajes. Además, para acceder a funciones más complejas y realizar extracciones de datos, es necesario crear una cuenta.

2.2.2. aprs.to

Por otro lado, aprs.to ofrece una interfaz más moderna y cuidada (ver Figura 2.3), lo que facilita la navegación y la búsqueda de información. Además, permite realizar búsquedas básicas y aplicar filtros para refinar los resultados. Sin embargo, al igual que aprs.fi, también requiere que los usuarios se creen una cuenta para acceder a ciertas funcionalidades avanzadas.

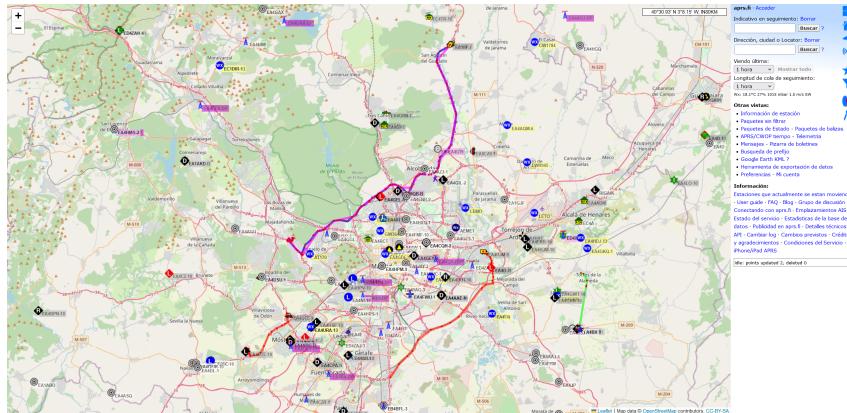


Figura 2.2: Interfaz de aprs.fi

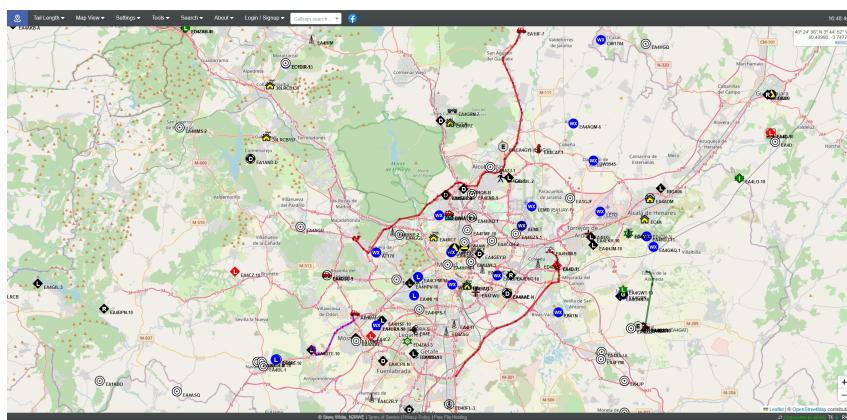


Figura 2.3: Interfaz de aprs.to

2.2.3. Comparación

En resumen, aprs.fi, la web de visualización de APRS por excelencia, ofrece una gran base de datos y una gran cantidad de información histórica de datos APRS, pero su interfaz puede resultar algo desactualizada y carece de capacidades avanzadas de filtrado. Por otro lado, aprs.to presenta una interfaz más moderna y permite realizar búsquedas básicas y aplicar filtros, aunque también requiere crear una cuenta para acceder a ciertas funciones. Ambas plataformas tienen puntos fuertes y débiles y por esa razón lo mejor es usarlas en conjunto.

Capítulo 3

Propuesta de la Idea

3.1. Requisitos

La solución propuesta debe cumplir con una serie de requisitos imprescindibles para asegurar su utilidad y accesibilidad:

- **Bajo Costo:** La solución debe ser económica para garantizar su accesibilidad a una amplia gama de usuarios, incluidos aquellos con presupuestos limitados.
- **Flexibilidad de Alojamiento:** La solución debe ofrecer la capacidad de autoalojamiento, los usuarios deben poder optar por alojarla en sus propios servidores o usar la solución en la nube según sus preferencias y requisitos específicos.
- **Extensible:** La solución debe ser extensible, lo que significa que debe permitir la integración de nuevas funcionalidades y la expansión de su capacidad según las necesidades cambiantes de los usuarios.
- **Mejor Filtrado:** Se debe ofrecer un filtrado más preciso y flexible en comparación con las plataformas existentes, lo que permitirá a los usuarios obtener información relevante y útil de los datos APRS para complementar la ya provista por las alternativas.
- **Análisis de Datos Integrado:** La solución debe incluir un análisis de datos integrado para ayudar a los usuarios a comprender mejor la información recibida a través del APRS y obtener perspectivas y conclusiones valiosas a partir de los mensajes transmitidos.

3.2. La Idea

Se propone una solución completa que incluya la adquisición, procesamiento, visualización y análisis de datos APRS. La solución se centrará en mejorar la experiencia del usuario y enriquecer la información disponible a través de la integración de datos de diversas fuentes abiertas y disponibles.

3.2.1. Enfoque

- **Experiencia de Usuario Mejorada:** La solución tendrá una interfaz intuitiva, rápida y útil que facilite la navegación y la interacción con los datos.
- **Análisis avanzado:** La solución contará con herramientas avanzadas para la visualización detallada del tráfico APRS, filtrado preciso y análisis de datos, con el objetivo de extraer inteligencia de los mensajes recibidos.
- **Complementación:** La solución no tiene como objetivo sustituir a aprs.fi, aprs.to u otras plataformas similares, sino ofrecer una alternativa con funcionalidades complementarias que enriquezcan la experiencia del usuario.
- **Enriquecimiento de Información:** Se recopilarán datos provenientes de diversas fuentes abiertas y disponibles para ofrecer una visión más completa del tráfico APRS y sus usuarios, mejorando así la calidad de la información disponible para los usuarios.

3.2.2. Beneficios

- **Mayor Comprensión del Tráfico APRS:** Los usuarios podrán obtener información más detallada y procesable a partir de los mensajes transmitidos, lo que les permitirá una mayor comprensión del tráfico APRS.
- **Toma de Decisiones más Efectiva:** El análisis de datos integrado ayudará a los usuarios a tomar decisiones más informadas en base a la información recibida, mejorando así la eficacia de sus acciones.
- **Flexibilidad:** La opción de autoalojamiento permitirá a los usuarios tener un mayor control sobre sus datos y privacidad, proporcionando así una mayor flexibilidad en cuanto a su gestión.

3.3. APRSINT

La solución propuesta se ha llamado **APRSINT**, que es una combinación de APRS y OSINT. APRSINT esta dividida en tres módulos principales:

- **Adquisición de Datos:** Este módulo se encargará de recopilar y almacenar los mensajes APRS de diversas fuentes, así como de integrar datos de otras fuentes abiertas y disponibles.
- **Procesamiento y Análisis:** Este módulo se encargará de procesar y analizar los datos recopilados, ofreciendo herramientas avanzadas de visualización, filtrado y análisis de datos.
- **Visualización y Presentación:** Este módulo se encargará de presentar los datos procesados y analizados de forma clara y comprensible, facilitando la interpretación y la toma de decisiones.

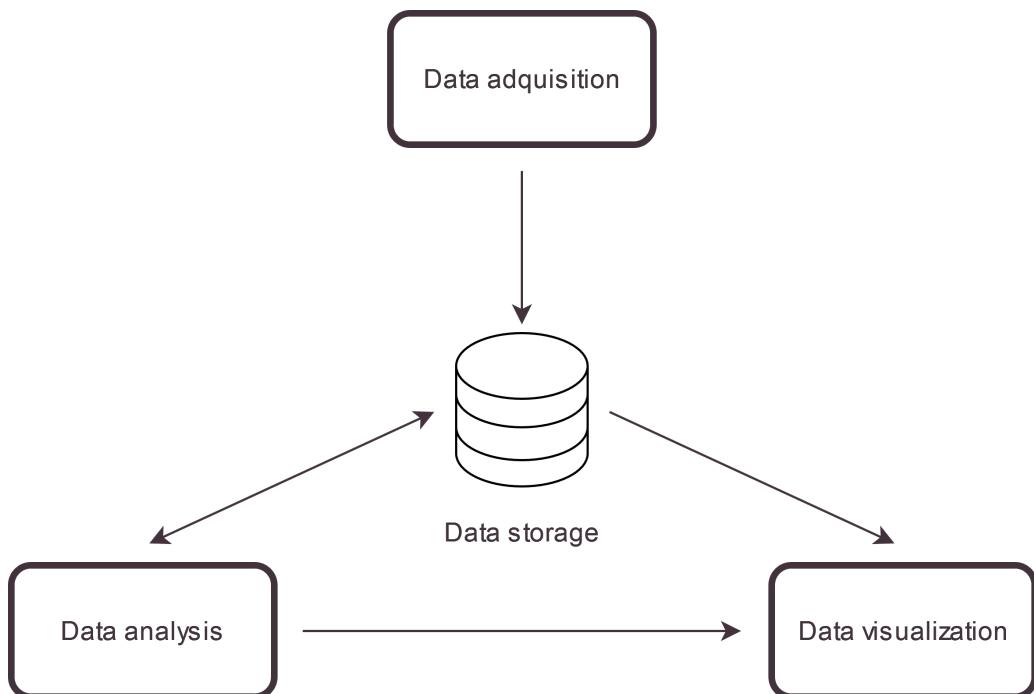


Figura 3.1: Estructura base de la aplicación.

Cada uno de estos módulos será explicado en detalle en el capítulo siguiente.

Capítulo 4

Arquitectura e Implementación

En este capítulo se describirá la arquitectura de la aplicación, así como las tecnologías utilizadas para su implementación.

4.1. Arquitectura

Se pueden categorizar los tres módulos (**Adquisición de datos, Procesamiento y análisis, Visualización y Presentación**) de la aplicación mencionados en el capítulo anterior en dos categorías principales:

- **Adquisición de paquetes APRS:** En este módulo se encuentran los componentes encargados de la adquisición de paquetes APRS. Estos componentes se encargan de recibir los paquetes APRS, procesarlos y almacenarlos en la base de datos de la aplicación.
- **Aplicación web APRSINT:** En este módulo se encuentran los componentes encargados del análisis, la interpretación y la visualización, de los datos almacenados en la base de datos de la aplicación. Estos componentes se encargan de presentar la información de manera rápida y cómoda para el usuario.

4.2. Tecnologías utilizadas (Marco Teórico)

Como se ha mencionado anteriormente, uno de los objetivos de APRSINT es el de ser una solución accesible y fácil de usar. Para lograr este objetivo, se han seleccionado tecnologías ampliamente utilizadas y bien documentadas. A continuación se describen las tecnologías utilizadas en cada uno de los módulos de la aplicación.

4.2.1. Raspberry Pi

La Raspberry Pi es un *Single board computer (SBC)* u ordenador de placa única desarrollada por la Fundación Raspberry Pi. Las raspberry pi son muy populares en el mundo de la informática y la electrónica por su bajo precio, reducido tamaño y su versatilidad.

- **Bajo precio:** La Raspberry es una opción económica para implementar desde prototipos hasta aplicaciones simples, lo que la hace muy accesible para una amplia gama de usuarios.
- **Bajo Consumo Energético:** Su diseño de bajo consumo energético la hace ideal para aplicaciones que requieren funcionamiento continuo.
- **Versatilidad:** La Raspberry Pi 4 es altamente versátil y puede adaptarse a una variedad de casos de uso, desde servidores ligeros hasta placas de desarrollo para robótica y automatización.

Se ha seleccionado la Raspberry Pi 4b¹ de 8GB de memoria ram como plataforma de hardware para la implementación de la solución, debido a sus características y capacidades.

¹Cuando se comenzó el proyecto todavía no se había lanzado la versión 5.

4.2.2. Disco duro ssd

Los discos duros de estado sólido (SSD) son una alternativa a los discos duros tradicionales (HDD) que ofrecen una mayor velocidad de lectura y escritura, menor consumo energético y mayor durabilidad. Se ha seleccionado un disco duro SSD de 250GB que se haya conectado a la raspberry-pi para almacenar el gran volumen de datos que consume y genera la aplicación a su velocidad y fiabilidad.

4.2.3. Python

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general. Es ampliamente utilizado en el desarrollo de aplicaciones web, científicas y de análisis de datos debido a su simplicidad, flexibilidad y facilidad de uso. Se ha seleccionado Python como lenguaje de programación principal para la implementación de la solución debido a la gran versatilidad que ofrece y sobretodo por la existencia de extensas bibliotecas de visualización y manejo de grandes volúmenes de datos.

4.2.4. Dash

Dash es un framework de Python creado por Plotly para la creación de aplicaciones web interactivas y visualizaciones de datos. Dash permite crear aplicaciones web interactivas y visualizaciones de datos atractivas utilizando Python como lenguaje de programación. Dash está construido encima de React. Se ha seleccionado Dash como framework para la implementación de la interfaz web de la aplicación debido a su facilidad de uso y a la gran cantidad de funcionalidades que ofrece. Dash está escrito encima de Plotly.js, React y Flask lo que permite una gran capacidad de personalización.

- **Interactividad:** Dash permite crear aplicaciones web altamente interactivas, lo que facilita la exploración de datos y la toma de decisiones.
- **Flexibilidad:** Su arquitectura modular y su amplia gama de componentes permiten la creación de aplicaciones web personalizadas y adaptadas a las necesidades específicas del usuario.
- **Integración con Plotly:** Al estar desarrollado por Plotly, Dash ofrece una integración perfecta con las capacidades de visualización de datos de Plotly, lo que permite crear gráficos y visualizaciones muy atractivas.

Para crear la aplicación web se consideraron algunas alternativas como Django, Streamlit y PowerBI, sin embargo, Dash fue la opción escogida debido a su extensa capacidad de personalización y personalización de la que carecían las demás opciones.

4.2.5. Cosmograph JS

Cosmograph es una biblioteca de JavaScript enfocada en la visualización de grandes grafos y redes complejas en aplicaciones web. Permite representar de manera interactiva relaciones entre entidades, facilitando la comprensión y el análisis de datos estructurados.

- **Visualización de Grafos:** Cosmograph ofrece herramientas avanzadas para la representación visual de grafos como líneas temporales, histogramas y búsquedas de nodos, una mayor comprensión y capacidad de análisis de la información.

- **Rendimiento:** Cosmograph a diferencia de otras librerías más populares como Sigma JS transfiere todos los cálculos de posiciones de los nodos y aristas así como la representación gráfica de este a la GPU. Esto permite la visualización de grafos con miles de nodos y aristas sin afectar el rendimiento de la aplicación.
- **Personalización:** Ofrece opciones de personalización para adaptar la apariencia y el comportamiento de los nodos y aristas según las necesidades específicas del usuario.

Después de una gran cantidad de pruebas y tras considerar muchas alternativas como Sigma Js, CytoScape y networkx, se acabó eligiendo Cosmograph sobre todo por su rendimiento.

4.2.6. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto y potente, conocido por su fiabilidad, robustez y capacidad para manejar grandes volúmenes de datos. Ofrece una amplia gama de características avanzadas que lo hacen adecuado para aplicaciones web y empresariales exigentes.

- **Fiabilidad y Robustez:** PostgreSQL es conocido por su alta fiabilidad y capacidad para manejar grandes cargas de trabajo sin sacrificar el rendimiento.
- **Escalabilidad:** Es altamente escalable y puede manejar grandes volúmenes de datos y transacciones concurrentes sin problemas.
- **Funcionalidades Avanzadas:** Ofrece una amplia gama de funcionalidades avanzadas, como soporte para transacciones ACID, vistas materializadas, procedimientos almacenados y Full Text Search (búsqueda de indizada).
- **Almacenamiento de datos semiestructurados:** PostgreSQL es capaz de almacenar y manipular datos no estructurados como JSON de manera eficiente, lo que lo hace adecuado para aplicaciones que requieren almacenamiento de datos semiestructurados.
- **Rendimiento:** PostgreSQL ofrece un rendimiento sólido, especialmente en entornos de alta concurrencia y cargas de trabajo intensivas.

La elección de PostgreSQL como sistema de gestión de bases de datos se debió a su robustez, facilidad de uso y a la gran cantidad de funcionalidades que ofrece.

4.2.7. SQLAlchemy

SQLAlchemy es una biblioteca de Python que facilita la interacción con bases de datos relacionales utilizando un enfoque orientado a objetos. Permite trabajar con diferentes motores de bases de datos, como PostgreSQL, MySQL, SQLite, entre otros, de una manera consistente y eficiente.

- **Abstracción de la Base de Datos:** SQLAlchemy proporciona una capa de abstracción sobre la base de datos, lo que permite a los desarrolladores interactuar con la base de datos utilizando objetos de python en lugar de consultas SQL directas.
- **Compatibilidad con Múltiples Motores:** Es compatible con una variedad de motores de bases de datos, lo que brinda flexibilidad para trabajar con diferentes sistemas de gestión de bases de datos según las necesidades del proyecto. En este caso se ha utilizado el motor psycopg2 para la conexión con PostgreSQL.
- **ORM (Mapeo Objeto-Relacional):** Ofrece un ORM potente y flexible que mapea objetos Python a tablas de bases de datos, facilitando el manejo de relaciones entre objetos y la persistencia de datos.

- **Seguridad:** SQLAlchemy proporciona herramientas para prevenir ataques de inyección SQL y otros problemas de seguridad comunes en el manejo de bases de datos.

La elección de SQLAlchemy se ha basado en su capacidad para mejorar el proceso de interacción con la base de datos. Permitiendo una estrecha integración entre la base de datos y la aplicación, permitiendo una flexibilidad muy grande a la hora de hacer consultas o inserciones y sobre todo creando una capa de seguridad para evitar ataques.

4.2.8. aprslib

aprslib es una biblioteca de Python que facilita la interacción con el sistema APRS. Permite recibir y decodificar paquetes APRS, así como enviar paquetes APRS a través de la red APRS-IS.

- **Recepción de Paquetes:** aprslib permite recibir paquetes APRS de la red APRS-IS de manera sencilla.
- **Decodificación de Paquetes:** Facilita la decodificación de paquetes APRS, permitiendo extraer información útil como la posición, velocidad y rumbo de los objetos rastreados.
- **Envío de Paquetes:** aprslib permite enviar paquetes APRS a través de la red APRS-IS, lo que facilita la integración con el sistema APRS.

Se ha elegido la librería aprs por su facilidad de uso, su documentación y su capacidad para decodificar los paquetes APRS.

4.2.9. AWS

TODO Amazon Web Services (AWS) es una plataforma de servicios en la nube ofrecida por Amazon. Proporciona servicios de infraestructura informática, almacenamiento, bases de datos, análisis e inteligencia artificial, entre otros.

- **Fiabilidad:** La infraestructura global de AWS está diseñada para ser altamente disponible y resistente a fallos, lo que garantiza la continuidad del servicio y la seguridad de los datos.
- **Variedad de Servicios:** AWS ofrece una amplia gama de servicios, desde almacenamiento y bases de datos hasta aprendizaje automático y análisis de datos, lo que permite a las empresas construir y desplegar una amplia variedad de aplicaciones y soluciones.
- **Flexibilidad:** AWS proporciona opciones flexibles de implementación, incluyendo la capacidad de utilizar infraestructura física, virtual o basada en contenedores, según las necesidades del proyecto.
- **Seguridad:** AWS cuenta con robustas medidas de seguridad para proteger los datos y las aplicaciones, incluyendo controles de acceso, cifrado de datos y protección contra amenazas.

La elección de AWS así como los detalles de la implementación en la nube se describirán en la siguiente sección.

4.2.10. Supervisord

Supervisor es un sistema de control de procesos para sistemas operativos tipo Unix, diseñado para iniciar, detener y gestionar procesos de manera sencilla y robusta. Permite supervisar y mantener en funcionamiento aplicaciones y servicios, reiniciándolos automáticamente en caso de fallos o reinicios del sistema.

- **Gestión de Procesos:** Supervisor facilita la gestión de procesos al permitir iniciar, detener, reiniciar y supervisar procesos de manera centralizada.
- **Monitorización:** Supervisor proporciona información detallada sobre el estado de los procesos, incluyendo registros de eventos y estadísticas de rendimiento.
- **Reinicio Automático:** En caso de fallos, Supervisor puede reiniciar automáticamente los procesos afectados, minimizando el tiempo de inactividad y manteniendo la disponibilidad del servicio.

Se ha seleccionado Supervidord como gestor de procesos gracias a su facilidad, flexibilidad (permitiendo ejecutar scripts de python directamente) y robustez.

4.2.11. Pandas

Pandas es la librería de facto de python para el manejo y análisis de datos estructurados. Está escrita sobre numpy (escrito en c) por lo que cuenta con muy buen rendimiento. Esta librería permite la lectura y escritura de datos en diferentes formatos, la manipulación de datos, la limpieza de datos y la creación de gráficos.

4.2.12. Apache Airflow

Apache Airflow es una plataforma de orquestación de tareas y flujos de trabajo. Es similar a Cron de Unix pero permite una mayor personalización y control de las tareas que ejecuta. Las tareas se definen en archivos separados facilitando la compartimentalización y ofreciendo una mayor robustez y escalabilidad.

- **Orquestación de Flujos de Trabajo:** Airflow facilita la orquestación de flujos de trabajo complejos al permitir definir tareas y sus dependencias como DAGs, lo que proporciona una visión clara de la lógica de ejecución.
- **Escalabilidad:** Airflow es altamente escalable y puede manejar flujos de trabajo de cualquier tamaño, desde tareas simples hasta flujos de trabajo altamente complejos.
- **Monitoreo y Alertas:** Proporciona una interfaz de usuario web para monitorear el estado de los flujos de trabajo, así como capacidades de alerta para detectar y responder a fallos o retrasos en la ejecución de tareas.
- **Extensibilidad:** Airflow es altamente extensible y permite integrar fácilmente con otros sistemas y herramientas, lo que permite construir flujos de trabajo personalizados que se adapten a las necesidades específicas del proyecto.

4.3. Adquisición de datos

En esta sección se describirá con detalle el módulo de adquisición de datos de la aplicación, incluyendo la arquitectura, los componentes y las tecnologías utilizadas.

4.3.1. RTL-SDR

En la primera fase del proyecto se compró un dongle RTL-SDR con el fin de recibir los paquetes APRS y procesarlos a partir de ello. Estos dongles son relativamente baratos oscilando entre los 30 y 100 euros, se conectan mediante usb al ordenador y tienen en la parte superior un conector SMA para conectar una antena. La peculiar de estos dispositivos reside en que son controlables

mediante software y se pueden sintonizar en un rango de frecuencias muy amplio (24MHz a 1.7GHz).

Para hacer uso de estos dispositivos se ha de usar software como GQRX, CubicSDR o SDRSharp. Es posible también usar rtl-sdr en la terminal creando posteriormente un microfono virtual para que el software de análisis pueda decodificar los paquetes.

Tras multitud de pruebas fallidas con distintos software de visualización y análisis y software de audio como direwolf, se decidió no continuar por ese camino y probar otras alternativas como la que finalmente se ha implementado el **APRS-IS**

4.3.2. APRS-IS

APRS-IS es un sistema que permite la transmisión de mensajes APRS por la red de Internet. La ventaja que ofrece este sistema frente al de recopilar información mediante una antena es que APRS-IS, permite obtener un flujo de datos mucho mayor ya que hay una gran cantidad de antenas y otra ventaja con la que cuenta es que no esta restringido al área que puede recibir la pequeña antena rtl-sdr sino a todo el mundo.

Infraestructura de la red APRS-IS

Componentes principales:

- **Trackers o TNC:** Son los emisores de los paquetes APRS, suelen corresponder a estaciones fijas o móviles que envían por radio los paquetes con información de posición, velocidad, rumbo, etc.
- **Digipeaters:** Son estaciones que reciben los paquetes APRS y los retransmiten, permitiendo que los paquetes lleguen a una mayor distancia. Son análogos a los repetidores de internet.
- **I-Gates:** Son estaciones que reciben los paquetes APRS por radio y los envían a la red APRS-IS, permitiendo que los paquetes sean accesibles a través de internet.
- **Servidores APRS-IS:** Son servidores que reciben los paquetes APRS de los I-Gates y los almacenan en una base de datos, permitiendo que los clientes accedan a los paquetes a través de internet. Los servidores APRS-IS suelen estar distribuidos geográficamente para mejorar la disponibilidad y la redundancia.
- **Clientes APRS-IS:** Pueden ser otros servidores, aplicaciones web o aplicaciones móviles que acceden a los servidores APRS-IS para obtener los paquetes APRS y mostrarlos a los usuarios.

Se muestra el flujo de datos de la red APRS-IS en la Figura 4.1.

Se ha seleccionado APRS-IS como fuente de datos para la aplicación debido a su facilidad de uso, su disponibilidad y la gran cantidad de información que ofrece.

4.3.3. Recepción de paquetes APRS

Para la recepción de paquetes APRS se ha utilizado la librería aprslib de Python. Esta librería permite conectarse a un servidor APRS-IS y recibir los paquetes APRS en tiempo real. Se ha creado un sistema que utilizando esta librería se conecta a un servidor APRS-IS, recibe los paquetes APRS y los guarda en un buffer en memoria. Cuando el buffer se llena con aproximadamente 10.000 mensajes en aproximadamente 15 segundos, se procede a escribir el buffer

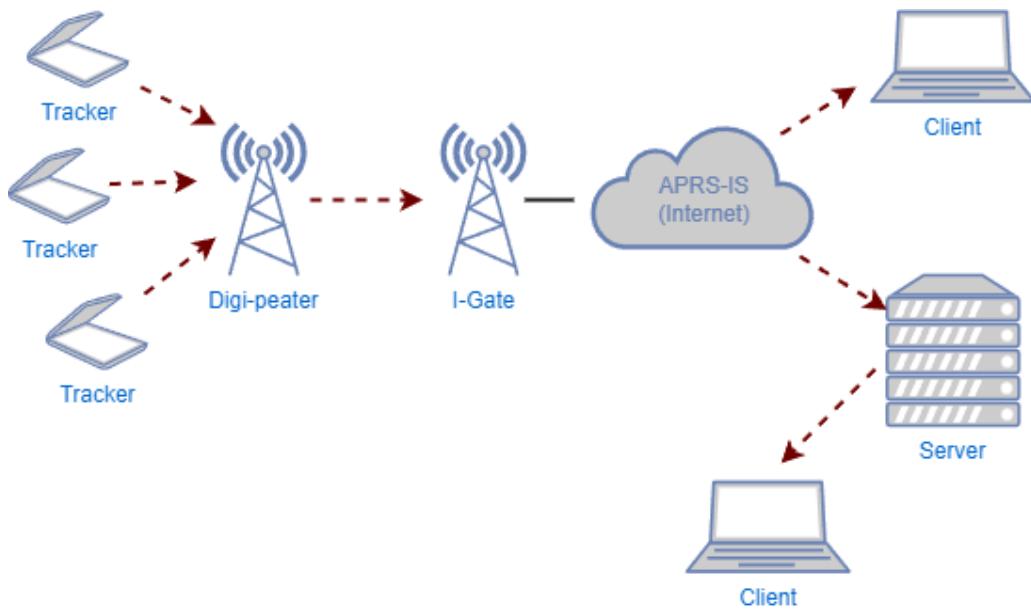


Figura 4.1: Infraestructura de la red APRS - APRS-IS.

en un fichero comprimido en el disco duro SSD. Una cosa curiosa es que los mensajes APRS no se decodifican en este punto, de hecho se guardan sin ningún tipo de procesamiento en ficheros binarios comprimidos en formato gzip para no ocupar mucho espacio en disco. Esto se hace así para evitar la sobrecarga de procesamiento en la raspberry pi y para poder procesar los mensajes en un servidor más potente en la nube. Es importante mencionar que el sistema de recepción de paquetes APRS se ejecuta en segundo plano en modo demonio utilizando el gestor de procesos Supervisord, lo que garantiza que el sistema se mantenga en funcionamiento incluso en caso de fallos o reinicios del sistema.

4.3.4. Procesado de paquetes APRS

Subida de ficheros a AWS S3

Una vez los ficheros se han guardado en el disco duro SSD, se procede a subirlos a un bucket S3 en AWS. Para ello se ha utilizado la librería boto3 de Python, que permite interactuar con los servicios de AWS desde Python. Se ha creado un script que se ejecuta una vez al día mediante la orquestación de Apache Airflow y que sube los ficheros comprimidos al bucket S3. Una vez subidos los ficheros, se eliminan del disco duro SSD para liberar espacio. Cuando un fichero se sube a S3 se desencadena un evento

Procesado de paquetes APRS en AWS lambda

Una vez los ficheros se han subido a S3, se desencadena un evento que ejecuta una función lambda en AWS. Esta función lambda se encarga de descomprimir el fichero, procesar los mensajes APRS y convertirlos en formato Json.

Por defecto lambda no puede hacer uso de librerías de terceros por lo que se ha tenido que encapsular la librería aprslib en un fichero zip, subirlo a lambda y establecerlo como una capa de lambda para poder hacer uso de ella. Para la descompresión de los ficheros se ha utilizado la librería gzip de Python, que permite leer y escribir ficheros comprimidos en formato gzip. Para el procesado de los mensajes APRS se ha utilizado la librería aprslib de Python como ya hemos mencionado, que permite decodificar los mensajes APRS y extraer información útil

como la posición, velocidad y rumbo de los objetos rastreados en formato clave-valor. Esta función lambda se ejecuta de manera asíncrona y paralela, lo que permite procesar grandes volúmenes de mensajes APRS de manera eficiente y escalable. Una vez procesados los mensajes APRS, se eliminan los ficheros comprimidos del S3 de entrada (aprsinput) para liberar espacio y evitar duplicados. Finalmente los mensajes procesados se guardan en un bucket S3 de salida (aprsoutput) en formato Json para su posterior descarga. La descarga de los ficheros Json ocurre una vez al día mediante la orquestación de Apache Airflow y la librería boto3. Estos ficheros (uno por fichero de entrada) se descargan a otra carpeta temporal en la raspberry pi.

4.3.5. Almacenamiento de paquetes APRS en la base de datos

Una vez los ficheros Json ya procesados se han descargado al directorio temporal en la raspberry pi, se procede a almacenarlos en la base de datos PostgreSQL. Para ello se ha utilizado la librería SQLAlchemy de Python, que permite interactuar con bases de datos relacionales de manera sencilla y eficiente. Se ha creado un script que se ejecuta una vez al día mediante la orquestación de Apache Airflow y que lee los ficheros Json, procesa los mensajes APRS y los almacena en la base de datos. Este paso esconde un gran problema que hubo de ser solucionado. El script en un principio seguía los siguientes pasos.

1. **Lectura de ficheros Json:** El script lee los ficheros Json de la carpeta local /tmp descargados del bucket S3.
2. **Extracción de la estacion:** El script extrae la estación origen y destino de cada mensaje y en caso de no existir en la BD las crea.
3. **Extracción de las posiciones:** Se extraen las posiciones (si las hay) de cada mensaje y el timestamp si lo hay y se almacenan en la BD.
4. **Identificación del país:** Mediante la librería geopy y ficheros de formas de países se identifica el país de la estación.
5. **Extracción de los mensajes:** Se extraen los mensajes de cada linea del JSON y se almacenan en la BD.

Este enfoque resultó tener varios problemas. El primero y más importante era el de rendimiento ya que cada archivo JSON contenía alrededor de 10000 mensajes y en la creación de los objetos, inserciones en la BD y detección del país se tardaba alrededor de 40 segundos por fichero JSON. Esto aunque no parezca demasiado tiempo, si se tiene en cuenta que la cantidad de mensajes APRS que se reciben en 40 segundos es alrededor de 45000 mensajes, se puede ver que el sistema no era escalable.

Optimizaciones

Para solucionar este problema se realizaron las siguientes optimizaciones:

- **Uso de sets:** Se ha cambiado el uso de consultas a la bd por sets en la creación de las estaciones para acelerar la comprobación de si una estación ya existe en la base de datos.
- **Batch Insert:** Se ha modificado la interfaz con Sqlalchemy para permitir inserciones en lotes de 250 mensajes.
- **Detección de países con geopandas:** Se ha cambiado la librería geopy por geopandas que permite la detección de países de manera mucho más rápida utilizando paralelización.

Estas optimizaciones han permitido que el tiempo de procesado de un fichero JSON haya pasado de 40 segundos a unos 2 segundos, lo que ha permitido que el sistema sea escalable y pueda procesar grandes volúmenes de mensajes APRS de manera eficiente.

4.3.6. Base de datos

Como se ha mencionado previamente, se ha elegido PostgreSQL como sistema de gestión de bases de datos para la aplicación. Esta elección se ha basado en la robustez, la fiabilidad y la capacidad de manejar grandes volúmenes de datos que ofrece PostgreSQL incluso en sistemas con recursos reducidos. La base de datos se ha diseñado siguiendo un modelo relacional Figura 4.2 que permite almacenar y relacionar la información de los mensajes APRS de manera eficiente, el modelo cuenta con las siguientes tablas.

- **stations:** Almacena la información de las estaciones APRS, incluyendo el identificador (CALLSIGN), el ssid y su símbolo.
- **station_locations:** Almacena la información de las posiciones ¹ de las estaciones APRS, incluyendo el país, la latitud, la longitud y la fecha y hora en la que se han recibido.
- **messages:** Almacena la información de los mensajes APRS, incluyendo el contenido del mensaje, el tipo de mensaje y el timestamp.
- **qrz_profiles:** Sirve como una cache que almacena la información de los perfiles de los usuarios de QRZ, incluyendo el identificador (CALLSIGN), el nombre, la dirección, la ciudad, el estado, el código postal, el país, la latitud, la longitud y la fecha de nacimiento entre muchos otros.

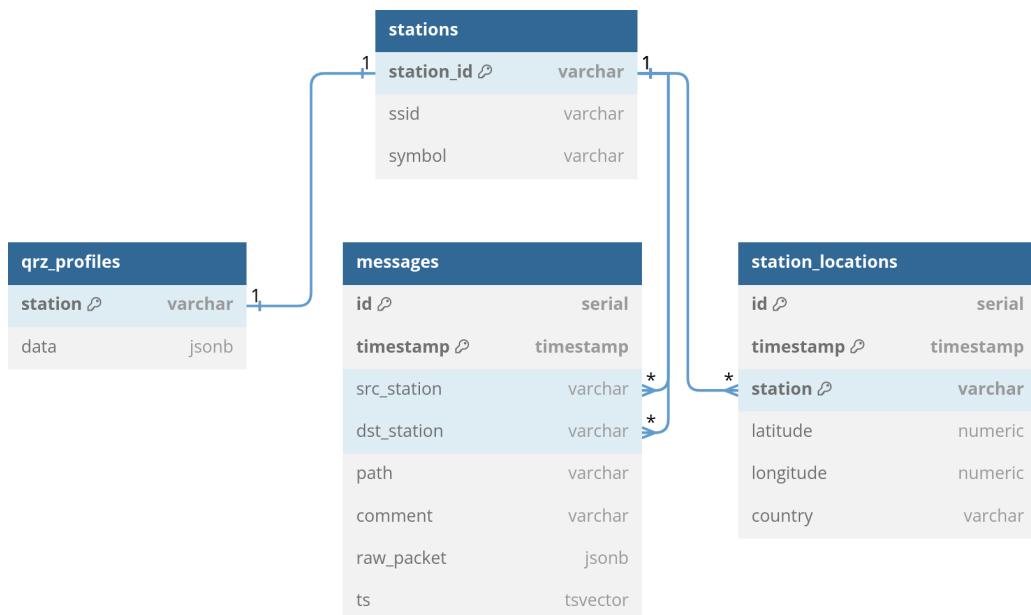


Figura 4.2: Estructura de la base de datos.

4.4. Visualización y Presentación de datos

En esta sección se describirá con detalle el módulo de visualización y presentación de datos de la aplicación, incluyendo la arquitectura, los componentes y las tecnologías utilizadas.

¹Esta tabla tiene un campo id en la clave primaria por si una estación ha emitido un mensaje sin timestamp

4.4.1. Dash

Dash es un framework de Python creado por Plotly para la creación de aplicaciones web interactivas y visualizaciones de datos. Dash permite crear aplicaciones web interactivas y visualizaciones de datos atractivas utilizando Python como lenguaje de programación. Dash está escrito encima de Plotly.js, React y Flask lo que permite una gran capacidad de customización. Dash ofrece una versión de pago llamada Dash Enterprise que ofrece una gran cantidad de funcionalidades para elaborar aplicaciones web de manera más rápida y sencilla. Sin embargo, para este proyecto se ha utilizado la versión gratuita de Dash, con algunas librerías adicionales entre las que se encuentran:

- **Dash Core Components:** Ofrece una amplia gama de componentes interactivos como gráficos, tablas, sliders, dropdowns, entre otros, que permiten a los usuarios interactuar con los datos de manera intuitiva.
- **Dash HTML Components:** Ofrece una amplia gama de componentes HTML como divs, spans, inputs, entre otros, que permiten personalizar la apariencia y el diseño de la aplicación web.
- **Dash Mantine Components:** Ofrece una amplia gama de componentes de Mantine como navbars, tarjetas, modales, entre otros, que permiten crear aplicaciones web atractivas y responsivas.
- **Dash Express:** Ofrece componentes extra como un panel de filtrado y una barra de navegación.²
- **Plotly Express:** Es el principal competidor de matplotlib en el mundo de la visualización de datos en Python. Ofrece una gran cantidad de gráficos y visualizaciones de datos interactivos.

4.4.2. Diseño

El diseño de la aplicación web se ha basado en la simplicidad, la usabilidad y la accesibilidad. El primer diseño del proyecto se realizó en la aplicación de prototipado Figma como se muestra en la Figura 4.3.

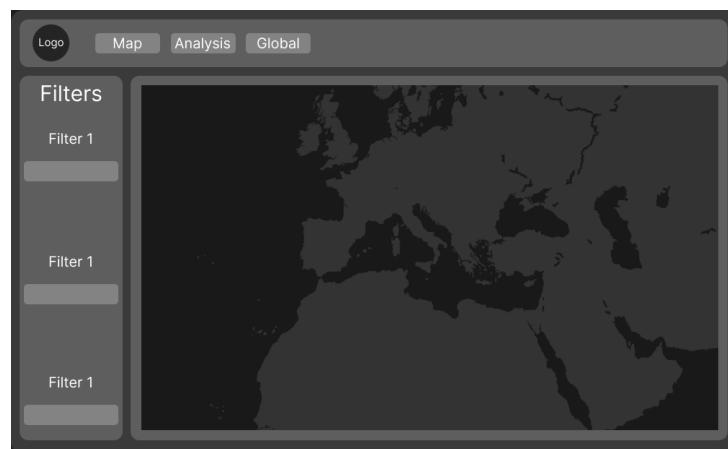


Figura 4.3: Primer diseño de la aplicación web en Figma.

Este diseño se ha ido modificando y adaptando a lo largo del desarrollo del proyecto para mejorar la usabilidad y la experiencia del usuario. Este diseño muestra la pantalla principal de

²Esta librería se ha modificado para adaptarla a las necesidades del proyecto.

la aplicación, que incluye un mapa con las estaciones APRS, y un panel de control con filtros y opciones de visualización.

4.4.3. Home - Primera pantalla

En esta sección se describirá la primera pantalla de la aplicación web, que muestra un mapa con las estaciones APRS y un panel de control con filtros y opciones de visualización.

Obtención de datos

Mapa de estaciones APRS

El mapa de estaciones APRS muestra la posición de cada una de las estaciones APRS, representadas por un marcador en el mapa. Es una mapa interactivo que permite hacer zoom, desplazarse y al posicionar el cursor encima de una estación aparecerá un recuadro ofreciendo información adicional.

Esta página hace uso de la librería plotly para renderizar el mapa y del servicio web de mapas Mapbox para obtener los mapas y estilos de mapa.

Filtros

El panel de control de la aplicación web incluye una serie de filtros y opciones de visualización que permiten al usuario personalizar la información mostrada en el mapa. La ventaja de estos filtros es que son aditivos y se pueden combinar para obtener con exactitud la información requerida. Los filtros disponibles son:

- **Filtro por Rango de Fechas:** Permite filtrar las estaciones APRS por rango de fechas, mostrando solo las estaciones de las que se ha recibido algún mensaje en el rango de fechas seleccionado.
- **Filtro por País:** Permite filtrar las estaciones APRS por país, mostrando solo las estaciones que se encuentran en el país seleccionado.
- **Filtro por contenido del mensaje:** Este es quizás el filtro más interesante ya que permite filtrar las estaciones por el contenido del mensaje, este filtro será explicado más adelante.
- **Filtro por Tipo de Estación:** Permite filtrar las estaciones APRS por tipo, mostrando solo las estaciones que corresponden al tipo seleccionado.
- **Filtro por ssid:** Permite filtrar las estaciones por ssid.

Búsqueda difusa en contenido de mensajes

Este filtro es personalmente el más interesante de todos los filtros disponibles. El objetivo es permitir al usuario buscar mensajes emitidos por las estaciones que contengan una palabra o frase específica. Para ello se han probado varias alternativas como el uso de expresiones regulares o una implementación en python puro. El problema de estas opciones es que son muy lentas para el enorme volumen de datos con el que se cuenta y al ser dinámico no era factible realizar esa búsqueda en tiempo real. Finalmente se ha optado por el uso de Postgres FTS (Full Text Search). Postgres FTS es un sistema de búsqueda de texto completo que permite realizar búsquedas de

texto en grandes volúmenes de datos de manera eficiente. Las ventajas que ofrece este sistema son:

- **Rendimiento:** Todo el proceso de búsqueda se realiza en la base de datos, lo que permite realizar búsquedas de texto sin transferir todos los mensajes al servidor para su búsqueda.
- **Búsqueda difusa:** Permite realizar búsquedas de texto difuso, lo que significa que se pueden buscar palabras o frases que contengan errores tipográficos o que no coincidan exactamente con el texto buscado.
- **Indexación:** Postgres FTS indexa automáticamente los mensajes de texto, lo que permite realizar búsquedas de texto de manera eficiente incluso en grandes volúmenes de datos.

Se ha creado un índice de texto completo en la columna de mensajes de la tabla de mensajes y se ha utilizado la función `to_tsvector` para convertir los mensajes en un vector de texto completo. Posteriormente se ha utilizado la función `plainto_tsquery` para convertir la palabra o frase de búsqueda en una consulta de texto completo y finalmente se ha utilizado la función `ts_query` para realizar la búsqueda de texto completo en la columna de mensajes. Este filtro permite al usuario buscar mensajes emitidos por las estaciones que contengan una palabra o frase específica, lo que facilita la identificación y el análisis de los mensajes relevantes. Cuando un usuario realiza una búsqueda de texto completo, se muestra un modal con la lista de todos los mensajes que contienen la palabra o frase de búsqueda, ordenados por relevancia haciendo uso de la función `ts_rank`. Otra funcionalidad interesante es que la palabra o palabras buscadas se resaltan en el mensaje para facilitar su identificación como se muestra en la Figura 4.4.

Station	Comment
K4SRC	Jay Hospital Emergency
K4SRC	Santa Rosa Hospital Emergency
K4SRC	Jay Hospital Emergency

Figura 4.4: Reultados de la búsqueda de «emergency hospital».

En el modal se muestra una tabla de dos columnas, la columna izquierda contiene un botón con el nombre de la estación y la columna derecha el contenido del mensaje.

4.4.4. Station - Segunda pantalla

Esta es la página que hace única a APRSINT. En esta página se muestra la información detallada de una estación APRS en concreto. Se puede llegar a esta pantalla desde la página principal haciendo clic en un marcador de una estación en el mapa, haciendo click en una estación en la búsqueda en los mensajes o haciendo click en un nodo del grafo de la página 3. La pantalla station esta dividida en tres secciones.

Perfil de QRZ

QRZ es una base de datos de radioaficionados que contiene información detallada sobre los radioaficionados de todo el mundo. Para acceder a la información disponible en la web qrz es necesario tener una cuenta y estar registrado. QRZ cuenta con un servicio de API que permite acceder a la información de los radioaficionados de manera programática. Para mantener APRSINT fiel a los principios del OSINT (fuentes abiertas) se ha optado por no utilizar la API e intentar otra opción.

Utilizando una cuenta normal gratuita en QRZ, se permiten realizar hasta 25 visualizaciones de perfiles diarias en su sitio web. Con el objetivo de aumentar la cantidad de visualizaciones disponibles, se ha optado por implementar web scraping mediante un sistema propio de rotación de cuentas gratuitas y utilizando la librería *BeautifulSoup*. Como parte de esta estrategia, se ha creado una tabla en la base de datos para almacenar en caché las solicitudes, lo que reduce el número de solicitudes a una misma estación si esta ya ha sido consultada previamente.

Este módulo es de los más complejos de la aplicación pero a su vez de los más útiles ya que permite al usuario obtener información muy detallada de una estación APRS en concreto y de la persona que está detrás. Algunos de los datos que se pueden obtener son:

- Nombre de la persona registrada
- Fechas de registro y última actualización en la web
- Alias conocidos de la estación
- Dirección de la persona registrada (Como link a google maps)
- Fecha de nacimiento de la persona registrada

4.4.5. Graph - Tercera pantalla

Esta pantalla es la última de la web, en ella se muestra un grafo dirigido en el que los nodos son las estaciones y las aristas dirigidas, mensajes como se muestra en la Figura 4.5. El grafo es interactivo y permite hacer zoom, desplazarse y al posicionar el cursor encima de un nodo o arista aparecerá un recuadro ofreciendo información adicional. El color y tamaño de los nodos depende de la cantidad de mensajes que ha emitido o recibido la estación y de misma manera el color y tamaño de las aristas depende de la cantidad de mensajes que se han emitido o recibido desde la estación origen y destino.

Las características que se buscaban en la solución final eran:

- **Interactividad:** Se buscaba un grafo interactivo que permitiera al usuario explorar las relaciones entre las estaciones APRS.
- **Rendimiento:** Quizás la característica más importante, el grafo debía ser capaz de mostrar una gran cantidad de estaciones y conexiones sin afectar al rendimiento de la aplicación.
- **Personalización:** Se buscaba un grafo que permitiera personalizar la apariencia y el comportamiento de los nodos y aristas según las características de cada estación y mensaje.

Para alcanzar la solución final que se presenta ahora en la página, se han probado muchas opciones que se han ido descartando por diferentes motivos.

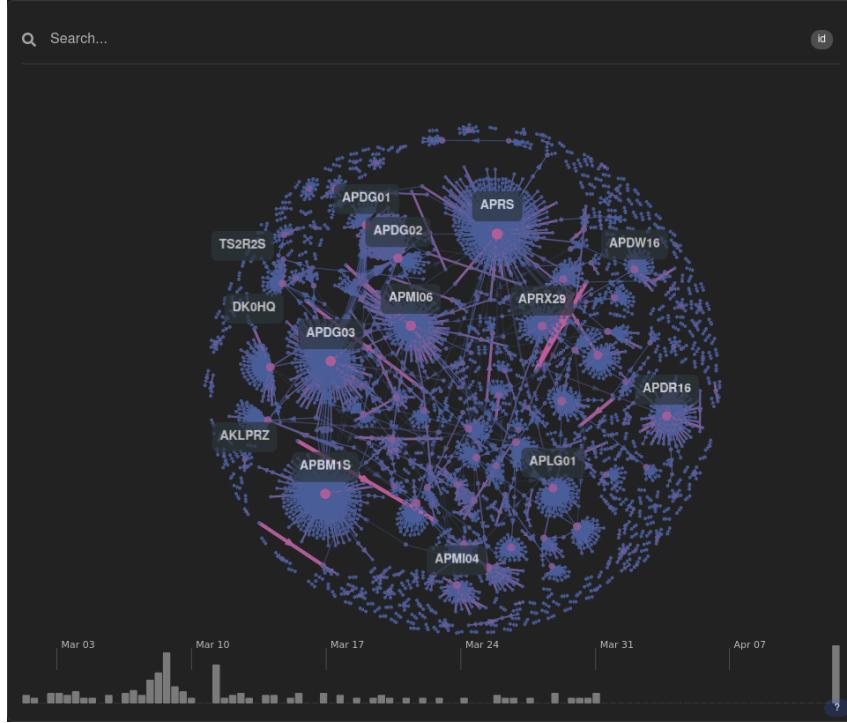


Figura 4.5: Grafo de estaciones APRS.

Alternativas probadas

En un primer momento se intentó utilizar la librería **networkx** de Python para la creación del grafo, sin embargo, se descartó debido a su rendimiento y a su falta de interactividad. Posteriormente se intentó utilizar la librería **Cytoscape** que es la solución estandard que ofrece Dash, esta ofrece una gran cantidad de funcionalidades para la creación de grafos interactivos, sin embargo, se descartó de nuevo debido a su pobre rendimiento.

Una de las alternativas que también se probó fue la librería **Sigma JS** que ofrece una gran cantidad de funcionalidades y personalización para la creación de grafos interactivos. Esta librería cuenta con una funcionalidad interesante *forceAtlas2* que permite calcular mediante una simulación de fuerzas la posición de los nodos y aristas en el grafo. Sin embargo, de nuevo se descartó debido a su rendimiento con una gran cantidad de nodos y aristas.

Finalmente se optó por la librería **Cosmograph JS**. Cosmograph utiliza la libreria *cosmos* de calculo de posiciones mediante la gpu y por tanto puede manejar un gran numero de nodos y aristas.

El grafo

El problema principal con Cosmograph ha sido la falta de documentación debido a que es bastante nueva y la dificultad de integrarla con Dash. Esta libreria esta publicada en npm y se ha tenido que utilizar bundle.js para convertir el código de JavaScript en un solo fichero que se pueda importar en Dash. Ha sido necesario modificar ligeramente la libreria para terminar de integrarla con Dash.

Se han utilizado tres componentes de Cosmograph JS para la creación del grafo, el componente *Cosmograph* que crea el grafo, el componente *CosmographSearch* que permite mediante una barra de búsqueda encontrar una estación y el componente *CosmographTimeline* que permite filtrar el grafo por rango de fechas.

Para mejorar el rendimiento general de la web se ha utilizado la técnica de lazy loading, que consiste en cargar los componentes del grafo solo cuando el usuario presiona el botón *Load graph*. En el momento que el usuario presiona el botón, se activa un callback de cliente de Dash. El callback primero lee los datos del grafo de un fichero CSV que se actualiza diariamente mediante Apache Airflow. Posteriormente se crean los nodos y las aristas y se inicializa el grafo, la barra de búsqueda y la barra de tiempo. Seguidamente se establecen los parámetros de la simulación de fuerzas y se inicia la simulación. Por último se establecen las propiedades visuales y funcionales de los nodos y aristas.

Cuando el usuario arrastra el cursor sobre un nodo, se muestra un recuadro con el nombre de la estación. Cuando el usuario hace clic en un nodo, se le redirige a la página **station** de la estación seleccionada.

Capítulo 5

Propuesta de la Idea

5.1. Implementacion y arquitectura

Aqui hablo de cosas

Capítulo 6

Conclusiones y trabajo futuro

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Conclusions and future work

Translate the previous chapter