TECNOLÓGICO DE ESTUDIOS SUPERIORES DE ECATEPEC

DIVISION DE INGENIERIA EN SISTEMAS COMPUTACIONALES

SISTEMA EMBEBIDO AEREO PARA TRANSMITIR Y REPLICAR DATOS GEOESPACIALES CIFRADOS POR SMS

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN SISTEMAS COMPUTACIONALES

P R E S E N T A N:

IVÁN AYALA BRITO

LESLY GUADALUPE GARCÍA LÓPEZ

DIRECTORA: GRISELDA CORTÉS BARRERA

ECATEPEC DE MORELOS, EDO. DE MÉXICO, 2021.

Agradecimientos

Agradezco a mi familia por apoyarme en todo momento, en especial a mis papás que siempre han estado conmigo en cada uno de mis logros y por todo lo que me han dado. También quiero agradecer a mis compañeros y amigos en especial a mí amiga Lesly que estuvo en todo momento conmigo apoyándome en la carrera, desde el inicio hasta el final.

De igual manera quiero agradecer a mis profesores por su dedicación y tiempo, así como todo el conocimiento que me brindaron, ya que con su apoyo y enseñanzas he conseguido terminar la carrera en ingeniería en sistemas computacionales. También darle las gracias a mi tutor de tesis, por cada una de las observaciones en mi proceso de titulación.

Gracias a todos por acompañarme en el transcurso de la carrera.

Iván Ayala Brito

A mi madre que ha dedicado su tiempo, fuerza y amor para que yo me convierta en la persona que soy ahora. Mi padre que me ha enseñado con su gran ejemplo a esforzarme para alcanzar cada una de mis metas. Les agradezco por ser mi pilar fundamental y por haberme apoyado incondicionalmente, pese a las adversidades e inconvenientes que se presentaron.

Agradezco a mis hermanos por enseñarme a enfrentar los problemas con optimismo, por estar presentes aportando cosas buenas en mi vida y apoyarme en todo momento.

A mi mejor amigo Iván por estar conmigo a lo largo de la carrera, mostrando siempre su lealtad y apoyo ante cualquier situación. A mis maestros, en especial a la Dra. Griselda Cortés Barrera, por guiar este proyecto y formar parte de otro objetivo alcanzado.

Lesly Guadalupe García López

Resumen

Este proyecto tiene como finalidad automatizar la transmisión de datos geoespaciales y asegurar su integridad, además de desarrollar un micro servidor que contenga dichos datos y estos puedan ser sincronizados a otros ordenadores utilizando la herramienta Bucardo, por esta razón se desarrolló una aplicación en Java donde se implementa un algoritmo de codificación AES. Para la transmisión de los datos se utilizó un módulo GSM SIM800L. Empleando la metodología de desarrollo de prototipos se llevó a cabo este proyecto y como resultados se obtuvieron los mensajes cifrados que fueron transmitidos de forma automática a través de un SMS, recupera y guarda cien por ciento de la información transmitida en la base de datos geoespacial, también se obtuvo como resultado la sincronización entre dos bases de datos con la información almacenada.

Contenido

índice de Figuras

Índice de tablas

Índice de graficas

Capitulo I
Introducción

Este capítulo aborda temas generales sobre encriptación de datos, uso de transceptores para la transmisión de información y la gestión de datos geoespaciales, además, se detalla información del proyecto a desarrollar como la problemática que presenta el cliente y la solución justificada, así como también los alcances, limitaciones y objetivos del proyecto.

1.1 Antecedentes

Algoritmos de codificación

Montenegro D., 2020 en el trabajo titulado “Comparación de algoritmos de encriptación para la transferencia de archivos en mensajería instantánea” proporciona un análisis de comparación de los algoritmos criptográficos para resolver el problema fundamental que está en garantizar la seguridad e integridad de los datos al momento de conectarnos en una gran red de objetos físicos, utilizando una metodología experimental se obtiene como resultado que el algoritmo Advanced Encryption Standard (AES) fracciona los datos en menor cantidad de paquetes necesitando menos tiempo para remitirlos comparándolos con otros algoritmos. En paquetes encriptados el algoritmo AES muestra mayor cantidad de paquetes encriptados y la cantidad de paquetes no encriptados es menor [1].

En el proyecto llamado “Análisis y comparación de los algoritmos de cifrados simétrico en plataformas Windows”, Carrera T.,(2016) proporciona la comparación de rendimiento entre dos de los algoritmos de cifrado más utilizados: AES (Rijndael) y Blowfish Análisis, para lograr el objetivo utiliza la metodología experimental, determinando así la influencia del sistema operativo en la velocidad de cada uno de los algoritmos elegidos, se ha concluido que Windows 10 acelera significativamente el cifrado y descifrado de archivos ya sea con AES o Blowfish [2].

Comunicación vía celular con SMS

En el trabajo titulado “sistema de monitoreo de una red de buses de transporte público e información para usuarios empleando transceptores GPS/GSM” realizado por J. Meza y Leaño Víctor en 2017 se resuelve el problema de monitoreo de los autobuses de una red de transporte público integral mediante el empleo de transceptores, se demostró mediante la implementación del sistema y las pruebas

realizadas que es posible la realización de un sistema de monitoreo, empleando comunicación GSM [3].

Chiapella M., 2020, “Instrumento de medición de factores climáticos”, resuelve la problemática de automatizar la lectura de factores climáticos y enviarlos mediante un SMS. Para realizar el instrumento de medición se utilizó un módulo SIM800L, un microcontrolador PIC 16F877A, en cuanto a programación se utilizó lenguaje C. Se obtiene como resultado medir y llevar un registro de distintos factores climáticos, logra comunicación inalámbrica a través de mensajes de textos (SMS). Se obtiene información de la cantidad de lluvia acumulada en el vaso de precipitado, como también la temperatura y humedad relativa, además todos los datos son obtenidos y mostrados instantáneamente [4].

En el año 2020, E. Llanos presenta el trabajo titulado “Diseño de un sistema inalámbrico de monitoreo para pacientes epilépticos de la clínica Angloamericana” donde utiliza módulos de sensores y un módulo SIM800L que ayudará a predecir un ataque epiléptico, enviando la alerta hacía el personal asistencial mediante un SMS al dispositivo móvil. El diseño de un sistema portátil de monitoreo para pacientes epilépticos mediante módulos de adquisición de síntomas sensoriales indicó positivamente la predicción de los episodios epilépticos mediante los síntomas sensoriales logrando definir los principales síntomas para los pacientes. También se aprovecha el almacenamiento de mensajes como historial de tiempo para determinar el momento en que se produce el episodio epiléptico [5].

Gestión de datos

“Aplicación web mapping para la visualización y consulta de información en la gestión municipal a partir de herramientas de código abierto” investigado por Ana Cecilia Lara Barrera en el año de 2017, solucionando la siguiente problemática, los gobiernos locales del país se han apoyado en la utilización de herramientas como los Sistemas de Información Geográfica para contribuir a la toma de decisiones en

los procesos de planificación y gestión ambiental del territorio mediante el análisis de información, con la metodología, secuencial donde se usó las siguientes herramientas, OpenGeo Suite, Geoserver, QGIS, PostgreSQL, PostGIS, Spring [6].

Carlos Raúl Montaño Espinosa en el año 2016, en su trabajo titulado “Diseño e implementación de una geobase de datos distribuidas, en un ambiente virtualizado, acoplada a un sistema de información geográfica”, resuelve la problemática de Las atribuciones y responsabilidades conferidas a la CONAGUA por la LAN, la gran cantidad de información ligada a su ubicación geográfica, la complejidad de los análisis requeridos para la toma de decisiones y la necesidad de conocer mejor las fases del ciclo hidrológico en los que la componente geográfica juega un papel primordial, hacen indispensable la vinculación de las bases de datos de la CONAGUA, los cuales incluyan la dimensión geográfica a fin de unificar datos y permitir el desarrollo en forma modular distribuida de la consulta de datos a nivel nacional, donde su principal objetivo es Implementar, mediante un marco metodológico y de actuación, una estructura, en red, de un sistema de bases de datos distribuido con al menos 3 geobases de datos para los respectivos organismos de Cuenca sobre el mismo número de servidores virtuales, a nivel central o local, vinculando adicionalmente a la geobase de datos central con al menos 3 de los sistemas de información más importantes para la Subdirección, mediante la metodología diseño de bases de datos distribuidas, junto con las herramientas de Virtualización de hardware, S.O, ASV, red, aplicaciones y bases de datos [7].

En el año 2017, con el tema de “Desarrollo e implementación de una visualizador web geográfico sobre aguas termales en el estado de México” investigado por Andy Mejía Olivares, resolvió el siguiente problema para cumplir las expectativas del CIRA, generando un medio que pueda gestionar los datos tanto geoespaciales como alfanuméricos, con un visualizador web cartográfico a través de servicios de mapas WMS, en el cual los datos estén almacenados, controlados y administrados en una base de datos central que funcione como proveedor de información a los

usuarios mediante un sistema distribuido libre de inversiones, con las siguientes herramientas como, OpenGeo Suite, Geoserver, QGIS, PostgreSQL, PostGIS, Spring. Usando la siguiente metodología en su proyecto “Componentes y flujo de procesos, UML, Caso de usos, Actores, secuencias [8].

En el año 2018, Flor Radilla López en su trabajo titulado “Modelado de datos para base de datos espaciales. Caso de estudio: sistemas de información geográfica”, usa la siguiente metodología de Modelo de datos, caso de estudio, diagrama de UML (Paquetes y Clases), el cual se traduce en procedimientos pragmáticos de implementación de software. La metodología inicia con la etapa de recopilación y análisis de requerimientos en donde se establece el modelo de empresa, pasando posteriormente por etapas de construcción de ecodiseño y refinamiento a través de los distintos niveles y diferentes orientaciones, para desarrollar una metodología y un sistema para automatizar el diseño conceptual de bases de datos y la creación de sistemas a partir de descripciones XML. Con base en modelos de desarrollo de software, estándares de información geográfica y técnicas de modelos de datos, se desarrollarán modelos conceptuales de bases de datos y sistemas con los siguientes componentes: estructural/semántica, interfaz de usuario, operacional/comportamiento con las siguientes herramientas como, API Java Swing, XML, PostgreSQL, PostGIS y NetBeans, logró su objetivo en resolver la problemática en su determinado tiempo [9].

1.2 Planteamiento del problema

El principal problema que presenta el cliente es la forma en la que monitorean actualmente sus aeronaves, ya que los controladores solo reciben la información del punto al que llegó el piloto y la hora en que lo hizo, posteriormente reciben el punto al que se dirige y en cuanto tiempo se estima su llegada ya que para un piloto no es posible enviar sus coordenadas mientras está volando. Debido a esto, el tiempo que tardan los controladores en recibir información sobre las aeronaves se prolonga, esto hace que el sistema actual sea obsoleto.

Además, no se cuenta con ningún método para resguardar la información y asegurar la integridad de los datos. Se sabe que el medio de comunicación más seguro y con mejor cobertura en aeronaves es por satélite, sin embargo, este medio resulta muy costoso.

Los aviones realizan algunas misiones, donde recolectan paquetes de datos, mediante transceptores y estos datos deben de almacenarse en un servidor que soporte datos geoespaciales, actualmente la empresa no cuenta con un servidor que almacene y administre estos datos.

Otro problema que tenemos es el poder sincronizar estos datos del servidor a una base de datos local cada cierto tiempo, donde cada uno debe de estar en distintas computadoras y el servidor debe de soportar hasta 100 conexiones.

Debido a lo anterior, se requiere automatizar la transmisión de los datos geoespaciales en tiempo de vuelo enviando las coordenadas de las aeronaves mediante un SMS de forma automática, cifrando los datos para asegurar la información de posibles ataques y crear un micro servidor que almacene los datos y sea capaz de sincronizar toda la información resguardada a otras bases de datos donde se pueda leer y manipular la información. También, es importante contemplar una Base de Datos geoespacial.

1.3 Objetivo general

Desarrollar un módulo que permite codificar y enviar paquetes de datos geoespaciales a través de un SMS utilizando un transceptor celular. Además, de crear un micro servidor que realice una sincronización con los datos mediante un administrador de bases de datos.

1.3.1 Objetivos específicos

- Analizar los requerimientos del cliente, así como investigar los temas pertinentes para dar solución al problema.

- Diseñar y representar la propuesta de solución al problema mediante diagramas.
- Desarrollar el software para el cifrado de mensajes, el circuito lógico para la trasmisión, así como la instalación de la base de datos espacial con las herramientas necesarias que permita la interacción, manipulación y sincronización de la información almacenada.
- Realizar pruebas del funcionamiento del prototipo en el cifrado de mensaje, la comunicación de la interfaz en java con el circuito lógico y verificar que la base de datos sea capaz de soportar, sincronizar y manipular un paquete de datos geoespacial, realizando mejoras y correcciones para un mejor funcionamiento.
- Comprobar la transferencia y almacenamiento de los datos entre la aplicación móvil y la base de datos, configurando y realizando una sincronización con otra base de datos, para la manipulación de los datos.

1.4 Justificación

El motivo que nos llevó a investigar y desarrollar el presente trabajo es la ausencia de herramientas tecnológicas para la transmisión, seguridad y almacenamiento de los paquetes de datos de las aeronaves, automatizando el envió de los datos geoespaciales en tiempo de vuelo mediante un SMS, utilizando un algoritmo de codificación para asegurar la información de posibles ataques y poder resguardar la información del sistema de control de mando principal, así como incrementar la conciencia situacional del personal involucrado en las operaciones aéreas militares, para una mejor toma de decisiones.

En el presente proyecto se evidencia el impacto tecnológico con la dotación de infraestructura requerida (equipo de cómputo, placa microcontrolador, módulo SIM800L), la implementación de un algoritmo de codificación en el desarrollo de software, así como almacenamiento y sincronización de datos geoespaciales, encaminado a facilitar y optimizar los procesos para mejorar la recuperación de datos.

Se utilizará el módulo SIM800L debido a que es un módulo económico y práctico que cumple con la función de enviar SMS. Para lograr el envío de los datos, el módulo necesita estar conectado a una placa microcontrolador, en este proyecto se utilizará Arduino uno por ser una placa económica y de plataforma abierta.

El software para este proyecto será desarrollado en lenguaje Java por ser un lenguaje multiplataforma y con el cual podremos implementar el algoritmo de codificación Advanced Encryption Standard (AES). Una de las ventajas de AES es que utiliza la misma clave para el cifrado y descifrado de los datos ya que es un algoritmo simétrico, además, cuenta con un gran número de claves posibles, este número es mayor al que ofrece cualquier otro algoritmo de codificación, otro aspecto destacable es el tamaño de los bloques y el número de rondas que ofrece, esto garantiza mayor seguridad.

Por otro lado, la base de datos será desarrollada en un lenguaje SQL. PostgreSQL realiza la función de servidor y con ayuda de la herramienta PostGIS, nos permitirá crear una base de datos geoespacial, también necesitamos de la herramienta de Bucardo ya que PostgreSQL no puede realizar sincronizaciones por sí solo, sino que necesitamos de esta herramienta para sincronizar los datos a otras bases de datos.

1.5 Hipótesis

Los datos geoespaciales de las aeronaves serán enviados como SMS de manera automática y segura cada 10 segundos y recibidos inmediatamente en el celular de destino, siendo almacenados y sincronizados a más de 100 computadoras en la base de control aérea.

1.6 Aporte

La presente investigación mostrará un prototipo para la recolección, transmisión y almacenamiento de paquetes de datos geoespaciales de las aeronaves, cumpliendo cada uno de los requisitos del cliente, mediante un sistema que envía

un paquete de datos codificados a través de un SMS para su posterior decodificación mediante una aplicación móvil y este a su vez se enviará a una base de datos para su almacenamiento y replicación a otras computadoras.

1.7 Contenido

El capítulo l con el título Introducción, muestra subtemas de investigación como los antecedentes, la problemática que se le presenta a nuestro cliente, los objetivos que se llevaron a cabo, la justificación, los alcances y límites e hipótesis.

En el capítulo II denominado marco teórico, se presentan los temas de investigación para el desarrollo del prototipo, entre los más relevantes se encuentran, sistemas de información geoespacial, replicación de bases de datos con PostgreSQL, integración en proyectos Arduino en java con PanamaHitek, sistemas criptográficos simétricos y transceptor celular.

En el capítulo lll Método de solución de prototipo, se abordan los temas necesarios para la elaboración del presente proyecto, los cuales son análisis de requerimientos, diseño, construcción y desarrollo del prototipo.

En el capítulo IV llamado resultados obtenidos con las pruebas del prototipo se muestran los siguientes subtemas: evaluación y modificación del prototipo y pruebas de funcionamiento, los cuales detallan los resultados obtenidos en el prototipo final.

En el capítulo V muestran las conjeturas a las que se llegaron al final del proyecto.

Capitulo ll
Marco teórico

En este capítulo se abordarán los diferentes conceptos involucrados en la presente investigación para la generación de una aplicación de escritorio en Java, la conexión de dicha aplicación con Arduino y el módulo SIM800L para la transmisión de los datos. También se presentan los temas relacionados con el almacenamiento de la información en una base de datos geoespacial y su replicación.

2.1 Sistemas de información geoespacial

2.1.1 Antecedentes de un SIG

En los años 70 surgieron como iniciativas individuales aisladas y se impone una actitud corporativa en su desarrollo. En los años 80 surge su fase comercial, su uso se hizo más frecuente gracias al desarrollo de una tecnología informática adecuada, que fomento la aparición de productos SIG *(Sistema de Información Geográfico)* en el mercado. Hoy en día, se están produciendo fuertes inversiones en el desarrollo de base de datos de información geográfica y en sistemas SIG, apoyados en las nuevas TIC's [10].

2.1.2 Base de datos geoespaciales

Para definir una base de datos es importante tener claro qué es un dato e información debido a que estos elementos son fundamentales para el desarrollo de las bases de datos, según [11]:

- *Dato:* es un conjunto de caracteres con algún significado, pueden ser numéricos, alfabéticos, o alfanuméricos, esta es la unidad mínima de información. Un dato dentro de una base de datos responde a la función (objeto, atributo, valor).
- *Información*: es un conjunto ordenado de datos los cuales son manejados según la necesidad del usuario, para que un conjunto de datos pueda ser procesado eficientemente y pueda dar lugar a información, primero se debe guardar lógicamente en archivos.

La información es el recurso más valioso en una base de datos, por tanto, esta debe ser [12]:

- *Accesible*: es la facilidad y rapidez para poder acceder a ella
- *Clara:* debe ser íntegra y fácil de entender
- *Precisa:* lo más exacta posible

- *Propia:* Debe haber la mayor similitud entre el resultado creado y lo que el usuario pide
- *Oportuna:* El proceso de entrada-procesamiento-entrega al usuario debe ser en el menor tiempo posible
- *Flexible*: la información se puede adaptar a la toma de decisiones que mejor convenga
- *Verificable*: la información debe ser totalmente fiable para que se pueda verificar en el momento deseado
- *Imparcial:* La información debe poder modificarse tanto por el administrador, como por el usuario dueño de la base
- *Cuantificable:* la información puede ser el resultado de cualquier dato procesado.

Una base de datos es un conjunto de datos almacenados entre los que existen relaciones lógicas y que ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización [13].

Una base de datos es un conjunto de datos que pertenecen al mismo contexto, almacenados sistemáticamente para su posterior uso, es una colección de datos estructurados según un modelo que refleje las relaciones y restricciones existentes en el mundo real [14].

Una base de datos espacial permite el almacenamiento de las geometrías de un archivo cartográfico dentro de una base de datos, de modo que podamos almacenar y analizar estos datos de un modo más eficiente [15].

2.1.3 Tipos de bases de datos geoespaciales

Un motor de base de datos es el servidor principal para almacenar, procesar datos. Proporciona, además acceso controlado y procesamientos de transición para cumplir con los requisitos de las aplicaciones [16].

MySQL cuenta con la extensión *MySQLSpatial*, que es una base de datos encontrada por defecto en plataformas de hosting y por lo tanto es muy usada por desarrolladores web. *MySQL* implementa extensiones espaciales como un

subconjunto del entorno *SQL with Geometry Types*. No es necesario habilitar las funciones espaciales *(PostGIS)*, puesto que en *MySQL* ya se encuentran habilitadas por defecto. Las extensiones espaciales de *MySQL* permiten la generación, el almacenamiento y el análisis de datos geográficos [17]:

- *MySQL* tiene tipos de datos para representar valores espaciales que corresponden a las clases de *OpenGIS*. Estos tipos de datos permiten representar geometrías como puntos, líneas o polígonos.
- Funciones para manipular valores espaciales. Estas funciones permiten trabajar con los formatos espaciales y realizar operaciones espaciales.
- Creación de índices espaciales para mejorar el tiempo de ejecución de las consultas espaciales.

SQLite cuenta con la extensión *Spatialite*, que es una base de datos muy simple basada en ficheros. *SpatiaLite* es un motor de base de datos de *SQLite* con funciones especiales añadidas además es un Sistema Gestor de Bases de Datos (DBMS, por sus siglas en inglés) que es simple, robusto, fácil de usar y ligero. Cada base de datos *SQLite* es simplemente un archivo. Se puede copiar, comprimir y portar entre Windows, Linux, MacOs, etc. Algunas características son [18]:

- Funciones SQL espaciales como: *AsText()*, *GeomFromText()*, *Area()* y *PointN()*.
- El conjunto completo de funciones *OpenGis* es soportado a través de GEOS, incluyendo *Overlaps(), Touches(), Unión() y Buffer().*
- Metadatos espaciales completos conforme a las especificaciones de *OpenGis*
- Numerosos formatos geométricos - EWKT, GML, KML, and GeoJSON.
- Importación y exportación de archivos shape.

PostgreSQL cuenta con la extensión de *PostGIS*, que es una potente base de datos multiplataforma es totalmente compatible con los estándares ofrecidos por el *Open Geospatial Consortium* (OGC), es decir, *PostGIS* soporta datos espaciales. La extensión *PostGIS* habilita el soporte para trabajar con objetos geográficos localizados en el espacio. En otras palabras: convierte *PostgreSQL* en una base de

datos espacial, que en la práctica funciona (quitando el apartado gráfico) exactamente como un auténtico Sistema de Información Geográfica de escritorio. *PostGIS* permite trabajar tanto con información geográfica vectorial como raster de múltiples formatos distintos, de esta forma trabaja con formatos vectoriales que permite [19]:

- crear y editar geometrías
- establecer relaciones espaciales entre elementos geométricos
- realizar análisis de distancia y enrutamiento con *pgRouting*
- realizar correcciones topológicas de elementos, etc.

2.2 Replicación de BD con PostgreSQL

PostgreSQL es un sistema de base de datos objeto-relacional de código abierto creado por el departamento de ciencias de la computación de *Berkeley* de la universidad de California, su arquitectura posee una gran reputación gracias a su fiabilidad, integridad de los datos y buen funcionamiento, la replicación en *PostgreSQL* es la transmisión de información derivada de las operaciones DML de una Base de Datos a otra, es decir, se transmite a otra Base de Datos las instrucciones *INSERT*, *UPDATE* y *DELETE* que se realicen en una Base de datos de forma que ambas BBDD tengan la misma información, así pues, se obtiene una redundancia de datos [21].

Puede instalarse en un gran número de sistemas operativos como es Linux, Unix y Windows.

Entre las soluciones de replicaciones más conocidas se encuentra:

- Almacenamiento compartido
- Replicación en Streaming
- Slony
- Bi Direccional
- Bucardo

Almacenamiento compartido.

Mediante esta opción, se evitan costes de sincronización, ya que solo existe una copia de la base de datos. Funciona mediante un array de discos compartidos por varios servidores, si la base de datos principal falla, el servidor *standby* reacciona y monta y arranca la base de datos automáticamente, esto permite un rápido *failover* sin pérdida de datos. Esta es una solución muy común en el almacenamiento en red. La desventaja que tiene este método se manifiesta si el fallo se origina en el disco compartido, convirtiendo tanto la copia primaria como la secundaria en inservibles. Mientras el servidor primario esté en funcionamiento, el servidor *standby* no podrá acceder a los datos [22].

Replicación Streaming

La Replicación *Streaming* es el sistema nativo de *PostgreSQL* de replicación, este tipo de replicación sólo puede ser de estructura Maestro-Esclavo. En la replicación *Streaming*, la copia standby se conecta con la copia primaria, la cual envía sus registros WAL a la *standby* según los genera. Las copias *standby* eliminan periódicamente los WAL recibidos, para evitar una carga excesiva del disco. La replicación *streaming* de *PostgreSQL* es asíncrona por defecto. Si el maestro falla, algunas de las transacciones realizadas, puede que no se hayan replicado a la copia *standby*, provocando una pérdida de datos. La cantidad de datos perdidos será proporcional al tiempo de retraso de la replicación durante el fallo. En el modo de replicación asíncrono, cada *commit* de una transacción de escritura esperará hasta recibir la confirmación de que la transacción ha realizado *commit* en todas las copias, salvo en las transacciones pesadas, como cargas masivas de datos o construcción de índices [23].

Replicación Slony

Slony es un módulo de replicación de transacciones Maestro-Esclavo asíncrono, para PostgreSQL que admite la conmutación en cascada, por ejemplo, un nodo puede alimentar a otro nodo que alimenta a otro nodo. Su funcionamiento se basa

en los *triggers* e incluye todas las características y capacidades necesarias para replicar grandes bases de datos a un número razonablemente limitado de sistemas esclavos. Slony es un sistema diseñado para su uso en centros de datos y sitios de respaldo, donde el modo normal de operación es que todos los nodos estén disponibles. Su nombre proviene del ruso, significa elefante, un claro guiño al logo de *PostgreSQL* [24].

Replicación Bi Direccional

Bi Directional Replication (DBR) es un sistema de replicación asíncrono multimaestro de código libre para *PostgreSQL*, especializado en bases de datos distribuidas geográficamente, utilizando una replicación lógica asincrónica eficiente y soportando cualquier nodo con más de 2 a 48 nodos en la base de datos distribuida. BDR no depende del uso de disparadores para la detección de modificaciones y la posterior replicación de estas, además que la gestión de conflictos es muy flexible, de forma predeterminada BDR, proporciona un algoritmo para a gestión de conflictos, también es posible escribir su propio algoritmo de resolución de conflictos, utilizando un procedimiento almacenado por parte del servidor [25].

Replicación Bucardo

Sistema de código libre de replicación asíncrona basada en triggers, que admite tanto la replicación multimaestro como la maestro-esclavo asíncrono para *PostgreSQL* desarrollado por Jon Jensen y Greg Sabino Mullane. Bucardo se instala en una sola base de datos, generalmente en el maestro, en la que se indica la ubicación del resto de nodos. En cada nodo se ejecutará un demonio en *Perl*, que se encargará de la replicación entre todos los nodos. Permite el balanceo de carga y data warehousing en los esclavos y permite la escritura en los nodos esclavos. Bucardo funciona entre distintos sistemas operativos, pero el maestro, donde debe estar instalado, tiene que ser una distribución de Linux [26, p. 19].

2.2.1 Replicación sincrónica y asincrónica

En una base de datos replicada, todos los cambios realizados a los datos deben ser aplicados en las múltiples copias, todas las copias tienen que ser idénticas. Tradicionalmente existen dos tipos de técnicas de replicación, sincrónica *(eager)* y asincrónica *(lazy)*. En una replicación sincrónica todos los cambios son enviados a todas las copias. Por otra parte, la replicación asincrónica, maneja la propagación de los datos con un breve retraso desde la realización del cambio. Otra característica que diferencia los tipos de réplicas es quien puede realizar los cambios, puede ser maestro-esclavo cuando los cambios tienen lugar primero en la copia primaria o maestro y después son transmitidos a las segundas copias o esclavos. La estructura maestro-maestro puede realizar cambios de los datos en cualquier copia, dicha copia procesará el cambio y se sincronizará con las restantes [23, p. 16].

Maestro esclavo síncronos

Es una técnica sencilla, ya que las modificaciones solo se pueden realizar sobre la copia primaria y no hay necesidades de coordinación. Este tipo de replicación es útil para entornos con muchas lecturas y pocas modificaciones en los datos y los cambios se propagan inmediatamente, disponiendo de los datos en tiempo real y asegurando la consistencia en caso de fallos [27].

Maestro-maestro síncrono

Es esta técnica mientras el *item* requerido por la transacción no esté bloqueado en todas las copias, no se puede acceder a él. Al inicio de la transacción se envían peticiones de bloqueo a los otros participantes, hasta que el bloqueo no esté garantizado en todas las copias, la transacción continuará. Tras recibir la confirmación de bloqueo, la transacción se ejecuta en todas las copias [27].

Maestro-esclavo asíncrono

Esta técnica evita los costes de la replicación síncrona, gracias a que se proporciona una respuesta al cliente antes de la coordinación entre copias. Las transacciones

de actualización se transmiten de la copia primaria las segundas copias y se ejecutan en las mismas. Normalmente los logs se propagan entre las segundas copias con un retraso después de haber hecho *commit* en la primaria. Esta técnica es útil cuando hay muchas lecturas y pocas escrituras. La propagación de los datos es en diferido, por lo que las modificaciones se toman un tiempo en realizarse desde el *commit* en la copia primaria, también existe la posibilidad que, al realizar una lectura local, esta no incluya las modificaciones más recientes realizadas en copia primaria [26, p. 12].

Maestro-maestro asíncrono

La transacción se ejecuta en el maestro delegado, el cliente obtiene una respuesta transcurrida un tiempo desde el *commit*, las modificaciones se propagan al resto de segundas copias. La propagación de las modificaciones al ser en diferido puede causar inconsistencia de los datos, pudiendo perder modificaciones en caso de caídas. En comparación entre las técnicas maestro-maestro y maestro-esclavo, la replicación multimaestro es una solución más elegante, ya que a diferencia de en la replicación maestro esclavo no necesita distribuir la carga entre las copias, como los datos son replicados en todos los nodos, las modificaciones tienen que ser ejecutadas en todos los nodos por igual. Una manera de mejorar el rendimiento de la replicación multimaestro es procesando las operaciones solo en un sitio y mandando los resultados al resto de los nodos, con esto, la transmisión de los datos entre las copias sería más robusta ante posibles fallos [28].

2.2.2 Proceso de Replicación en PostgreSQL

La replicación es un servicio casi imprescindible para garantizar una alta disponibilidad, rendimiento y fiabilidad. En una base de datos replicada, se guardan múltiples copias de datos en múltiples nodos, pudiendo estar cada uno en distintas maquinas. Con la replicación de los datos, la base de datos puede mantener el servicio si una de las máquinas falla. Podemos encontrar 2 tipos de servidores [26, p. 9]:

- *Maestros*: procesan peticiones de escritura y lectura.
- *Esclavos*: procesan peticiones solo de lectura.

Las replicaciones de una base de datos implican:

Consistencia de los datos: mantiene la integridad y consistencia de los datos es un aspecto de vital importancia, como por ejemplo en las aplicaciones de misión crítica, que requiere una estricta consistencia de los datos modificados por transacciones.

Tiempos de respuesta bajos durante la creación de la réplica: puede ser afectado durante el proceso de creación de la réplica, debido a razones de consistencia.

El mantenimiento: al duplicar los datos, se duplica el espacio ocupado, lo que implica tener que administrar la réplica. Alto tiempo de escritura: las operaciones de escritura en la base de datos pueden ralentizarse durante periodos con gran carga de modificaciones, ya que la transacción necesita transmitirse en múltiples copias [29, p. 14].

2.2.2.1 Proceso de seguridad maestro-esclavo

Inicie sesión en el repositorio principal y cree un usuario con permisos de replicación para transferir datos (también puede usar Superusuario como usuario de replicación directamente, pero no se recomienda). Modifique el *pg_hba.conf* de la biblioteca maestra y agregue las direcciones IP de la biblioteca maestra y la biblioteca en espera a la lista blanca de la política de red de la biblioteca maestra, de modo que la biblioteca en espera pueda conectarse a la biblioteca maestra y, al mismo tiempo, sea conveniente que el maestro y la copia de seguridad cambien. En la configuración de replicación, es mejor escribir las direcciones de *host* maestra y de reserva [30].

Modifique el archivo de configuración *postgresql.conf* de la biblioteca maestra. El parámetro correspondiente al parámetro *synchronous_standby_names* es un nodo de garantía de replicación síncrona. Si este parámetro no está vacío, cualquier

transacción de nivel superior esperará la sincronización con el nodo especificado por este parámetro antes de enviarla a la biblioteca maestra. Si la biblioteca en espera no responde, la biblioteca maestra lo hará. Si este parámetro está vacío, significa replicación asincrónica. Este parámetro se puede configurar con múltiples nodos de garantía, separados por comas, y el PG intentará desde el primero. Habilitar la replicación sincrónica ahorra la seguridad de los datos en la mayor medida, pero afectará la disponibilidad de la aplicación. En modo síncrono, si la red entre *Master* y *Standby* es mala, la replicación síncrona reducirá en gran medida el rendimiento de todo el sistema; si *Standby* deja de funcionar, la base de datos *Master* se bloqueará. Aunque aquí solo se registra la réplica asincrónica, pero qué copia usar depende del escenario de su negocio [31].

El archivo de configuración modificado *postgresql.conf* se puede usar como un archivo de configuración para la biblioteca *Master* y como un archivo de configuración para la biblioteca *Standby*. Este archivo solo se usa cuando se cambian el maestro y la copia de seguridad, y es posible que no se configure, pero la base de datos de la copia de seguridad debe tener este archivo (copie *recovery.conf.sample* de la carpeta compartida en datos y cámbiale el nombre por *recovery.conf*). Ingrese el directorio bin postgresql, abra la Terminal y haga una copia de seguridad de los datos de la base de datos principal en los datos de la base de datos local [32].

D: ruta donde se encuentran los datos, ruta de almacenamiento de datos de la base de datos.

h: dirección de la biblioteca principal remota.

U: qué usuario se utiliza para iniciar sesión.

2.3 Integración de proyectos Arduino en Java con PanamaHitek

Para realizar la integración de proyectos de Arduino en Java se utiliza la librería llamada *PanamaHitek*. Esta librería es una herramienta desarrollada para facilitar la integración de proyectos de Arduino + Java [33].

La librería cuenta con 4 clases [34]:

- *PanamaHitek_Arduino*: esta clase permite manejar todo lo relacionado a la conexión con el Arduino a través del puerto serie. Contiene los métodos para iniciar y detener la conexión, los parámetros de comunicación serie y las funciones para envío y recepción de datos.
- *PanamaHitek_Multimessage*: clase que permite recibir múltiples datos en el Arduino de forma simultánea. Es muy útil a la hora de trabajar con varios sensores a la vez y cuando queremos que el Arduino nos entregue los valores de diferentes al mismo tiempo.
- *PanamaHitek_SpreadSheetHandler*: una clase que estamos desarrollando para almacenamiento de datos y su posterior exportación a ficheros de MS Excel (formato .xls).
- *ArduinoException*: gestor de excepciones de la librería.

Instalación de la Librería

Para instalar la librería se debe descargar el archivo Jar, este se puede encontrar en los repositorios de PanamaHitek (https://github.com/PanamaHitek/PanamaHitek_Arduino/releases/tag/3.0.0).
Cuando se haya descargado el fichero *JAR* podemos agregarlo a un proyecto y empezar a programar. Si usamos Netbeans como IDE de Java, basta con dar clic derecho en la carpeta *"Libraries"* o seleccionar la opción "*Add JAR/Folder"* (Figura 1) y seleccionar el archivo descargado.

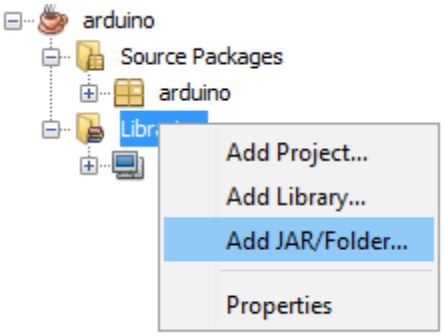


Figura 1 Agregar archivo .Jar en Java

2.3.1 Antecedentes Arduino Uno

Las placas Arduino están dentro de las placas básicas, las cuales permiten aprender a manejar el programa Arduino, está compuesto por un microcontrolador (el cual puede programarse muchas veces) junto con una serie de pines tipo hembra que facilitan la conexión de sensores y actuadores. Existen muchos modelos de placas Arduino, lo característico de todas es que tienen la familia de microcontroladores incorporados, arquitectura de tipo *AVR*. De acuerdo con la necesidad del proyecto se considerará el Arduino ideal ya que muchos difieren en tamaño, número de pines, modelo del microcontrolador, etc [35].

Arduino UNO es una placa basada en el microcontrolador ATmega328P. Tiene 14 pines de entrada/salida digital (de los cuales 6 pueden ser usando con PWM), 6 entradas analógicas, un cristal de 16Mhz, conexión USB, conector jack de alimentación, terminales para conexión ICSP y un botón de reinicio. Tiene toda la electrónica necesaria para que el microcontrolador opere, simplemente hay que conectarlo a la energía por el puerto USB ó con un transformador AC-DC [36].

Tanto las placas como el software han ido evolucionando, al inicio las primeras placas utilizaban un chip FTDI para comunicarse por el puerto USB a la computadora y un procesador para ser programado, luego se utilizó un microcontrolador especial para cumplir esta función como en el caso de Arduino uno que tenían un micro para ser programado y otro para la comunicación, en la actualidad se utiliza un único microcontrolador que lleva a cabo la comunicación y sobre el que también se descargan las instrucciones a ejecutar [35].

2.3.2 Entornos, procesos y paquetes de desarrollo en Arduino + Java

Java es un lenguaje de programación desarrollado por *Sun Microsystems*, se ha convertido en un lenguaje muy popular ya que los programas Java se pueden ejecutar en diversas plataformas con sistemas operativos como Windows, Mac OS, Linux o Solaris. Para conseguir la portabilidad de los programas Java se utiliza un entorno de ejecución para los programas compilados. Este entorno se denomina

Java Runtime Enviroment (JRE), la Figura 2 muestra puede observar de manera gráfica el JRE. Es gratuito y está disponible para los principales sistemas operativos [37].

Figura 2 Figura 2. Entorno JRE *[38]*

Los programas Java son portables, porque pueden ejecutarse en cualquier ordenador o dispositivo móvil, un programa Java puede ejecutarse en un ordenador de mesa, un ordenador portátil, una tableta, un teléfono, un reproductor de música o en cualquier otro dispositivo móvil con cualquier sistema operativo [39].

Entornos de desarrollo

Los entornos de desarrollo de aplicaciones Java se denominan IDE *(Integrated Development Environment)*, son productos que ofrecen al programador un entorno de trabajo integrado para facilitar el proceso completo de desarrollo, desde el diseño, la programación, la documentación y la verificación de los programas. Existen entornos de distribución libre como: *NetBeans*, *Eclipse* o *BlueJ*. Entre los productos comerciales están *JBuilder o JCreatorPro* [37].

Para utilizar un entorno de desarrollo es necesario instalar el *Java Runtime Environment* (JRE) apropiado para el sistema operativo. El JRE se descarga de la página de Oracle Java [39]:

Proceso de desarrollo de software

Es necesario seguir un conjunto de pasos para desarrollar correctamente un producto software. El proceso clásico de desarrollo de software es ampliamente utilizado por su sencillez. Este proceso se compone de las fases: especificaciones, diseño, codificación, prueba y mantenimiento [38]:

Especificaciones: En esta fase se decide la funcionalidad, las características técnicas de una aplicación y sus condiciones de uso.

Diseño: Se utiliza toda la información recogida de la fase de especificaciones y se propone una solución que corresponda a las necesidades del usuario y se pueda desarrollar.

Codificación: Consiste en la programación en Java de las especificaciones del diseño.

Prueba: Se compila y se ejecuta la aplicación para verificar que cumple con los requisitos funcionales y técnicos definido en la fase de especificaciones.

Mantenimiento: En esta fase se corrigen errores de funcionamiento de la aplicación, se modifica la funcionalidad o se añaden las nuevas funcionalidades que requiere el usuario.

Paquetes (packages)

Los paquetes de Java desempeñan un papel equivalente al de las librerías de otros lenguajes. De hecho, las librerías estándar que incorpora el JDK están definidas como *packages* [37].

Los paquetes de java tienen 3 utilidades:

- Organizar y agrupar clases básicamente relacionadas entre sí.
- Evitar conflictos en los nombres de las clases. Dos clases definidas por el usuario pueden tener el mismo nombre siempre y cuando residan en paquetes distintos.
- Permitir el acceso a las variables y métodos definidos en una clase

Físicamente, un paquete será un conjunto de ficheros .class que residen dentro de un mismo directorio o bien un único fichero *.jar* que los contiene a todos, este no es más que un fichero *.zip* al que se le puede añadir una firma digital e información adicional sobre su contenido [39].

2.4 Sistemas Criptográficos Simétricos

El proceso de cifrado permite desarrollar sistemas criptográficos definiendo el tipo en base al tipo de llave a utilizar distinguiendo dos métodos:

- Sistemas de clave única o métodos simétricos en los cuales el proceso de cifrado y descifrado son llevados por una misma clave.
- Sistemas de llave pública o asimétrica cuyo proceso de cifrado y descifrado son llevados a cabo por llaves distintas y complementarias.

Ambos sistemas protegen los datos utilizados en una comunicación y se han incluido en el desarrollo de aplicaciones de comunicaciones como: Servicio de Seguridad: *Secure Socked Layer* (SSL), para el intercambio de registros, Transaccion Electronica Segura *(Secure Electronic Transaction)* por sus siglas en inglés SET para servicios de pagos electrónicos creado por *VISA* y *Master Card* y Servicio de Seguridad (*Private Enhanced Mail)* por siglas en inglés PEM entre otros. Estas aplicaciones hacen uso de criptografía a través de cifrado simétrico, asimétrico, funciones *Hash* y firma digital en parte o bien incluyendo todo en un solo paquete y que en la actualidad han sido de éxito y se mantienen. En particular el cifrado simétrico preserva la confidencialidad tanto en las transmisiones de información como en su almacenamiento, protegiendo los archivos y evitando que personas ajenas a la información tengan acceso [40].

2.4.1 Arquitectura criptográfica de Java (JCA)

La *JCA* es un marco de trabajo para acceder y desarrollar funciones criptográficas en la plataforma Java. Se diseñó alrededor de dos principios básicos [41]:

Independencia e interoperabilidad de las implementaciones:

- La independencia de la implementación se consigue empleando una arquitectura basada en proveedores. El término proveedor de servicios criptográficos se refiere a un paquete o conjunto de paquetes que proporcionan una implementación concreta de los aspectos criptográficos de la *API* de seguridad de Java (como algoritmos de firmado, de funciones de dispersión o conversión de claves.
- La interoperabilidad de las implementaciones significa que cada una de ellas puede trabajar con las demás, usar claves generadas por otra implementación o verificar sus claves.

Independencia y extensibilidad de los algoritmos:

- La independencia de los algoritmos se consigue definiendo tipos de servicios criptográficos y definiendo clases que proporcionan la funcionalidad de estos servicios. Estas clases se denominan clases motor y ejemplos de ellas son las clases *MessageDigest, Signature y KeyFactory*.
- La extensibilidad de los algoritmos significa que nuevos algoritmos que entren dentro de alguno de los tipos soportados (es decir, sean compatibles con las clases motor) puede ser añadido fácilmente.

La separación entre el *JCA* y el *JCE* está motivada por las reglas de exportación de los EEUU para la encriptación. Las clases distribuidas con el *JDK* estándar sólo proporcionan herramientas para el resumen de mensajes y las firmas digitales, lo que permite tener un sistema de autentificación fiable sobre el que implementar un sistema de control de acceso más flexible que el modelo del cajón de arena. El problema es que estas herramientas no son suficientes para el envío seguro de datos, que necesitan algoritmos de encriptación. La solución a este último problema es el *JCE*, que emplea la misma estructura que el *JCA* y proporciona clases motor para implementar criptografía de clave simétrica y para la generación y manipulación de las claves que estos emplean [39].

Las clases Motor

Se utiliza el término motor para referirnos a una representación abstracta de un servicio criptográfico que no tiene una implementación concreta. Un servicio criptográfico siempre está asociado con un algoritmo o tipo de algoritmo y puede tener alguna de las siguientes funciones [37]:

- Proporcionar operaciones criptográficas (como las empleadas en el firmado y el resumen de mensajes).
- Generar o proporcionar el material criptográfico (claves o parámetros) necesario para realizar las operaciones.
- Generar objetos (almacenes de claves o certificados) que agrupen claves criptográficas de modo seguro.

En el *JDK 1.2* el *JCE* define las clases motor, presentadas en la Tabla 1.

Tabla 1 Clases motor JCE *[42]*

Clase JCE 1.2	**Función**
java.crypto.Cipher	Proporciona encriptación y desencriptación.
java.crypto.KeyAgreement	Proporciona un protocolo de intercambio de claves.
java.crypto.KeyGenerator	Proporciona un generador de claves simétricas.
java.crypto.Mac	Proporciona un algoritmo de autentificación de mensajes.
java.crypto.SecretKeyFactory	Representa una factoría de claves secretas.

2.4.2 Características generales de secuencias criptográficas

DES

El algoritmo DES *(Data Encryption Standard)* es un algoritmo de cifrado desarrollado por la NASA a petición del gobierno de EEUU bajo la presión de las empresas por la necesidad de un método para proteger sus comunicaciones. *DES* fue escogido como FIPS *(Federal Information Processing Standard)* en el año 1976 y su uso se extendió por todo el mundo. Hoy en día *DES* es considerado inseguro dada su clave de 56 bits, insuficiente frente al poder computacional actual. La opción que se ha tomado para poder suplantar a *DES* ha sido usar lo que se conoce como cifrado múltiple, es decir aplicar varias veces el mismo algoritmo para fortalecer la longitud de la clave, esto ha tomado la forma de un nuevo sistema de cifrado que se conoce actualmente como *triple-DES* o *TDES* [43]*:*

TDES

El funcionamiento de *TDES* consiste en aplicar tres veces *DES*. Utiliza una clave de 168 bits, aunque se ha podido mostrar que los ataques actualmente pueden romper a *TDES* con una complejidad de 2112, es decir efectuar al menos 2112 operaciones para obtener la clave a fuerza bruta, además de la memoria requerida. Se optó por *TDES* ya que es muy fácil interoperar con *DES* y proporciona seguridad a medio plazo [43].

En los últimos 20 años se han diseñado una gran cantidad de sistemas criptográficos simétricos, entre algunos de ellos están: *RC-5, IDEA, FEAL, LOKI'91, DESX, Blowfish, CAST, GOST,* etc. Sin embargo, no han tenido el alcance de *DES*, a pesar de que algunos de ellos tienen mejores propiedades [42].

AES

El algoritmo AES *(Advanced Encryption Standard)* también conocido como Rijndael fue el ganador del concurso convocado en el año 1997 por el *NIST* con objetivo de escoger un nuevo algoritmo de cifrado. En 2001 fue tomado como *FIPS* y en 2002 se transformó en un estándar efectivo. Desde el año 2006 es el algoritmo más

popular empleado en criptografía simétrica. *AES* opera sobre una matriz de 4x4 *bytes*. Mediante un algoritmo se reordenan los distintos *bytes* de la matriz. El cifrado es de clave simétrica, por lo que la misma clave aplicada en el cifrado se aplica para el descifrado. Basado en El algoritmo *Rijndael*, Al contrario que su predecesor *DES*, Rijndael es una red de sustitución permutación, no una red de Feistel. *AES* tiene 10 rondas para llaves de 128 bits, 12 rondas para llaves de 192 bits y 14 rondas para llaves de 256 bits. En el año 2006, los mejores ataques conocidos fueron las 7 rondas para claves de 128 bits, 8 rondas para llaves de 192 bits, y 9 rondas para claves de 256 bits [40].

IDEA

El IDEA (*International Data Encription Algorithm*) es un algoritmo de cifrado por bloques de 64 bits iterativo. La clave es de 128 bits. La encriptación precisa 8 rotaciones complejas. El algoritmo funciona de la misma forma para encriptar que para desencriptar (excepto en el cálculo de las subclaves). El algoritmo es fácilmente implementable en hardware y software, aunque algunas de las operaciones que realiza no son eficientes en software, por lo que su eficiencia es similar a la del *DES*. Es considerado inmune al criptoanálisis diferencial y no se conocen ataques por criptoanálisis lineal ni debilidades algebraicas. La única debilidad conocida es un conjunto de 251 claves débiles, pero dado que el algoritmo tiene 2^{128} claves posibles no se considera un problema serio. Este algoritmo es mucho más robusto que *DES*, según numerosos expertos criptográficos, *IDEA* es el mejor algoritmo de cifrado de datos existente en la actualidad ya que existen 2^{128} claves privadas que probar mediante el ataque de fuerza bruta [42].

BLOWFISH

Es un algoritmo de cifrado por bloques de 64 bits desarrollado por Scheiner. Es un algoritmo de tipo Feistel y cada rotación consiste en una permutación que depende de la clave y una sustitución que depende de la clave y los datos. Todas las operaciones se basan en o-exclusivas sobre palabras de 32 bits. La clave tiene tamaño variable (con un máximo de 448 bits) y se emplea para generar varios vectores de subclaves. Este algoritmo se diseñó para máquinas de 32 bits y es

considerablemente más rápido que el *DES*. Es considerado seguro, aunque se han descubierto algunas claves débiles, un ataque contra una versión del algoritmo con tres rotaciones y un ataque diferencial contra una variante del algoritmo [44].

A continuación, se presenta un cuadro comparativo (*Tabla 2*) con los aspectos más destacables de cada algoritmo mencionado anteriormente.

Tabla 2 Cuadro comparativo de algoritmos de codificación *[44]*, *[45]*, *[46]*.

Algoritmo	**Clave Efectiva**	**Tamaño De Bloque**	**Claves Posibles**	**Rondas**	**Resistencia**
DES	56 bits	64 bits	2^{56}	16	Insuficiente al poder computacional actual.
TDES	168 bits	64 bits	2^{128}	48	Fuerza bruta Criptoanálisis diferencial.
AES	128 bits, 192 bits, 256 bits	Mínimo 128 Máximo 256	2^{128} 2^{192} 95^{192}	10(128 bits), 12(192 bits),14(256 bits)	Criptoanálisis: Lineal, diferencial, interpolación, plazas de ataques.
IDEA	128 bits	64 bits	2^{128}	8	Ataque de fuerza bruta Criptoanálisis diferencial
BLOWFISH	32 bits hasta 448	16 bits	2^{128}	16	Criptoanálisis

El cifrador aplica al estado cuatro operaciones durante un número determinado de rondas. Dicho número de rondas *(Nr)* viene definido por la longitud de clave utilizada, siendo *Nr = 10* para una longitud de clave de 128 bits, *Nr = 12* para 192 bits y *Nr = 14* para 256 bits. Las cuatro operaciones realizadas en el cifrado son denominadas [47]:

- *SubBytes* — en este paso se realiza una sustitución no lineal donde cada *byte* es reemplazado con otro de acuerdo con una tabla de búsqueda.
- *ShiftRows* — en este paso se realiza una transposición donde cada fila del *«state»* es rotada de manera cíclica un número determinado de veces.
- *MixColumns* — operación de mezclado que opera en las columnas del *«state»*, combinando los cuatro *bytes* en cada columna usando una transformación lineal.
- *AddRoundKey* — cada byte del *«state»* es combinado con la clave *«round»*; cada clave *«round»* se deriva de la clave de cifrado usando una iteración de la clave.

Hay que recordar que *AES* es un cifrado de clave simétrica, lo que quiere decir que la misma clave que se ha usado para cifrar los datos es la que se usará para descifrarlos. Básicamente, el proceso de descifrado consiste en aplicar, en orden inverso al del cifrado, las operaciones inversas a las descritas en el encriptado. Las operaciones que se deben llevar a cabo son [48].

- *InvSubBytes:* la operación *InvSubBytes* es una sustitución no lineal de bytes, es la transformación en el cifrado inverso que es el inverso de *SubBytes ()*
- *InvShiftRows:* en una rotación cíclica hacia la derecha de las filas de la notación matricial del Estado, de manera que la primera fila permanece igual, la segunda fila se rota hacia la derecha una posición, la tercera fila se rota hacia la derecha dos posiciones y, por último, la cuarta fila se rota hacia la derecha tres posiciones.
- *InvMixColumns:* es la operación inversa a *MixColumns*
- *AddRoundKey:* La operación *AddRoundKey* es la misma que en la encriptación, es decir, una operación XOR entre el Estado y la subclave correspondiente.

2.4.3 Esquema de clave en el AES de 128bits

En la Figura 3 se explica cómo se distribuyen las operaciones realizadas en el cifrado a lo largo de las 10 rondas necesarias para una clave de 128 bits.

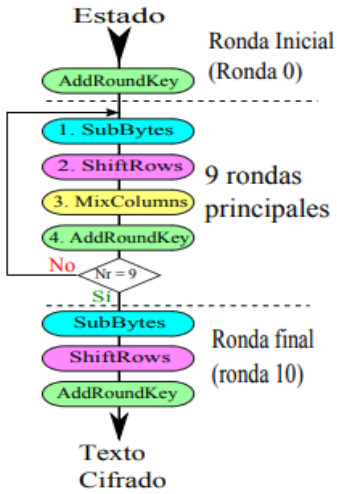


Figura 3 proceso para cifrado de mensaje con clave de 128 bits

En la Figura 4 se explica cómo se distribuyen las operaciones realizadas en el descifrado.

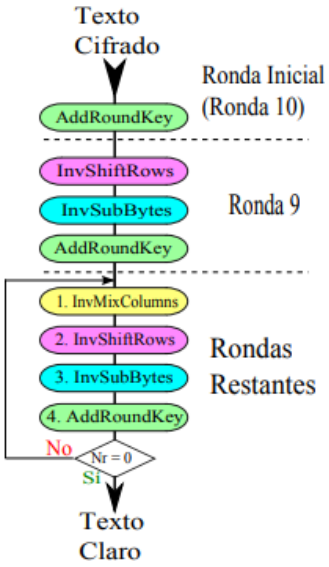


Figura 4 Proceso de descifrado con clave de 128 bits

2.5 Transceptor celular

El SMS funciona sin WIFI y en cualquier dispositivo. Todos los teléfonos móviles tienen la capacidad de enviar y recibir SMS, independientemente de su antigüedad y del sistema operativo que tengan. Actualmente también podemos enviar SMS a través de computadoras o transceptores; un transceptor es un dispositivo que cuenta con un transmisor y un receptor que comparten parte de la circuitería o se encuentran dentro de la misma caja. Cuando el transmisor y el receptor no tienen en común partes del circuito electrónico se conoce como transmisor-receptor.

[49], [50], [51].

2.5.1 Comunicación vía celular con SMS

SMS significa servicio de mensajes cortos, un protocolo utilizado para enviar mensajes cortos a través de redes inalámbricas. A diferencia de muchos servicios actualmente en uso, como MMS y otros servicios basados en datos, SMS todavía funciona en la red de voz fundamental, y se basa en las tres grandes tecnologías de red GSM, CDMA y TDMA, lo que lo convierte en un servicio universal. Los mensajes SMS permiten hasta 160 caracteres de longitud. Cuando creamos mensajes que exceden este límite, se envían como mensajes SMS concatenados [52].

2.5.2 Estándar y parámetros de SMS

El estándar de SMS define qué información se envía en un mensaje de texto, qué bits de código binario componen cada letra, y cómo se organiza esta información para que los dispositivos de envío y recepción puedan comunicarse entre sí. El formato de datos real para el mensaje incluye cosas como la longitud del mensaje, una marca de tiempo, el número de teléfono de destino y el mensaje real. El formato PDU se compone de los siguientes datos en cada mensaje de texto. Los primeros octetos contienen información sobre a dónde enviar el mensaje, qué centro de

mensajes cortos (SMC) y también el número del remitente. La longitud de la información también debe definirse en la cadena, de modo que el receptor sepa exactamente qué buscar [49].

Parámetros de los SMS

Cuando un usuario envía un SMS, o lo recibe, se incluyen con su carga útil *(payload)* al menos los siguientes parámetros [53].

- Marca de tiempo de envío *(timestamp)*.
- Periodo de validez del mensaje, desde una hora hasta una semana.
- Número de teléfono del remitente y del destinatario.
- Número del SMSC que ha originado el mensaje.

De este modo se asegura el correcto procesamiento del mensaje en el SMSC y a lo largo de toda la cadena.

2.5.3 Módulo Sim800L

Módulo para comunicación telefónica que permite añadir voz, texto, datos y SMS a un sistema. Se controla por un microcontrolador de 3 – 5V con UART para el envío y recepción de datos a través de los pines RX/TX, a continuación, se muestran sus principales características [50]:

- Voltaje: 3.4V - 4.4V DC.
- Consumo de corriente (máx.): 500 mA.
- Interfaz: Serial UART.
- Quad-band 850/900/1800/1900MHz – se conectan a cualquier red mundial.
- Hacer y recibir llamadas de voz usando un auricular o un altavoz de 8Ω externo + micrófono.
- Enviar y recibir mensajes SMS.
- Enviar y recibir datos GPRS (TCP/IP, HTTP, etc.).
- Escanear y recibir emisiones de radio FM.
- Soporta A-GPS. ▪ Datos GPRS.
- Velocidad de transmisión 85.6 Kbps.

- Protocolo TCP/IP en chip.
- Velocidades de transmisión serial desde 1200bps hasta 115 200 bps.
- Tamaño de la SIM: Micro SIM.

Capítulo lll
Método de solución del prototipo

El presente proyecto se basa en la metodología de desarrollo de prototipos, donde se muestra lo que se realizó, como son los análisis de requerimientos, diseño de la arquitectura del software, desarrollo del proyecto, así como la evaluación del prototipo y modificaciones.

Se desarrollaron 6 etapas a lo largo del proyecto para alcanzar el objetivo, en la Figura 5 se muestran dichas etapas. Inicialmente se encuentra la etapa de la investigación preliminar en la cual se encuentra la problemática y los requerimientos del cliente, una vez identificado el problema al cual se dará la solución se presenta un diseño básico del prototipo a realizar, continuamos con el diseño detallado del prototipo donde se incluyen todos los diagramas que dan solución al problema y posteriormente se comienza a desarrollar el prototipo. La siguiente etapa es la evaluación, en esta etapa se verifica que el prototipo está realizando la función correctamente y de no ser así, pasamos a la etapa de modificaciones, donde se corrigen los errores presentados en la etapa de evaluación, al modificar el proyecto se realiza una vez más la evaluación y si ya no se presentan errores o mejoras en el prototipo, se llega a la etapa final que muestra el funcionamiento del prototipo final.

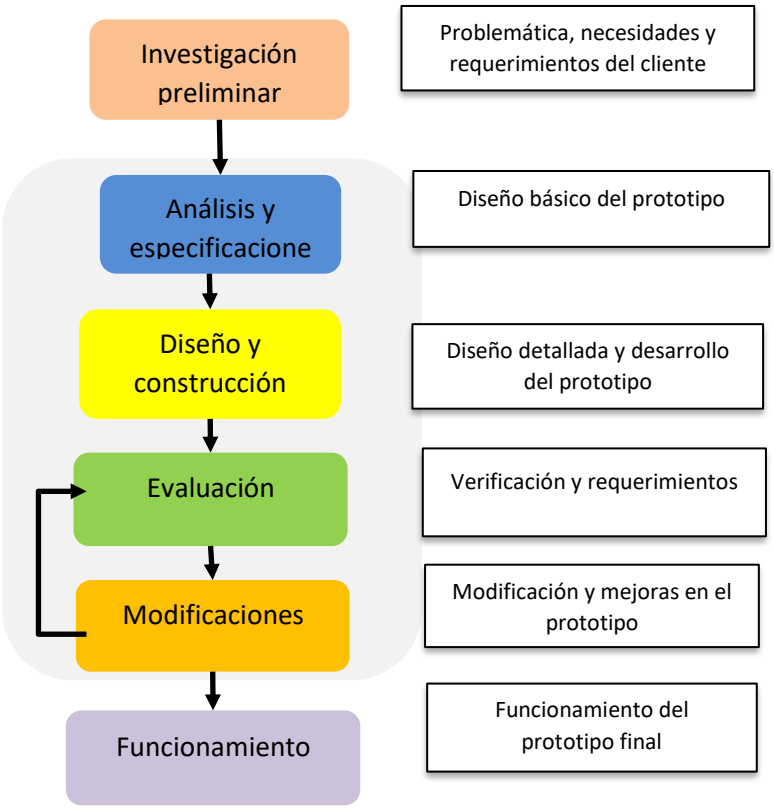


Figura 5 Diagrama de las etapas del proyecto

3.1 Análisis de requerimientos

Se realizó una entrevista a nuestro cliente con el objetivo de recabar la mayor información posible. Esta información es analizada y resumida en una tarjeta de necesidades de usuario Tabla 3.

Tabla 3. Tarjeta de necesidades de usuario TNU (Elaboración propia)

1. DESCRIPCIÓN GENERAL DE LA NECESIDAD
Nombre del Desarrollo: ALGORITMO AES PARA TRANSMISIÓN SEGURA DE DATOS GEOESPACIALES Y SISTEMAS DE REPLICACIÓN ASÍNCRONA. ID:
Descripción:
Se requiere una aplicación que optimice la transmisión de datos geoespaciales de las aeronaves, asegurando que toda información sea transmitida y que esté protegido de posibles ataques. Es necesario que la información pueda ser transmitida por un medio de comunicación de cobertura amplia y que resulte económico para el usuario. Además de sincronizar una base de datos local con un servidor con la implementación de filtros para seleccionar la información a sincronizar, donde se deben de organizar los datos en la base de datos dentro del servidor.
Contexto de la necesidad:
Asegurar la integridad y almacenamiento de los datos contra posibles ataques o mal uso de la información del sistema y sincronizar la información geoespacial de cada una de las misiones que realizan los aviones en un servidor.

DEFINICIONES ESPECÍFICAS DE LAS NECESIDADES		
No.	**Necesidad**	**Definición**
1	**Optimización**	Mejorar el sistema de monitoreo y seguimiento de ruta de las aeronaves.
2	**Tiempo**	Enviar la ubicación del piloto a los controladores en un lapso de tiempo mucho menor.
3	**Economía**	Buscar alternativas que resulten económicas para la empresa.
4	**Transmisión de datos**	Enviar los datos geoespaciales mediante un mensaje de texto.
5	**Transceptor celular**	Hacer uso de un transceptor celular para enviar los datos durante el vuelo y no solamente cuando el piloto esté en tierra.
6	**Seguridad**	Utilizar un algoritmo de codificación para asegurar la información transmitida de cualquier ataque.
7	**Criptografía sólida**	El algoritmo de codificación utilizado no debe ser fácilmente vulnerado.
8	**Clave modificable**	Utilizar una clave de cifrado y descifrado que pueda ser modificada por el usuario.
9	**Recuperación de datos**	El paquete codificado debe recuperar el 100% de la información enviada.
10	**Base de Datos y servidores**	La base de datos, debe tener implementado la herramienta para soportar datos geoespaciales. Este debe de realizar la función de un servidor y otra como una base de datos local (cliente), y deben de estar implementados en distintos equipos. El servidor debe de soportar múltiples conexiones simultáneas para diversos clientes, hasta 100 conexiones.
11	**Sincronización**	La sincronización nos permitirá copiar, distribuir datos y objetos a la base de datos, para después sincronizar entre ambas y así mantener la consistencia e integridad de los datos. La configuración debe incluir la configuración que contiene los paquetes a replicar y además las operaciones que se capturaron.
12	**selección selectiva (filtros)**	Se definirán en las consultas algunas cláusulas para definir las condiciones de búsqueda. Actuará como filtro sobre los resultados devueltos por las cláusulas, cada fila se evaluará contra las condiciones de búsqueda dentro del servidor, para seleccionar los datos que se mandaran a la base de datos local.
13	**Análisis de eventos**	La creación de eventos será utilizada para crear copias de seguridad, eliminar registros, agregar datos en los informes, este objeto se desencadenará por el paso del tiempo, estos eventos serán programados para que se ejecuten una vez en un intervalo de tiempo.
14	**Organización de datos**	Se busca realizar consultas donde podamos organizar los datos, para esto se va utilizar algunas cláusulas para ordenar los datos de varias tablas, esto con el fin de tener ordenados los paquetes, y así poder manejar y encontrar los datos de una forma más fácil y específica.

3.1.1 Diagrama general de solución

En la Figura 6 se muestra el diagrama básico del prototipo que da solución a la problemática expuesta. En la aplicación desarrollada en Java se implementará el algoritmo de cifrado *AES* para encriptar los datos que serán obtenidos de un módulo GPS, estos datos serán procesados y finalmente cifrados para ser transmitidos a través del puerto serial a Arduino. Mediante el puerto TX Arduino envía los datos al módulo *SIM800L* y este finalmente transmite los datos cifrados por SMS que será recibido en un celular, utilizando una app se descifra el texto de los mensajes y se almacenan en una base de datos geoespacial para finalmente ser replicados en otros equipos.

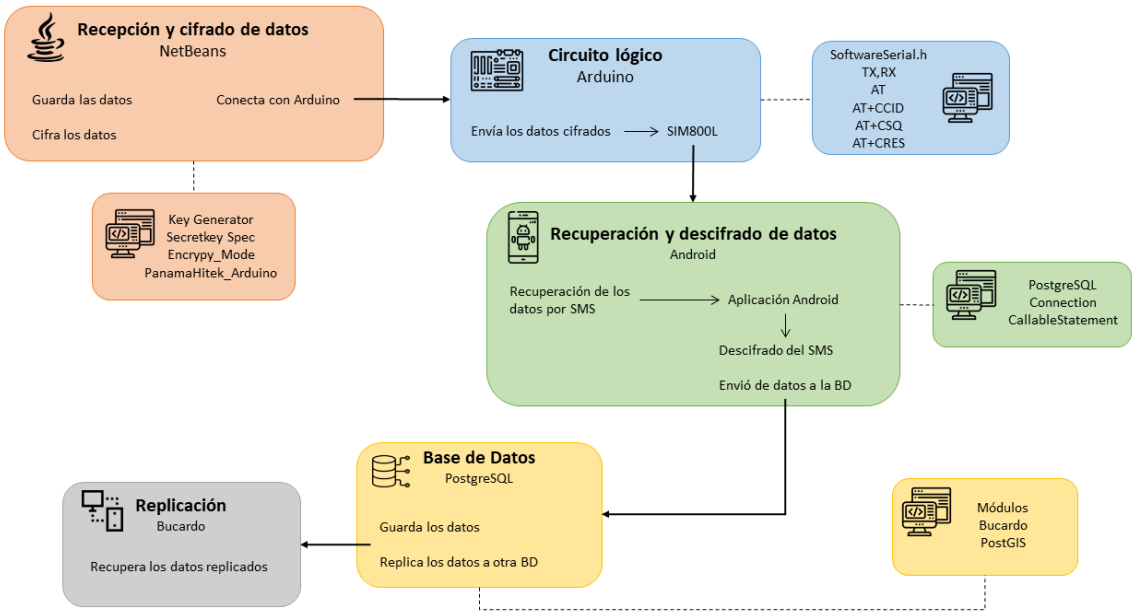


Figura 6 Diagrama básico del prototipo

3.1.2 Diagrama de flujo general

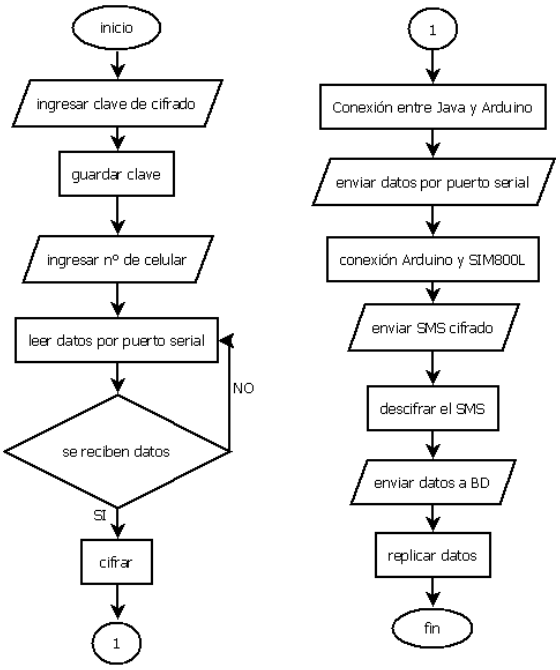


Figura 7 Diagrama de flujo general del sistema

El desarrollo general del proyecto se muestra en la **¡Error! No se encuentra el origen de la referencia.**. Se inicia con una aplicación en Java la cual va a leer la clave de cifrado, una vez ingresada se guarda y después se debe ingresar un número de celular que será utilizado como destinatario, esta misma aplicación se encarga de recibir las coordenadas por el puerto serial y asigna las asigna como mensaje en el método de cifrado que será implementado para la encriptación de los datos. Una vez cifrado el mensaje se instancia un objeto PanamaHitek para realizar la conexión entre Java y Arduino. Los datos enviados de la interfaz en java son recibidos por Arduino y transmitidos al módulo SIM800L que envía los datos cifrados mediante un SMS. Posteriormente serán descifrados con una aplicación móvil y enviados a la base de datos, que almacenará y distribuirá un paquete de datos geoespacial a otros equipos, a través de la replicación asincróna con la herramienta Bucardo.

3.1.3 Diagrama de flujo de Cifrado

Para el desarrollo de cifrado que se requiere en el presente proyecto se utiliza el algoritmo de codificación AES, como primer paso se define el tipo de cifrado. Posteriormente declaramos una variable que guardará la clave de tipo String que será ingresada por el usuario y la cual puede ser modificada. Se asigna el tamaño de la clave ingresada en bytes y después se inicializa la llave, una vez hecho esto se debe codificar la llave en bytes para poder construir la clave, se crea un objeto de cifrado de tipo AES.

Ahora es tiempo de asignar el mensaje que será encriptado en una variable. Entramos en modo de encriptación y se finaliza la transformación del mensaje original, esto nos da como resultado el mensaje cifrado y terminamos con la función. Todo el proceso descrito anteriormente se puede observar en la Figura 8.

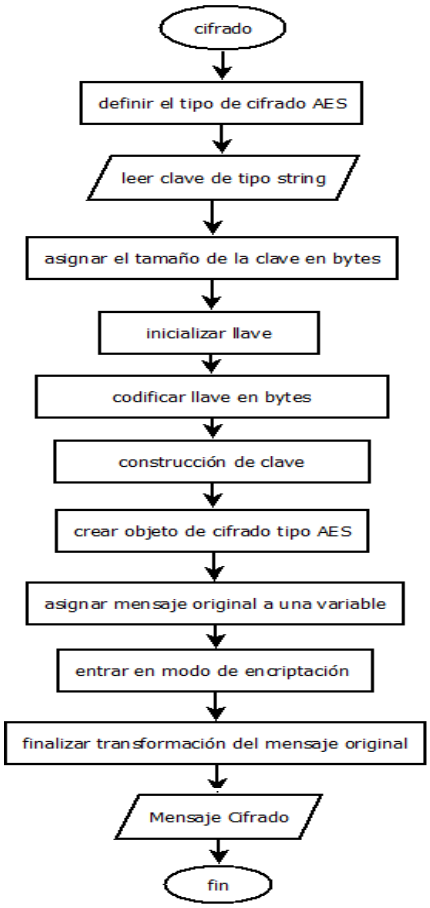


Figura 8 Diagrama de flujo de cifrado (Elaboración propia)

3.1.4 Diagrama de flujo de Descifrado

En la Figura 9Figura 9 se observa el flujo a seguir para la función del descifrado de los mensajes el cual comienza recibiendo la clave de cifrado la que será validada para poder desencriptar los datos, en caso de que la clave no sea correcta se muestra un mensaje de error al usuario y solicita la clave nuevamente.

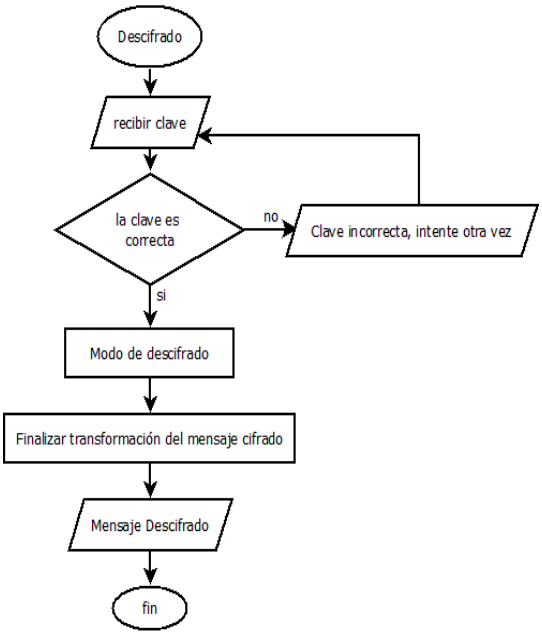


Figura 9 Diagrama de flujo de descifrado (Elaboración propia)

3.1.5 Diagrama de flujo de Aplicación

Como se ha mencionado antes, una de las necesidades del cliente es recuperar el 100% de los datos transmitidos y almacenarlos en una base de datos, así que se desarrolló una aplicación móvil que permite descifrar el texto de los mensajes y enviarlos a la base de datos pulsando un botón. En la Figura 10 se puede observar el diagrama diseñado para el desarrollo de la aplicación.

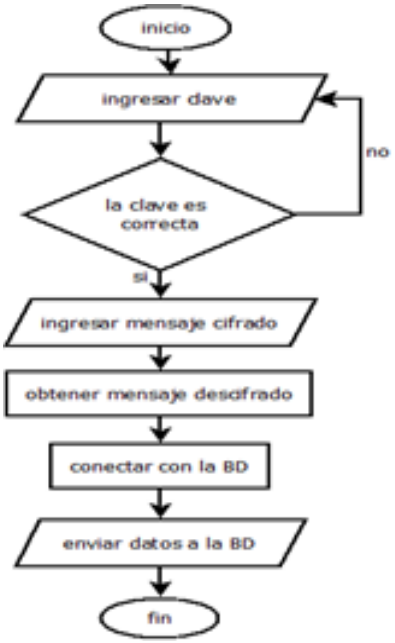


Figura 10 Diagrama de flujo de la aplicación (Elaboración propia)

3.2 Diseño y construcción

3.2.1 Diagrama Entidad Relación de la Base de Datos

En la Figura 11 se creó el diagrama entidad relación para la base de datos, se compone de 5 entidades donde cada uno tiene relación para tener comunicación entre las entidades y cada uno de ellos contiene ciertos atributos, así como su llave primaria y la llave foránea que une a las entidades de otras.

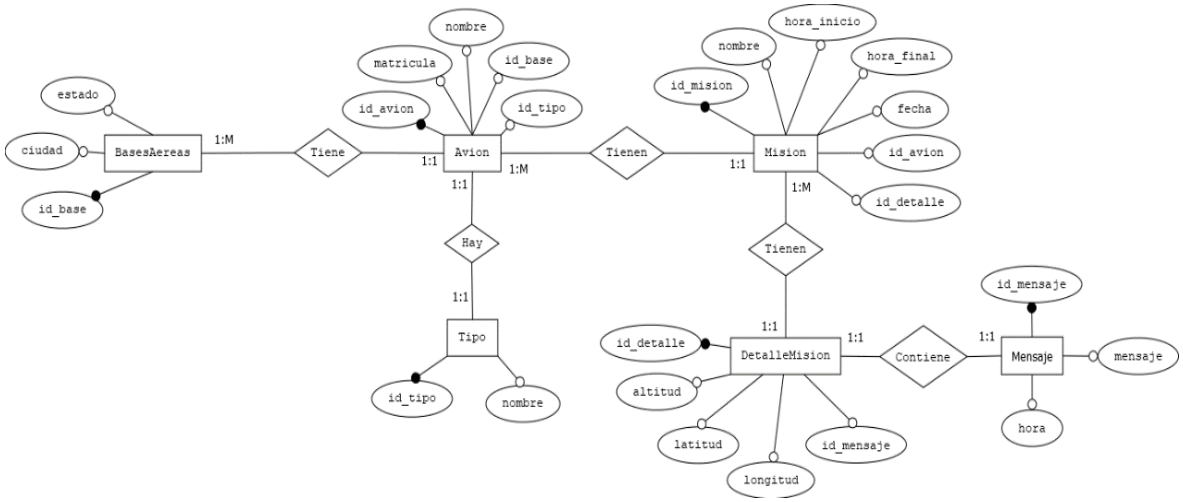


Figura 11 Diagrama entidad relación para la base de datos (Elaboración propia)

3.3.1 Diagrama de bloques de funcionamiento

En el siguiente diagrama (**¡Error! No se encuentra el origen de la referencia.**) se muestran 2 bloques principales; Bloque 1 “Cifrado y transmisión de datos”, que muestra la interacción entre tres sub-bloques en los que se definen las clases y librerías utilizadas para el cifrado de los mensajes, el desarrollo del circuito lógico y la aplicación móvil que descifra y almacena los datos, y el Bloque 2 “Almacenamiento y sincronización de datos” donde se describe la creacion, instalacion y configuracion de la base de datos para realizar una sincronización asíncrona.

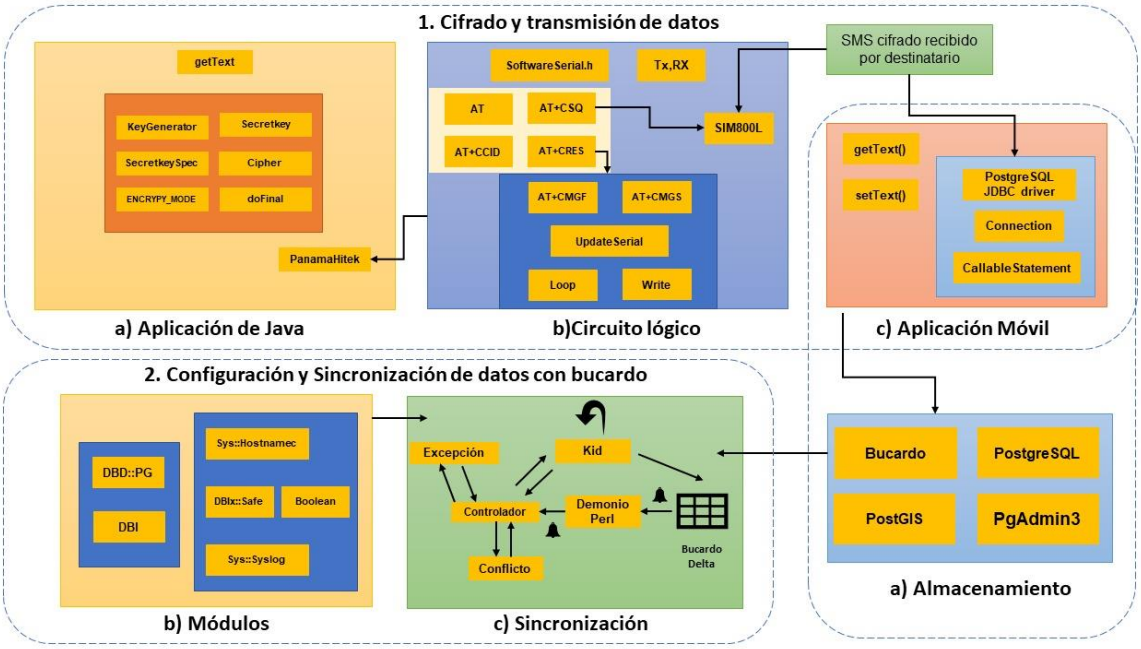


Figura 12 Diagrama de bloques del funcionamiento lógico

1.- Cifrado y transmisión de datos

a) ***Aplicación en Java:*** sirve para recuperar los datos y cifrarlos, una vez cifrados llama a la función para enviarlos vía SMS. En dicho bloque se observa inicialmente al método *getText*, este método sirve para trasladar el contenido que introduce un usuario a través de un control visual a una variable, en este caso se utiliza para ingresar la clave de cifrado y el número telefónico del destinatario. El siguiente código muestra cómo se recupera la clave de cifrado que introduce el usuario y la validación que se aplica al tamaño de la cadena.

```
cc=txtclave.getText();
            if(cc.equals("") || txtNumero.getText().equals("")){
                 JOptionPane.showMessageDialog(null,"Favor de llenar los
campos");
            }
            else {
                if(cc.length()<16){
                JOptionPane.showMessageDialog(null,"la contraseña debe ser
de 16 caracteres exactos,"+
                     " Por favor intente otra vez");
                }
```

```
                else {
                        lblclave.setText(cc);
                        ht.start();
                        arduino();
                }
            }
```

Por otro lado, tenemos las clases utilizadas en el método de cifrado las cuales son: *KeyGenerator*, *SecretKey*, *SecretKeySpec* y *Cipher*, también se muestra la instrucción que sirve para encriptar el mensaje (*ENCRYPT_MODE*) y el método *doFinal* que sirve para transformar el mensaje original.

```
class AES{
    private byte[] valClave;
    public AES(String key){
        valClave= key.getBytes();
    }
public Key generarClave(){
    Key key = new SecretKeySpec(valClave,"AES");
    return key;
}
public String cifrado(String mensaje) throws InvalidKeyException, NoSuchA
lgorithmException, IllegalBlockSizeException, NoSuchPaddingException,
BadPaddingException, UnsupportedEncodingException{

      Key key=generarClave();
      Cipher cipher= Cipher.getInstance("AES");
      cipher.init(Cipher.ENCRYPT_MODE, key);
      byte [] encVal=cipher.doFinal(mensaje.getBytes("UTF-8"));
      return Base64.getEncoder().encodeToString(encVal);

}
```

Para finalizar este bloque tenemos el objeto *PanamaHitek*, esta clase nos ayuda a conectar Java con Arduino enviando los datos a través del puerto serial, a continuación, se muestra la forma de importar la librería `import com.panamahitek.ArduinoException;` para obtener más detalles del código desarrollado diríjase al anexo A.

b) ***Circuito lógico*:** observamos los puertos de conexión TX y RX, a través de ello se conecta al módulo *SIM800L*, también tenemos la librería `#include<SoftwareSerial.h>` que permite la comunicación en serie por los pines del Arduino; el código para incluir la librería es el siguiente:
Tenemos los comandos AT para la conexión entre Arduino y el módulo, a continuación, se definen los comandos y su función [54]:

AT: Es el comando AT más básico. También inicializa el Auto-baud'er. Si funciona, deberías ver el eco de los caracteres AT y luego OK, diciéndote que está bien y que te está entendiendo correctamente.

AT+CSQ: Comprueba la "fuerza de la señal".

AT+CCID: Obtener el número de la tarjeta SIM. Esto prueba que la tarjeta SIM se encuentra bien y puede verificar que el número está escrito en la tarjeta.

AT+CREG: Comprueba que estás registrado en la red.

Encontramos también los comandos *AT* que se utilizan para enviar el mensaje de texto, estos son los siguientes:

AT+CMGF: Es el que fija que formato de mensajes se va a utilizar para los mensajes cortos (SMS).

AT+CMGS: Sirve para asignar el número de teléfono al que será enviado el mensaje.

c) ***Aplicación móvil:*** explica el uso de las funciones *getText()* y *setText()* los cuales sirven para obtener la clave de cifrado y el texto cifrado, y posteriormente mostrar el texto descifrado en pantalla, los métodos se utilizan como se muestra en el siguiente código de ejemplo.

```
AES aes=new AES(txtClave.getText().toString());
        smsDes=aes.descifrado(txtMensajeCifrado.getText().toString());
        txtMensajeDescifrado.setText(smsDes);
```

Para el descifrado de los datos se utilizan las clases mencionadas en la **¡Error! No se encuentra el origen de la referencia.** en el método de cifrado de la aplicación en java.

Una vez descifrado el mensaje se utiliza el controlador *PostgreSQLJDBC* y el objeto *Connection* para realizar la conexión con la base de datos.

```
public class Conexionbd {
        Connection conexion=null;

        public  Connection conexionBD(){
            try{
```

```
                StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolic
y.Builder().permitAll().build();
                StrictMode.setThreadPolicy(policy);
                Class.forName("org.postgresql.Driver");
                conexion = DriverManager.getConnection("jdbc:postgresql://1
92.168.1.81:5432/base1", "postgres", "1234");

            }catch (Exception er){
                System.err.println("Error Conexion"+ er.toString());
            }
            return  conexion;
        }

        //Creamos la funcion para Cerrar la Conexion
        protected  void cerrar_conexion(Connection con)throws  Exception{
            con.close();
        }
    }
```

y por último se muestra el objeto *CallableStatement* utilizado para llamar a los procedimientos almacenados y enviar los datos de la aplicación a la base de datos.

```
CallableStatement cStmt=con.conexionBD().prepareCall(storeProcedureCall);
```

2.- Almacenamiento y sincronización de datos

a) ***Almacenamiento:*** en este apartado se necesitan instalar estos 4 componentes:

- *PostgreSQL* que es nuestro gestor de base de datos.

```
sudo apt-get install postgresql
```

- *PgAdmin3* es nuestra aplicación de diseño para el manejo de PostgreSQL.

```
sudo apt install pgadmin3
```

- *PostGIS* es la herramienta que usamos para manejar datos espaciales.

```
sudo apt install postgis
```

- *Bucardo* es la herramienta que nos permite realizar la sincronización de una base de datos a otra.

```
wget http://bucardo.org/downloads/Bucardo-5.4.1.tar.gz
```

b) ***Módulos para Bucardo*****:** tenemos que instalar los siguientes 6 módulos ya que sin estos módulos no podemos realizar la sincronización entre dos bases de datos o más. A continuación, se define lo que hace cada módulo [55]:

DBI: Se define un conjunto de métodos, variables y convenciones que proporcionan una interfaz de BD.

```
yum -y install perl-DBI
```

DBD::pg: Se trabaja con el módulo *DBI* para proveer acceso a BD.

```
yum install perl-DBD-pg
```

Sys::Hostanme: Se obtiene el nombre de la host.

```
sudo nano pg_hba.conf
```

Sys::Syslog: Se facilita el envío de mensajes al demonio de *syslog*.

```
sudo apt-get install syslog -y
```

DBIx::Safe: Se permite un acceso seguro y controlado para manejar la *DBI*.

```
tar xvfz dbix_safe.tar.gz
```

c) ***Sincronización con Bucardo*****:** es donde se lleva a cabo el proceso de replicación asíncrona siguiendo los pasos listados como se muestra continuación [56]:

1. Se hace un cambio en la tabla y luego se graba en la tabla bucardo_delta
   ```
   alter database bucardo owner to bucardo
   ```
2. Se envía una notificación al demonio principal de Bucardo para hacerle saber que la tabla ha cambiado.
   ```
   perl Makefile.PL
   ```
3. El demonio notifica al controlador para que se sincronice y vuelve a escuchar.
   ```
   sudo bucardo install
   ```

4. El controlador crea un kid para manejar la replicación, o señales de una ya existente.

```
sudo mkdir /var/run/bucardo
```

5. El kid comienza una nueva transacción y deshabilita los disparadores y reglas en las tablas correspondientes

```
sudo chmod 777 /var/run/bucard
```

6. A continuación, se recopila una lista de las filas que han cambiado desde la última replicación, y luego las compara para determinar qué debe hacerse.

```
bucardo add all tables db=db1 --herd=db1_db2
bucardo add all tables db=db2 --herd=db2_db1
```

7. Si se produce un conflicto, entonces el controlador de conflictos estándar, o uno personalizado se ejecuta para resolver el conflicto.

```
bucardo add sync sync_b1ab2 relgroup=db1_db2 dbs=db1,db2
```

```
bucardo add sync sync_b2ab1 relgroup=db2_db1 dbs=db2,db1
```

8. Los disparadores y las reglas se habilitan nuevamente y se confirma la transacción.

```
sudo touch /var/log/bucardo/fullstopbucardo
```

9. Si falla la transacción, entonces se ejecutan los controladores de excepciones personalizados.

```
sudo bucardo start
```

10. El programa kid le indica al controlador de que ha finalizado.

```
sudo bucardo status
```

3.4 Desarrollo del prototipo

3.4.1 Aplicación en Java

Inicialmente se desarrolla la interfaz gráfica de usuario descrita a continuación, en la cual se van a implementar las clases necesarias para cifrar y transmitir los datos,

a continuación, se explica brevemente el desarrollo de las clases principales de la aplicación.

3.4.1.1 Diseño de interfaz gráfica de la aplicación en Java para el cifrado y la transmisión de los datos.

En la Figura 13 se muestra la interfaz gráfica de la aplicación desarrollada en java que sirve para cifrar y enviar los mensajes de texto mediante una conexión con Arduino, para más detalles del código desarrollado véase anexo A.

Como se puede observar tiene un cuadro de texto que sirve para ingresar la clave de cifrado y otro donde se ingresará el número telefónico del destinatario. Además, cuenta con un botón para iniciar el proceso, un botón para detener el proceso y otro más para limpiar todos los campos. Por otro lado, se encuentra un reloj que se encarga de mostrar el tiempo transcurrido desde que se inició el proceso hasta que finaliza. Para más detalles sobre el funcionamiento de la interfaz gráfica véase Anexo B.

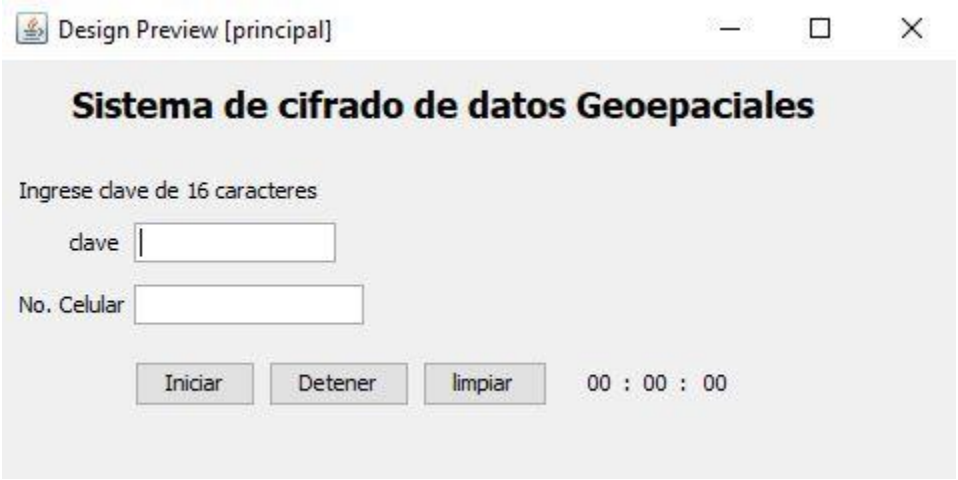


Figura 13 Interfaz gráfica del usuario

3.4.1.2 Código de cifrado

Dentro del método *serialEvent* que se encarga de escuchar el puerto serial y obtener las coordenadas se implementa la función *cifrado()* para procesar los datos

obtenidos y obtener los mensajes encriptados. Véase anexo A para más detalles de la función *cifrado()*.

```
class AES{
    private byte[] valClave;
    public AES(String key){
        valClave= key.getBytes();
    }
public Key generarClave(){
    Key key = new SecretKeySpec(valClave,"AES");
    return key;
}
public String cifrado(String mensaje) throws InvalidKeyException, NoSuchAlgorithm
Exception, IllegalBlockSizeException, NoSuchPaddingException,
BadPaddingException, UnsupportedEncodingException{

      Key key=generarClave();
      Cipher cipher= Cipher.getInstance("AES");
      cipher.init(Cipher.ENCRYPT_MODE, key);
      byte [] encVal=cipher.doFinal(mensaje.getBytes("UTF-8"));
      return Base64.getEncoder().encodeToString(encVal);
     }
```

3.4.2 Circuito lógico en Arduino Uno

Para la elaboración del circuito se utilizó:

- Una placa con microcontrolador Arduino Uno
- Protoboard
- Un módulo SIM800L
- Un SIM movistar
- Cables Jumper macho-macho
- Cable USB
- Antena para modulo GMS
- Batería Lipo de 3.7V a 3700 mAh

Se siguió el diagrama mostrado en Figura 14 Circuito para envió de SMS con modulo SIM800 (Elaboración propia)Figura 14.

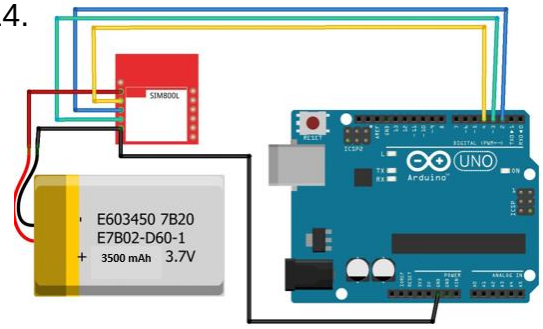


Figura 14 Circuito para envió de SMS con modulo SIM800 (Elaboración propia)

Para comenzar se soldaron los pines del módulo y la antena que incluye. El puerto 2 y 3 del Arduino se conectan a los puertos RX y TX del módulo SIM800L, se conectó la antena al módulo y la fuente de alimentación externa como se muestra en el diagrama.

3.4.3 Código de conexión de Arduino con Java

Los datos recuperados en la interfaz gráfica de Java deben ser transmitidos a través del puerto serial a la placa Arduino para que sean enviados por el módulo *SIM800L*. Para esto se carga la librería *panamaHitek* al proyecto de Java y también la clase *PanamaHitek_Arduino* que sirve para realizar la conexión con Arduino y posteriormente se debe crear un oyente como se muestra a continuación, que se ejecutará cada que se reciba un dato por el puerto serial.

```
import com.panamahitek.ArduinoException;
import com.panamahitek.PanamaHitek_Arduino;

      PanamaHitek_Arduino ino =  new  PanamaHitek_Arduino ();
      PanamaHitek_Arduino inoSend =  new  PanamaHitek_Arduino ();
      SerialPortEventListener listener =  new SerialPortEventListener () {

        public  void serialEvent ( SerialPortEvent  spe ) {
            try {
                if (ino.isMessageAvailable ()) {

                       if(i<=100){
                        lista.add(ino.printMessage());

                        AES aes1=new AES(cc);
                        String smsCifrado=aes1.cifrado(lista.get(i).toString());
                        inoSend.sendData(txtNumero.getText()+"\n");
                        inoSend.sendData(smsCifrado+"\n");

                        System.out.println(lista.get(i).toString());

                        i++;
                    }
                    else{
                        lista = new ArrayList();
                        i = 0;
                }
              }


            } catch (SerialPortException ex) {
                Logger.getLogger(principal.class.getName()).log(Level.SEVERE,
null, ex); }
```

Una vez hecho esto podemos llamar al método *arduinoRX()* para hacer la conexión con el trasmisor y java que se encuentra en el puerto *"COM3"*, también tenemos el método *arduinoRXTX()* que hace la conexión con Arduino desde Java y permite enviar y recibir datos, este método recibe como parámetro el puerto al que está conectado el Arduino, la velocidad a la que está trabajando y el oyente declarado anteriormente. Para ver el código completo diríjase al anexo A.

```
try {
      ino.arduinoRX("COM3", 115200, listener);
      inoSend.arduinoRXTX ( "COM4" , 9600 , listener);
    } catch (ArduinoException ex) {
      Logger.getLogger(principal.class.getName()).log(Level.SEVERE ,null, ex);
}
```

3.4.4 Aplicación móvil

A continuación, se describen de manera general los aspectos destacables de la aplicación desarrollada en Android Studio que sirve para descifrar y guardar la información en la base de datos. Para más detalles del código de la aplicación véase el Anexo A.

3.4.4.1 Interfaz

La interfaz gráfica (Figura 15) de la aplicación cuenta únicamente con tres cuadros de texto, el primero se utiliza para que el usuario pueda ingresar la clave de descifrado, el segundo sirve para que el usuario ingrese el texto cifrado que se desea descifrar y el tercero muestra el texto descifrado. También cuenta con tres botones, el primero sirve para descifrar el texto, el segundo para guardar los datos en la base de datos y por último un botón para limpiar los cuadros de texto.

Figura 15 Interfaz gráfica de aplicación móvil

3.4.4.2 Código de descifrado

Para poder descifrar los mensajes se lleva a cabo un proceso muy similar al del cifrado solo que esta vez el proceso es a la inversa. Es necesario que el usuario ingrese la misma clave con la que el mensaje fue cifrado y también el texto cifrado, la clave y el texto son procesados hasta que finalmente se obtiene el contenido del mensaje original. Para obtener más detalles del código, diríjase al Anexo A.

```
public String descifrado(String mensajeEnc) throws NoSuchAlgorithmException,
NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException,
BadPaddingException, IOException {

        Key key=generarClave();
        Cipher cipher= Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte[] valordec = cipher.doFinal(Base64.decode(mensajeEnc,DEFAULT));
        return new String(valordec);

    }
```

3.4.4.3 Conexión BD

Para lograr la conexión de nuestra aplicación con la base de datos se cargó el controlador *PosgreSQL JDBC* y posteriormente se creó la clase que realiza la conexión remota, en esta clase es necesario agregar información de la base de datos como el *hostname, el puerto, el nombre de la base de datos, el usuario y la contraseña.* Finalmente se llaman los procedimientos almacenados que se encargan de insertar los datos en la base. Para obtener más detalles, diríjase al Anexo A.

3.4.5 Configuración del SGBD

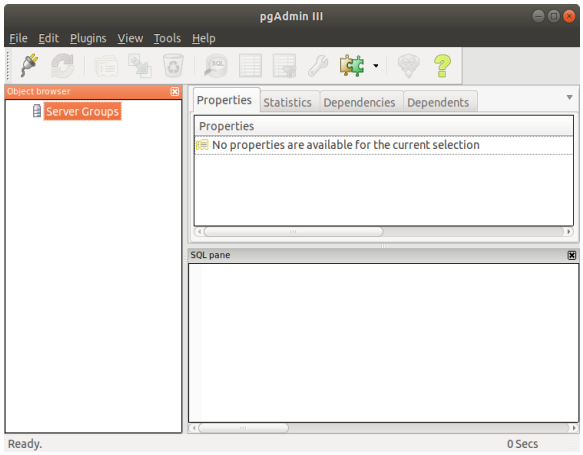


Figura 16 Interfaz pgAdmin3

En la Figura 16 se desarrolló el servidor en la interfaz de pgAdmin3, que es una herramienta para la administración de la base de datos *PostgreSQL*, donde se realizaron las consultas para el desarrollo de nuestra base de datos.

Al crear nuestro servidor con la interfaz, tendremos que colocar *un nombre, el host en este caso colocaremos la dirección IP de la máquina* con la que se quiera conectar, colocamos el puerto en este caso se utilizó el puerto 5432, también se colocó el nombre de la *base, usuario y la contraseña para tener conexión*. Para ver el desarrollo del servidor véase al Anexo A.

3.4.6 Esquema bucardo y extensión POSTGIS

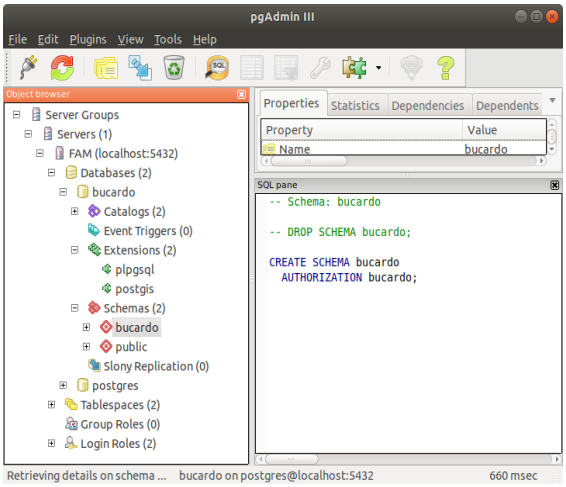


Figura 17 Comprobación de la instalación de bucardo y postgis

En la Figura 17 tenemos la extensión postgis que nos sirve para el uso de datos geoespaciales. También tenemos instalado el esquema bucardo, que nos permite realizar la sincronización entre varias bases de datos. La extensión y el esquema se instaló desde la consola de Ubuntu. Para obtener más detalles, diríjase al Anexo A.

3.4.7 Creación de la Base de Datos

Se crearon dos bases de datos en el mismo servidor como se muestra en la Figura 18 en este caso la primera base de datos es “base1” y la segunda base de datos es “base2”. Debemos tener en cuenta que se debe de tener instalado la extensión postgis y el esquema bucardo de la Figura 17 Comprobación de la instalación de bucardo y postgis.

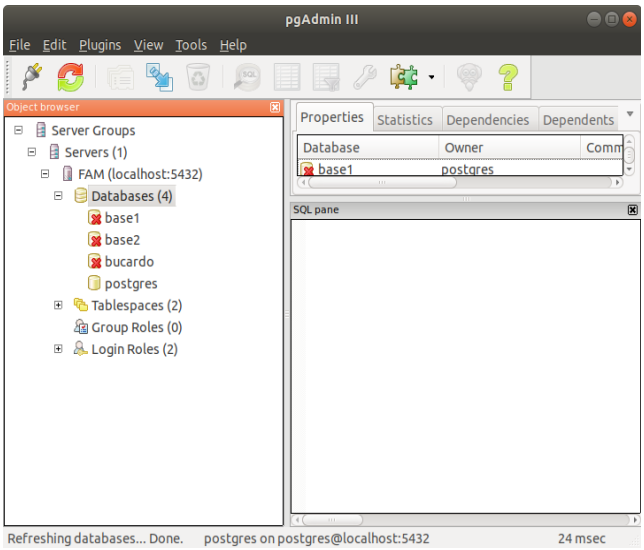


Figura 18 Creación de las dos bases de datos en el servidor

3.4.7.1 Creación de las Tablas

En cada una de las bases de datos creadas, como se muestra en la Figura 19. Creación de las Tablas, se crearon 6 tablas (avión, bases aéreas, detalle misión, misión y tipo avión) cada uno tiene diferentes campos, así como sus llaves primarias y llaves foráneas para poder tener relación y usar sus atributos. Para obtener más detalles del código, diríjase al Anexo A.

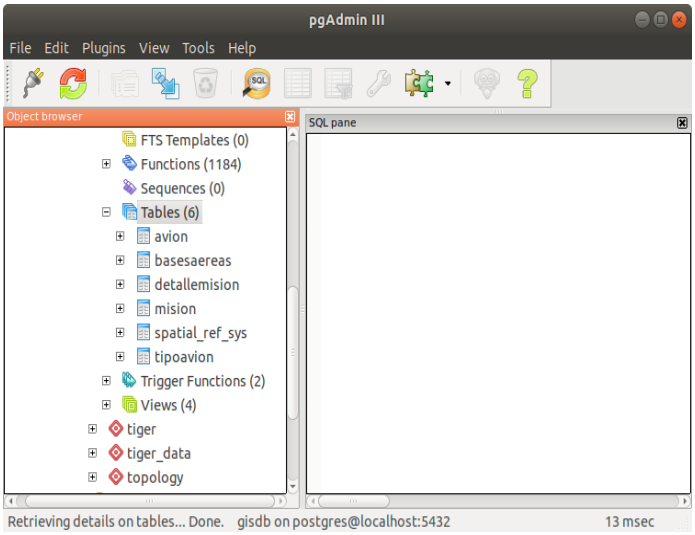


Figura 19 Creación de las tablas en la “base1”

3.4.7.2 Procedimientos almacenados

Se crearon dos procedimientos almacenados en la base de datos de *PostgreSQL*, esto con el fin de poder hacer la conexión con la aplicación de Android, donde este mandará un paquete de datos geoespaciales y un texto, todo esto será guardado

en los procedimientos almacenados. Las tablas que se usaron para los procedimientos son: *Detalle Misión* y *Mensaje*. Para obtener más detalles del código, diríjase al Anexo A.

3.4.8 Sincronización

Para realizar la sincronización, se deben de añadir las bases de datos y sus respectivas tablas, luego debemos agruparlas, una vez realizado esto debemos añadirlos a la sincronización, todo esto se realiza desde la terminal de Ubuntu en la carpeta de bucardo.

3.4.8.1 Añadir las bases de datos

Ingresamos la siguiente línea de comando, como se muestra en la Figura 20, donde agregamos la base1 con el nombre db1 y hacemos lo mismo con la base2 con el nombre db2.

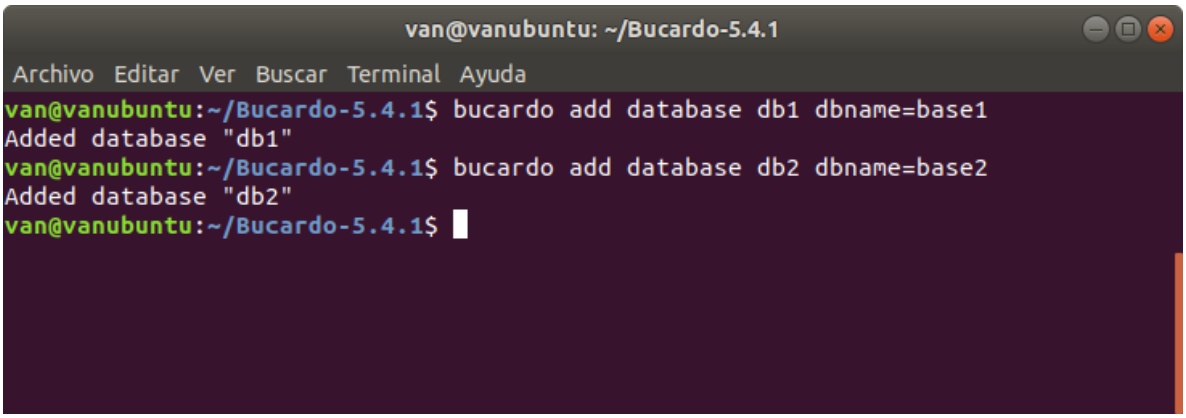


Figura 20 Se agregaron las bases de datos a la carpeta bucardo

3.4.8.2 Añadir tablas

Ingresamos el siguiente comando, como se muestra en la figura 2. En esta línea agregamos todas las tablas que contiene la base de datos db1 y db2 a bucardo, hay que recordar que son los nombres que se le asignaron en la Figura 21.

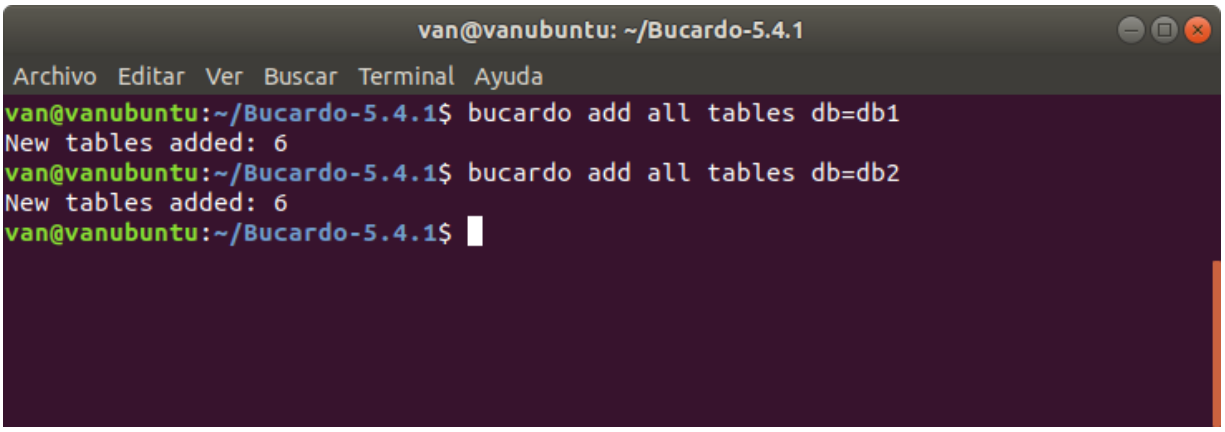


Figura 21 Se agregaron las tablas a la base de datos

3.4.8.3 Añadimos grupos

En la Figura 22, se agregaron todas las tablas de la base de datos db1 a un grupo que lo nombramos db1_db2, donde podemos ver que todas las tablas se agregar al grupo creado.

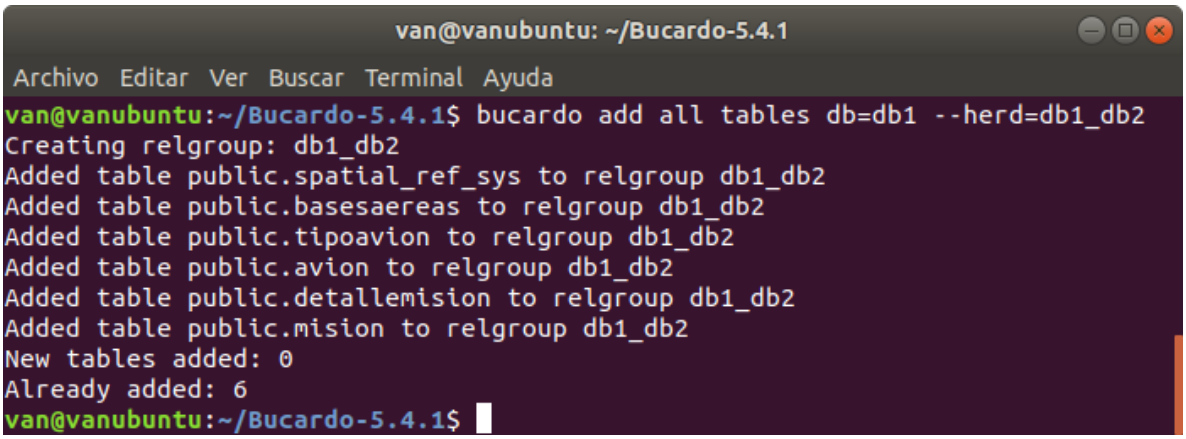


Figura 22 Se agregaron las tablas del la base1 a un grupo

En la Figura 23, se agregaron todas las tablas de la base de datos db2 a un grupo que lo nombramos

db2_db1, donde podemos ver que todas las tablas se agregar al grupo creado.

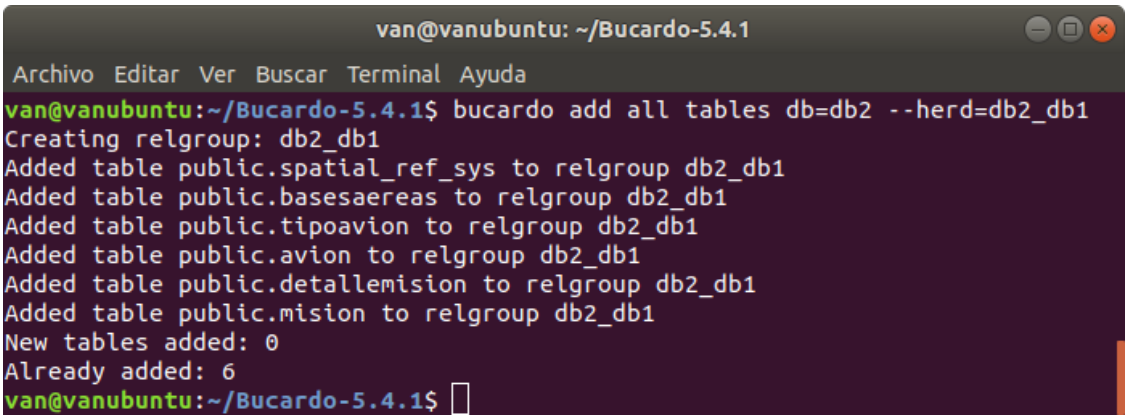


Figura 23 Se agregaron las tablas de la base2 a un grupo

3.4.8.4 Añadimos Sincronización

En la Figura 24 agregamos el grupo db1_db2 a la sincronización, mencionando el nombre que se le dio a las bases de datos como se muestra en la figura 20 y se crea el grupo sync_b1ab2, hacemos lo mismo con el grupo db2_db1 solo que invertimos las bases de datos y quedaría db2 a db1 y se crea el grupo sync_b2ab1.

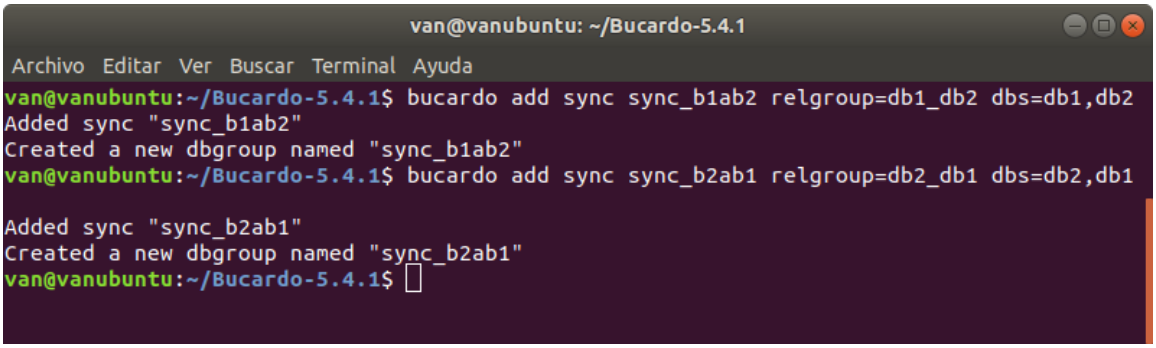


Figura 24 Se agregaron los grupos a la sincronizan

3.4.8.5 Activación de Bucardo para la replicación

En la Figura 25**¡Error! No se encuentra el origen de la referencia.** se muestra la última parte de la sincronización donde colocamos el comando *bucardo start* para iniciar las funciones de bucardo. En caso de que te mande este error, solo se deberá crear el directorio y luego iniciar otra vez el bucardo y te mandara el mensaje de que bucardo se inició correctamente.

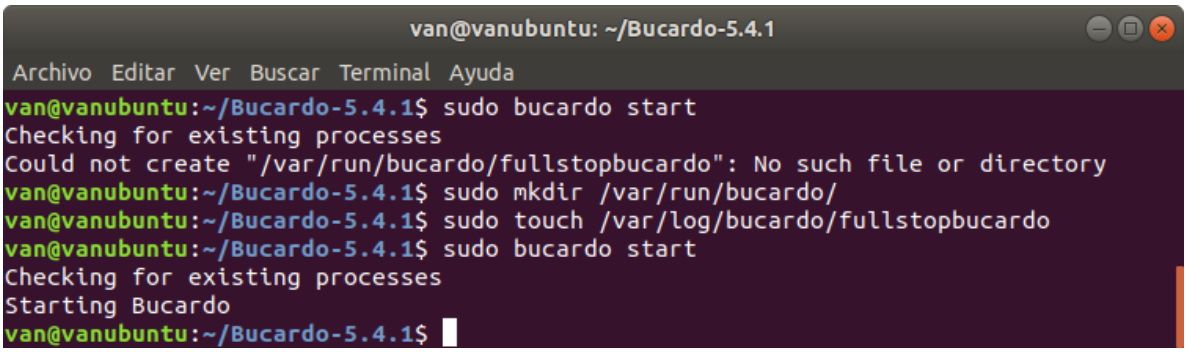


Figura 25 Inicializando bucardo para la sincronización

nota

En el siguiente capitulo abarcaremos los resultados del proyecto, donde se mostrarán las evaluaciones, modificaciones y las pruebas que se usaron con Arduino con la comunicación con el módulo SIM800L, así como él envió del SMS a la aplicación y el descifrado de las coordenadas que serán enviadas a la base de datos y este realizara la sincronización de datos a otra base de datos.

Capítulo IV
Resultados obtenidos con las pruebas del prototipo

En este capítulo se aborda la evaluación, modificaciones y el funcionamiento del prototipo final, detallando cada una de las etapas para finalmente presentar los resultados obtenidos de las pruebas realizadas, donde se utilizó un formato JSON con datos geoespaciales que simulen las coordenadas de la aeronave.

4.1 Evaluación y modificaciones del prototipo

Al llevar a cabo la primera evaluación del prototipo se pudo notar que el módulo SIM800L no se comunicaba con Arduino. Este fallo se debe a que inicialmente el módulo se estaba alimentando directamente del Arduino, es por lo que se utilizó una fuente de alimentación externa, en este caso una batería Lipo de 3.7V a 3500mAh.

En la segunda evaluación del prototipo se logra la comunicación entre Arduino y el módulo, sin embargo, el módulo no logra conectarse a la red, esto se debe a dos posibles fallos, la tarjeta SIM no tiene buena cobertura o la antena no tiene buen alcance.

Para resolver estas fallas, se sustituyó la antena que incluye el módulo por una antena para modulo GMS y se realizaron pruebas con más de 3 tarjetas SIM de diferentes compañías. La tarjeta SIM de la compañía Telcel alcanzó una intensidad de señal de 22 dBm, sin embargo, no logró conectarse a la red, la tarjeta AT&T no fue reconocida por el módulo por lo tanto no logró conectarse a la red. Se logró la conexión del módulo a la red con una tarjeta SIM Movistar, esta tarjeta alcanzo una intensidad de 23 dBm por lo tanto logro registrarse en la red. También, se configuró la banda de frecuencia con el comando AT+CBAND=”PCS_GMS”, este comando se utiliza para configurar la banda de operación móvil con la cual podremos enviar información a través de la red**.** Al ejecutar este comando indicando que la banda es igual a “PCS_GMS”, establecemos que el módulo hará uso de una banda de frecuencia del espectro radioeléctrico que es utilizada para el servicio móvil en México [57].

4.2 Pruebas de funcionamiento

Para realizar las pruebas se utilizó un transmiso que recupera datos geoespaciales (Figura 26). Los datos son encriptados y mostrados en la consola de NetBeans al mismo tiempo que son enviados vía SMS. La aplicación en Java funciona correctamente, recibe los datos conforme los envía el transmisor y encripta los datos,

se muestran en la consola cada 10 segundos aproximadamente, además, el destinatario recibe los datos cifrados como mensaje de texto.

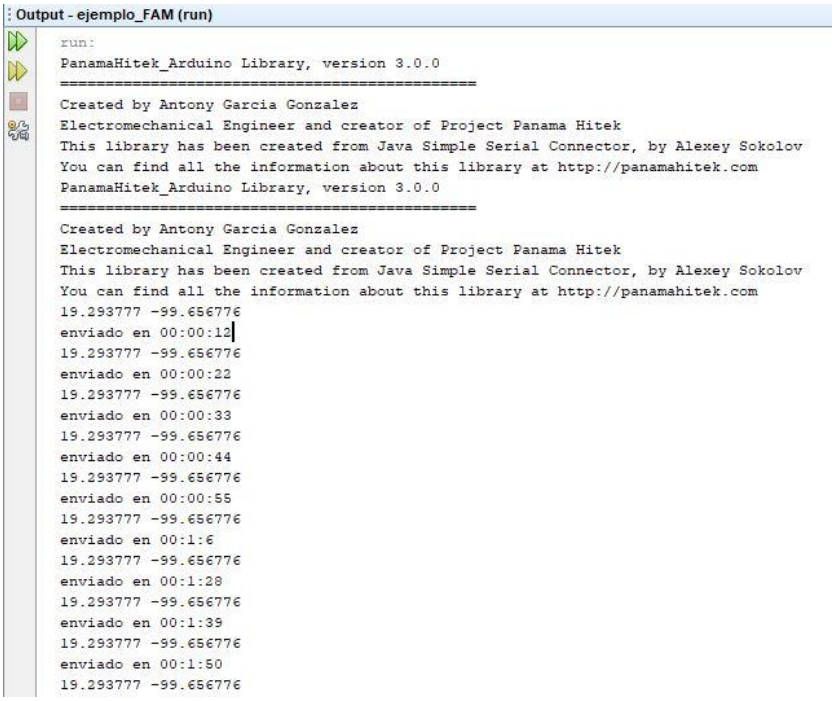


Figura 26 Coordenadas recibidas en NetBeans

Es necesario saber si la información transmitida se recupera al 100 por ciento, para esto, se desarrolló una aplicación que descifra el contenido de los mensajes y lo guarda en una base de datos.

Para poder descifrar el contenido del mensaje se ingresó la misma clave de cifrado utilizada al enviar los mensajes, como ya se ha mencionado, AES es un algoritmo de cifrado simétrico, por lo tanto, la clave con la que se encriptan los datos debe ser utilizada para desencriptarlos.

Una vez ingresada la clase, se copia el contenido de un mensaje recibido y se ingresa en la aplicación como se muestra en la Figura 27, como se puede observar se desencripta el primer mensaje que se recibió.

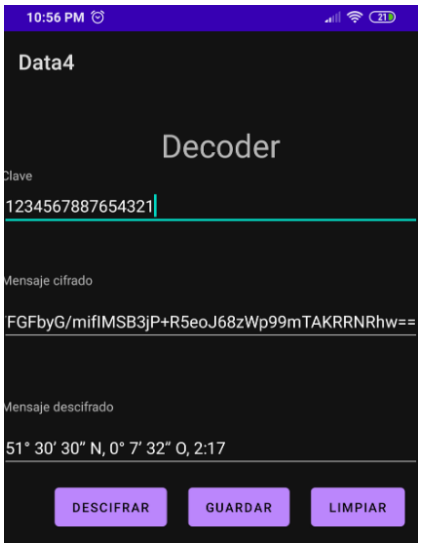
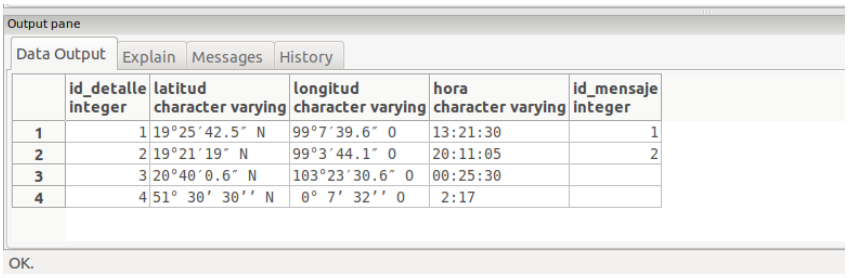


Figura 27 Prueba de funcionamiento de descifrado

Una vez que se ingresaron los datos en la aplicación pulsamos el botón “GUARDAR”, los datos descifrados se envían a la base de datos y se almacenan en la tabla “detallemision” en sus campos correspondientes, en la Figura 28 se muestran los datos almacenados correctamente.



	id_detalle integer	latitud character varying	longitud character varying	hora character varying	id_mensaje integer
1	1	19°25′42.5″ N	99°7′39.6″ O	13:21:30	1
2	2	19°21′19″ N	99°3′44.1″ O	20:11:05	2
3	3	20°40′0.6″ N	103°23′30.6″ O	00:25:30	
4	4	51° 30′ 30′′ N	0° 7′ 32′′ O	2:17	

Figura 28 Almacenamiento de los datos enviados por la aplicación de tipo coordenadas

Al tener los datos almacenados en la tabla “*detallemision*” de la base1, podemos observar que en la tabla “*detallemision*” de la base2 se sincroniza con la base1 y obtendrá los mismos registros almacenados que tiene, esto es gracias a que la sincronización con bucardo esta activa y todos los datos que se almacenen a la base de datos, se sincronizan con la otra base de datos de manera automática. Como se muestra en la Figura 29.

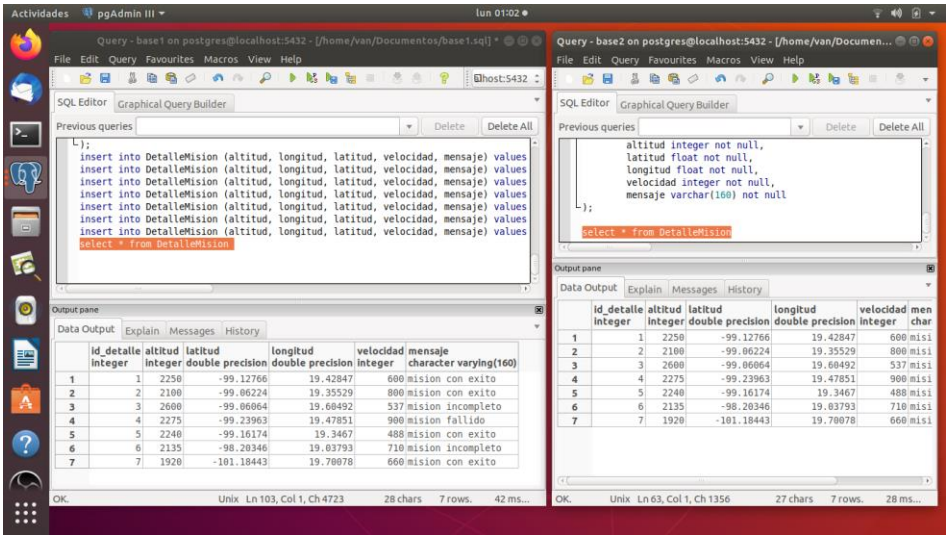


Figura 29 Sincronización de la tabla detalle misión.

nota

4.3 Resultados de las pruebas

En la

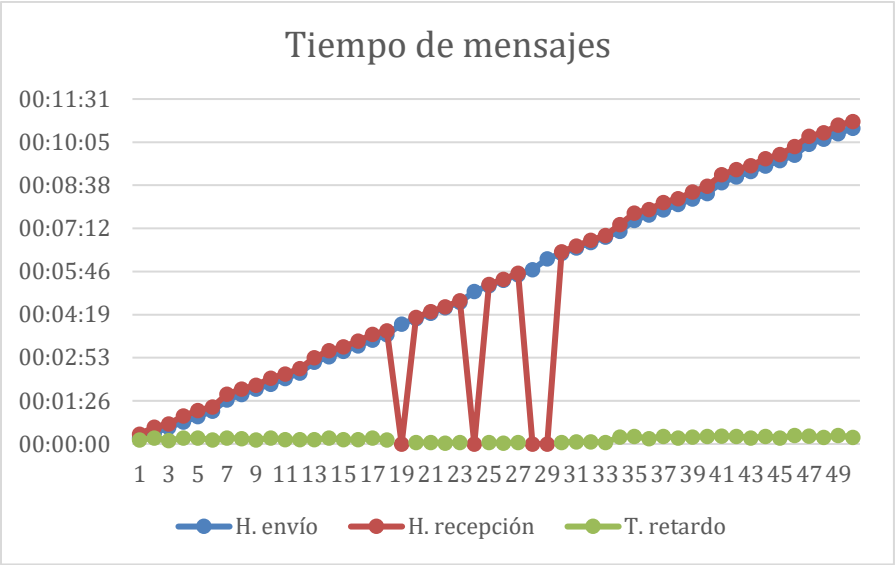
Tabla *4* se muestran los tiempos de envío, recepción y retardo entre cada mensaje. Se envía en promedio un mensaje cada 10 segundo, con variantes de hasta 2 segundos de retardo en el envío, el mayor tiempo de retardo entre el envió y la recepción es de 17 segundos y el menor fue de 2 segundos. Las pruebas se realizaron con una muestra de 53 mensajes donde 49 mensajes fueron recibidos correctamente y 4 mensajes se perdieron, esto nos da un 92.45% de mensajes enviados con exito y un 7.55% de mensaje perdidos.

H. envío	H. recepción	T. retardo
00:00:12	00:00:20	00:00:08
00:00:22	00:00:34	00:00:12
00:00:33	00:00:40	00:00:07
00:00:44	00:00:56	00:00:12
00:00:55	00:01:07	00:00:12
00:01:06	00:01:14	00:00:08
00:01:28	00:01:40	00:00:12
00:01:39	00:01:50	00:00:11
00:01:50	00:01:58	00:00:08
00:02:00	00:02:12	00:00:12
00:02:11	00:02:20	00:00:09
00:02:22	00:02:31	00:00:09

00:02:44	00:02:53	00:00:09
00:02:55	00:03:07	00:00:12
00:03:06	00:03:15	00:00:09
00:03:17	00:03:26	00:00:09
00:03:28	00:03:40	00:00:12
00:03:39	00:03:47	00:00:08
00:04:00	X	
00:04:11	00:04:14	00:00:03
00:04:22	00:04:25	00:00:03
00:04:33	00:04:35	00:00:02
00:04:44	00:04:47	00:00:03
00:05:06	X	
00:05:17	00:05:20	00:00:03
00:05:28	00:05:30	00:00:02
00:05:39	00:05:42	00:00:03
00:05:49	X	
00:06:11	X	
00:06:22	00:06:25	00:00:03
00:06:33	00:06:37	00:00:04
00:06:44	00:06:48	00:00:04
00:06:55	00:06:58	00:00:03
00:07:06	00:07:20	00:00:14
00:07:28	00:07:43	00:00:15
00:07:39	00:07:50	00:00:11
00:07:49	00:08:04	00:00:15
00:08:00	00:08:12	00:00:12
00:08:11	00:08:25	00:00:14
00:08:22	00:08:37	00:00:15
00:08:44	00:09:00	00:00:16
00:08:55	00:09:10	00:00:15
00:09:06	00:09:18	00:00:12
00:09:17	00:09:32	00:00:15
00:09:28	00:09:40	00:00:12
00:09:39	00:09:56	00:00:17
00:10:01	00:10:17	00:00:16
00:10:11	00:10:24	00:00:13
00:10:22	00:10:39	00:00:17
00:10:33	00:10:46	00:00:13

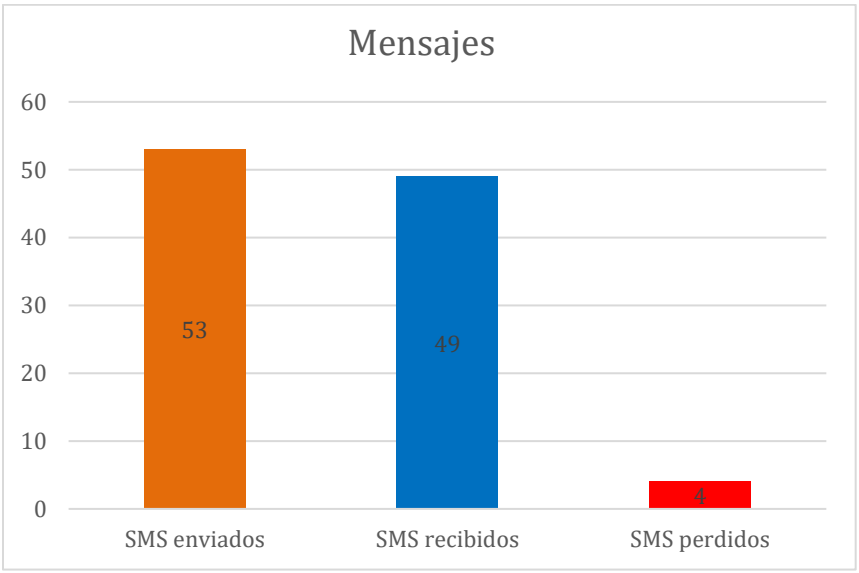
Tabla 4 Mensajes envidos correctamente

A continuación, se muestra la Grafica 1, se muestra la representación del tiempo que se tardaron los mensajes enviados, tenemos 3 campos el *tiempo de envío* que se muestra en color azul, el *tiempo de recepción* que se muestra en color rojo y el tiempo de retardo que se muestra de color verde.



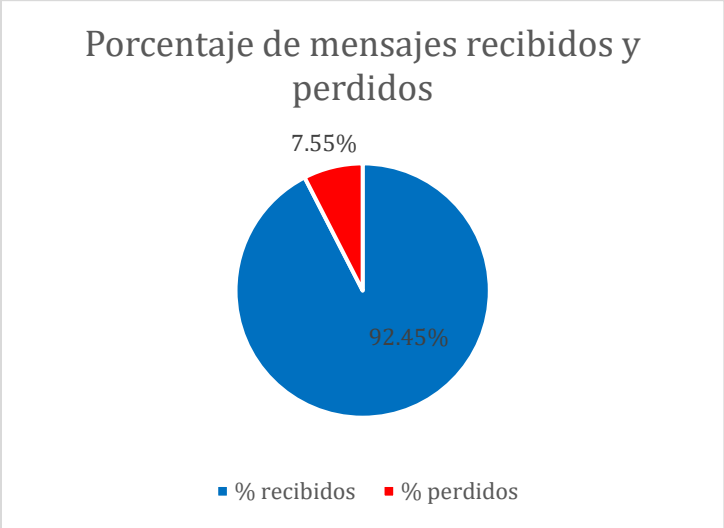
Grafica 1 Tiempo de los mensajes enviados.

En la Grafica 2 se muestran los SMS enviados de color naranja que nos dice que se enviaron 53, de color azul tenemos los SMS recibidos donde se muestran 49 SMS y de color rojo tenemos los SMS que se perdieron, en este caso fueron 4 SMS.



Grafica 2 Mensajes enviados, recibidos y perdidos.

En la Grafica 3 tenemos de color azul el porcentaje de los mensajes recibidos donde se muestra el 92.45% del 100% de los mensajes que se enviaron y de color rojo tenemos el porcentaje de los mensajes perdidos, en este caso fue el 7.55%.



Grafica 3 Porcentaje total de los mensajes recibidos y perdidos.

En la siguiente

Tabla 5, podemos ver las tablas de la base de datos que utilizamos para el almacenamiento y sincronización, donde tenemos cada una de las características de la tabla, así como el total de bits que se utilizó por registro en cada una de las tablas.

Tabla	Campo	Tamaño	Tipo de datos	Bits	Total, de bits
Base aérea	Id		Serial	32	32
	Ciudad	50	Varchar		50
	Estado	50	Varchar		50
	Total, de bits por registro				**132**
Tipo avión	Id		Serial	32	32
	Nombre	50	Varchar		50
	Total, de bits por registro				**82**

Avión	Id		Serial	32	32
	Matricula	8	Varchar		8
	Nombre	50	Vrachar		50
	Id base		Integer	32	32
	Id tipo		Integer	32	32
	Total, de bits por registro				**154**
Misión	Id		Serial	32	32
	Nombre	30	Varchar		30
	Hora inicio		Time	64	64
	Hora termino		Time	64	64
	Fecha		Date	32	32
	Id avión		Integer	32	32
	Id detalle		Integer	32	32
	Total, de bits por registro				**286**
Detalle misión	Id		Serial	32	32
	Altitud		Integer	32	32
	Latitud		Float	32	32
	Hora		Time	64	64
	Id mensaje		Integer	32	32
	Total, de bits por registro				**192**
Mensaje	Id		Serial	32	32
	Mensaje	255	Varchar		255
	Hora		Time	64	64
	Total, de bits por registro				**351**
Total, de bits en la base de datos por registro					**1197**

Tabla 5 Registro de bits por registro en la base de datos.

En la

Tabla *6* tenemos los tiempos que se registraron en la prueba, en la primera columna tenemos el tiempo de envió a la base de datos donde se registra un tiempo de 30 segundos en cada envió, la segunda columna tenemos el tiempo que se recibieron

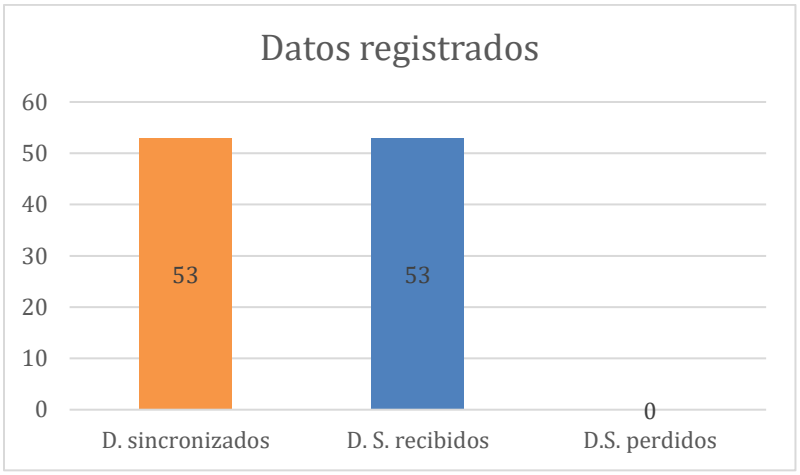
los datos a PostgreSQL, la tercera columna tenemos los tiempos en que se realizó la sincronización y en la ultima columna tenemos el tiempo de retardo que tuvo la sincronización. En esta prueba, los resultados fueron muy exitosos, ya que no tuvimos perdida de información a la hora de sincronizar.

T. Envio DB	T.Recepción DB	T.Sincronización	T. Retardo sincronozación
00:00:00	00:00:05	00:00:07	00:00:02
00:00:30	00:00:36	00:00:37	00:00:01
00:01:00	00:01:10	00:01:15	00:00:05
00:01:30	00:01:37	00:01:39	00:00:02
00:02:00	00:02:11	00:02:14	00:00:03
00:02:30	00:02:34	00:02:35	00:00:01
00:03:00	00:03:08	00:03:13	00:00:05
00:03:30	00:03:40	00:03:45	00:00:05
00:04:00	00:04:12	00:04:14	00:00:02
00:04:30	00:04:36	00:04:41	00:00:05
00:05:00	00:05:07	00:05:11	00:00:04
00:05:30	00:05:34	00:05:37	00:00:03
00:06:00	00:06:10	00:06:17	00:00:07
00:06:30	00:06:36	00:06:43	00:00:07
00:07:00	00:07:06	00:07:11	00:00:05
00:07:30	00:07:38	00:07:46	00:00:08
00:08:00	00:08:14	00:08:20	00:00:06
00:08:30	00:08:40	00:08:44	00:00:04
00:09:00	00:09:09	00:09:15	00:00:06
00:09:30	00:09:35	00:09:46	00:00:11
00:10:00	00:10:06	00:10:13	00:00:07
00:10:30	00:10:38	00:10:41	00:00:03
00:11:00	00:11:04	00:11:10	00:00:06
00:11:30	00:11:38	00:11:43	00:00:05
00:12:00	00:12:10	00:12:17	00:00:07
00:12:30	00:12:37	00:12:44	00:00:07
00:13:00	00:13:05	00:13:13	00:00:08
00:13:30	00:13:39	00:13:50	00:00:11
00:14:00	00:14:08	00:14:19	00:00:11
00:14:30	00:14:42	00:14:47	00:00:05
00:15:00	00:15:10	00:15:16	00:00:06
00:15:30	00:15:36	00:15:43	00:00:07
00:16:00	00:16:05	00:16:15	00:00:10
00:16:30	00:16:39	00:16:50	00:00:11
00:17:00	00:17:08	00:17:21	00:00:13
00:17:30	00:17:43	00:17:49	00:00:06
00:18:00	00:18:12	00:18:24	00:00:12
00:18:30	00:18:35	00:18:43	00:00:08
00:19:00	00:19:04	00:19:24	00:00:20
00:19:30	00:19:34	00:19:50	00:00:16
00:20:00	00:20:10	00:20:25	00:00:15
00:20:30	00:20:38	00:20:47	00:00:09
00:21:00	00:21:03	00:21:15	00:00:12
00:22:30	00:21:37	00:21:49	00:00:12
00:22:00	00:22:15	00:22:29	00:00:14
00:22:30	00:22:39	00:22:46	00:00:07
00:23:00	00:23:11	00:23:18	00:00:07
00:23:30	00:23:37	00:23:45	00:00:08
00:24:00	00:24:15	00:24:24	00:00:09
00:24:30	00:24:40	00:24:48	00:00:08
00:25:00	00:25:09	00:25:18	00:00:09
00:25:30	00:25:42	00:25:51	00:00:09

00:26:00	00:26:11	00:26:18	00:00:07

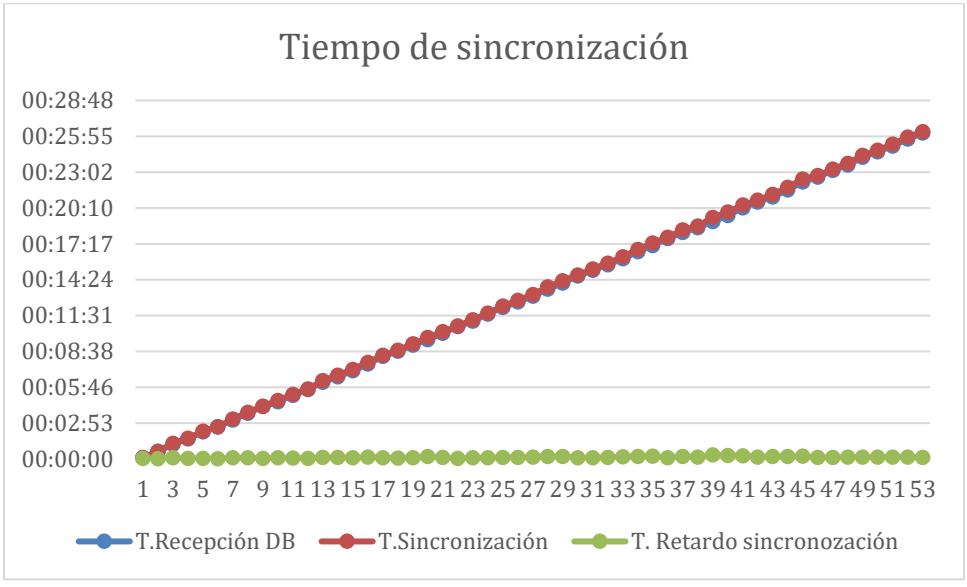
Tabla 6 Tiempos de envío para la sincronización

A continuación, se muestra la Grafica 4, donde se representan los datos registrados que se sincronizaron a la base de datos, así como los datos que se recibieron correctamente y los datos que se perdieron.



Grafica 4 Total de datos sincronizados correctamente.

La Grafica 5 nos muestra los tiempos de recepción que se tardo en llegar a la base de datos y se muestra de color azul, el tiempo que inicio la sincronización se representa de color rojo y de color verde el tiempo de retardo en la sincronización.



Grafica 5 Tiempos de recepción, sincronización y retardo.

Capítulo V
Conclusiones

En este capítulo se presentan las conclusiones después de haber desarrollado un prototipo de cifrado, transmisión y almacenamiento de datos geoespaciales con el fin de realizar un monitoreo seguro de las aeronaves, además de facilitar y

optimizar los procesos para mejorar la recuperación de datos.

Se alcanzó el objetivo fundamental de esta tesis, desarrollando un prototipo capaz de cifrar y transmitir datos geoespaciales de manera automática a través de un SMS con un tiempo de retardo de 10 segundos aproximadante entre cada mensaje, el mayor tiempo de retardo en la recepción de los mensajes es de 17 segundos y el tiempo mínimo registrado es de 2 segundos, por otro lado, desencripta correctamente los datos recibidos y recupera 100% de la información almacenándola en una base de datos geoespacial. Se logra una replicación de datos con un 100% de éxito en él envió de la información.

El uso de un algoritmo de cifrado es de suma importancia cuando se requiere transmitir información o datos sensibles que pueden estar sujetos a constantes ataques a través de un canal de comunicación público o poco seguro. AES es un algoritmo seguro, resistente a ataques de criptoanálisis lineal y diferencial, entre otros tipos de ataques.

El módulo SIM800L es un módulo práctico y sumamente económico, sin embargo, este módulo no cumple con las características para satisfacer las necesidades del cliente en cuanto a conectividad, puesto que se requiere de un dispositivo que tenga buena cobertura desde un avión y aunque el módulo logra conectarse a la red teniendo una intensidad mínima 23 dBm en la señal, puede perder la conexión con facilidad y esto ocasiona que los paquetes de datos dejen de ser transmitidos o que tarden demasiado tiempo en llegar al destinatario. La velocidad de recepción de los mensajes dependerá de la calidad de señal que tiene el módulo y también depende de la señal del móvil que recibe los datos.

La replicación con Bucardo en PostgreSQL es una herramienta muy completa, generalmente se instala en una base de datos maestro, funciona con nodos, donde se indica la ubicación de cada uno de los nodos y se ejecutan en un demonio Perl, que se encarga de la replicación. Además, cuenta con la resolución de conflictos en

modo multimaestro y la propagación es en cascada, esto facilita el trabajo a la hora de replicar los datos. La extensión PostGIS convierte una base de datos normal a una base de datos espacial y la combinación de ambos es una gran solución para el almacenamiento, gestión y mantenimiento de datos espaciales.

El presente proyecto se puede usar para futuros trabajos relacionados con la codificación y decodificación de datos transmitidos a través de un módulo SIM800L y mejorado con una aplicación móvil que cuente con los permisos necesarios para recibir los datos cifrados directamente en forma de SMS y así lograr un descifrado más fácil y rápido desde la misma aplicación. Además, puede mejorar agregando algún módulo de seguridad dentro de la base de datos. También podemos implementar la herramienta QGIS en el proyecto, proporcionando un visor de datos SIG para la base de datos y optimizar algunos procesos en general para su mejor funcionamiento.

Referencias

[1] D. Montenegro Torres, «Comparacion de algoritmos de encriptación para la transferencia de archivos en mensajería instantánea(tesis de pregrado),» Pimentel, 2020.

[2] T. W. Carrera Albán, «Análisis y comparación de dos algoritmos de cifrado simétrico en plataformas Windows(tesis de pregrado),» Samborondón, 2016.

[3] J. C. Meza Roman y V. G. Leaño Pariona, «Sistema de monitoreo de una red de buses de transporte público e información para usuarios empleando transceptore GPS/GSM(Tesis de Pregrado),» Lima, 2017.

[4] D. M. Chiapella, «Instrumento de medición de factores climáticos,» 2020.

[5] E. L. Llanos Mora, «Diseño de un sistema inalámbrico de monitorio para pacientes epilépticos de la clínica Anglo Americana,» Lima, 2020.

[6] A. C. Lara Barrera, «Aplicación web mapping para la visualización y consulta de información en la gestión municipal a partir de herramientas de código abierto,» Instituto Tecnológico de Culiacán , Sinaloa , 2017.

[7] C. R. Montaño Espinoza, «Diseño e implementación de una geobase de datos distribuidas, en un ambiente virtualizado, acoplada a un sistema de información geográfica,» Morelos, 2016.

[8] A. Mejía Olivares, «Desarrollo e implementación de una visualizador web geográfico sobre aguas termales en el estado de México,» Toluca de Lerdo, 2017.

[9] F. Radilla López, «Modelado de datos para base de datos espaciales. Caso de estudio: sistemas de información geográfica,» Ciudad de México, 2018.

[10] V. Olaya, «Historia de los SIG,» de *Sistemas de Información Geográfica*, New York, CreateSpace Independent Publishing Platform, 2014, p. 25.

[11] C. S. L. A. Campos Paréz Rafael, «Concepto y origen de las BD y de los SGBD,» de *Bases de Datos*, Barcelona, Eureca Media, 2005, p. 7.

[12] V. Olaya, «Sistemas de Gestores de Bases de Datos,» de *Sistemas de Información Geográfica*, New York, CreateSpace Independent Publishing Platform, 2014, p. 214.

[13] M. V. Mannino, «Introducción a la Administracion de Bases de Datos,» de *Administracion de Bases de Datos Diseño y Desarrollo de Aplicaciones*, D.F., McGRAW-HILL/INTERAMERICANA, 2017, p. 6.

[14] A. Mora Rioja, «Sistemas de Almacenamiento de la Información,» de *Bases de Datos Diseño y Gestión*, Madrid, Síntesis, S.A., 2014, p. 20.

[15] V. Olaya, «Bases de Datos Espaciales,» de *Sistemas de Información Geográfica*, New York, CreateSpace Independent Publishing Platform, 2014, p. 220.

[16] I. Gilfillan, «Guia rapida para MySQL,» de *La Biblia de MySQL*, Madrid, Anaya Multimedia, 2000, p. 41.

[17] O. a. i. affiliates, «Extenciones espaciales de MySQL,» de *MySQL 5.0 Reference Manual*, New York, Oracle, 2014, p. 992.

[18] A. Chavez, «live.osgeo,» 2017. [En línea]. Available: https://live.osgeo.org/es/overview/spatialite_overview.html#:~:text=Base%20de%20Datos%20Espacial,SQLite%20es%20simplemente%20un%20archivo.. [Último acceso: 22 Septiembre 2020].

[19] R. Estévez, «geomapik,» 26 09 2019. [En línea]. Available: http://www.geomapik.com/desarrollo-programacion-gis/que-es-postgis/. [Último acceso: 25 09 2020].

[20] V. Olaya, «Evolución de las bases de datos en los SIG,» de *Sistemas de Información Geográfico*, New York, CreateSpace Independent Publishing Platform, 2014, p. 221.

[21] M. R. J. R. R. C. F. F. Zea Ordóñez Mariuxi Paola, «Replicación,» de *Administración de Bases de Datos con PostgreSQL*, San Lus, Área de Innovación y Desarrollo, 2017, p. 12.

[22] H. T. Álvaro, «PDFSlide,» 05 Marzo 2014. [En línea]. Available: https://pdfslide.tips/technology/replicacion-y-alta-disponibilidad-en-postgresql.html. [Último acceso: 23 Octubre 2020].

[23] A. G. Marcelo, «biblioteca digital,» 20 Agsoto 2015. [En línea]. Available: http://bibliotecadigital.uda.edu.ar/objetos_digitales/525/tesis-4849-confiabilidad.pdf. [Último acceso: 02 Octubre 2020].

[24] B. Christopher, «slony.info,» 15 Agosto 2013. [En línea]. Available: https://www.slony.info/adminguide/2.1/doc/adminguide/slony.pdf. [Último acceso: 01 Octubre 2020].

[25] R. B. V. Milagros, «Universidad Central “Marta Abreu” de las Villas,» 22 Mayo 2011. [En línea]. Available: https://dspace.uclv.edu.cu/bitstream/handle/123456789/5789/TESIS-%20Modelo%20de%20Replicaci%C3%B3n%20de%20Datos%20para%20el%20Sistema%20de%20Controll%20y%20Cobro%20de%20Multas.pdf?sequence=1&isAllowed=y. [Último acceso: 26 Octubre 2020].

[26] E. M. Alejandro, «ebuah,» 16 Septiembre 2017. [En línea]. Available: https://ebuah.uah.es/dspace/bitstream/handle/10017/32022/TFG-Escobar-Mart%C3%ADn-2017.pdf?sequence=1&isAllowed=y. [Último acceso: 03 Octubre 2020].

[27] cybertec, «cybertec,» 01 Enero 2020. [En línea]. Available: https://www.cybertec-postgresql.com/es/servicios/replicacion-postgresql/replicacion-sincronica-asincronica/. [Último acceso: 04 Octubre 2020].

[28] M. C. Javier, Artist, *Replicación asíncrona de base de datos.* [Art]. Instituto de Investigaciones en Ciencia y Tecnología, 2015.

[29] T. R. Rommel, «researchgate,» 01 Agosto 2018. [En línea]. Available: https://www.researchgate.net/publication/331742536_ANALISIS_Y_DISENO_DE_TECNICAS_DE_REPLICACION_DE_UNA_BASE_DE_DATOS_HIBRIDA_Y_DISTRIBUIDA_EN_UN_AMBIENTE_DE_ANCHO_DE_BANDA_LIMITADO. [Último acceso: 15 Octubre 2020].

[30] Programmer, «programmerclick,» 01 enero 2020. [En línea]. Available: https://programmerclick.com/article/22283200/. [Último acceso: 20 Octubre 2020].

[31] F. M. Israel, «Isra Blog,» 10 Noviembre 2011. [En línea]. Available: https://iffm.me/configurando-replicas-maestro-esclavo-con-postgres.html. [Último acceso: 22 Octubre 2020].

[32] Linux, «Linux-console.net,» 01 Febrero 2019. [En línea]. Available: https://es.linux-console.net/?p=1368. [Último acceso: 22 Octubre 2020].

[33] A. García Gónzalez, «PanamaHitek,» 23 Noviembre 2016. [En línea]. Available: http://panamahitek.com/arduino-java-facil-y-rapido/. [Último acceso: 15 Diciembre 2020].

[34] «Grupo EBIM,» 20 Julio 2020. [En línea]. Available: https://www.ebim.pe/post/panamahitek-arduino-java. [Último acceso: 18 09 2020].

[35] E. L. Llanos Mora, «Diseño de un sistema inalámbrico de monitorio para pacientes epilépticos de la clínica Anglo Americana,» 2020. [En línea]. Available: https://repositorio.utp.edu.pe/bitstream/handle/20.500.12867/3125/Erika%20Llanos_Trabajo%20de%20Suficiencia%20Profesional_Titulo%20Profesional_2020.pdf?sequence=1&isAllowed=y. [Último acceso: 20 12 2020].

[36] P. Porcuna López, Robótica y Domótica básica con Arduino, Madrid: RA-MA, 2016.

[37] P. Deitel y H. Deitel, Java como programar, CDMX: Pearson, 2012.

[38] J. Martínez Ladrón de Guevara, «Fundamentos de programación en Java,» 2020. [En línea]. Available: https://www.tesuva.edu.co/phocadownloadpap/Fundamentos%20de%20programacion%20en%20Java.pdf. [Último acceso: 2020 12 11].

[39] S. Oask, Java Performance In-Depth Advice for Tuning and Programming Java 8,11 and Beyond, California: O'Reilly, 2020.

[40] «Digital Guide IONOS,» 12 Septiembre 2017. [En línea]. Available: https://www.ionos.mx/digitalguide/servidores/seguridad/todo-sobre-los-metodos-de-encriptado/. [Último acceso: 15 Diciembre 2020].

[41] J. M. Ortega Candel, «7.2 JCA. Java Cryptography Architecture,» de *Seguridad en aplicaciones Web Java*, Madrid, Ra-Ma, 2018, p. 311.

[42] S. Talens-Oliag, «www.uv.es,» [En línea]. Available: https://www.uv.es/~sto/cursos/seguridad.java/html/sjava-40.html. [Último acceso: 15 Diciembre 2020].

[43] D. Montenegro Torres, «Comparacion de algoritmos de encriptación para la transferencia de archivos en mensajería instantánea(tesis de pregrado),» 2020. [En línea]. Available: https://www.bing.com/search?q=libro+de+Java&qs=n&form=QBRE&msbsrank=3_3__0&sp=-1&pq=libro+de+java&sc=3-13&sk=&cvid=A2C42BB95D1E44B99E193ED0DBF3B429. [Último acceso: 15 11 11].

[44] T. W. Carrera Albán, «Análisis y comparación de dos algoritmos de cifrado simétrico en plataformas Windows(tesis de pregrado),» Febrero 2016. [En línea]. Available: https://1library.co/document/zx5k3l4q-analisis-comparacion-algoritmos-cifrados-simetrico-plataformas-windiws.html. [Último acceso: 15 10 2020].

[45] J. Botaya Villanúa, «CHAT SEGURO,» 2005.

[46] D. Montenegro Torres, «Comparacion de algoritmos de encriptación para la transferencia de archivos en mensajería instantánea(tesis de pregrado),» 2020. [En línea]. Available: https://www.bing.com/search?q=libro+de+Java&qs=n&form=QBRE&msbsrank=3_3__0&sp=-1&pq=libro+de+java&sc=3-13&sk=&cvid=A2C42BB95D1E44B99E193ED0DBF3B429. [Último acceso: 15 11 2020].

[47] J. Botaya Villanúa, «CHAT SEGURO,» 10 febrero 2010. [En línea]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/674/1/34993tfc.pdf. [Último acceso: 2020 septiembre 28].

[48] L. A. Lezama Quintero, «Course Hero,» 3 Septiembre 2020. [En línea]. Available: https://www.coursehero.com/file/67755108/Advanced-encryption-standardpdf/. [Último acceso: 10 10 2020].

[49] J. L. Pérez, «COMUNYCARSE,» 4 Diciembre 2017. [En línea]. Available: https://www.comunycarse.com/es/que-es-sms-y-como-funciona/. [Último acceso: 15 Diciembre 2020].

[50] E. L. Llanos Mora, «Diseño de un sistema inalámbrico de monitorio para pacientes epilépticos de la clínica Anglo Americana,» 2020. [En línea]. Available: https://repositorio.utp.edu.pe/bitstream/handle/20.500.12867/3125/Erika%20Llanos_Trabajo%20de%20Suficiencia%20Profesional_Titulo%20Profesional_2020.pdf?sequence=1&isAllowed=y. [Último acceso: 15 Octubre 2020].

[51] J. C. Meza Roman y V. G. Leaño Pariona, «Sistema de monitoreo de una red de buses de transporte público e información para usuarios empleando transceptore GPS/GSM(Tesis de Pregrado),» 28 11 2017. [En línea]. Available: http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/9781. [Último acceso: 18 12 2020].

[52] «ALTIRIA TIC,» 08 Diciembre 2018. [En línea]. Available: https://www.altiria.com.mx/2927/sms-como-canal-de-comunicacion-de-la-empresa/. [Último acceso: 3 Noviembre 2020].

[53] P. A. Fain, «Pablo Alejandro Fain,» 14 enero 2009. [En línea]. Available: https://blog.pablofain.com/2009/03/14/el-camino-de-un-sms/. [Último acceso: 20 octubre 2020].

[54] T. A. Óscar, Arduino curso práctico de formación, CDMX: Alfaomega, 2013.

[55] M. Sabino, «metacpan,» 13 Agosto 2020. [En línea]. Available: https://metacpan.org/pod/DBD::Pg. [Último acceso: 20 Octubre 2020].

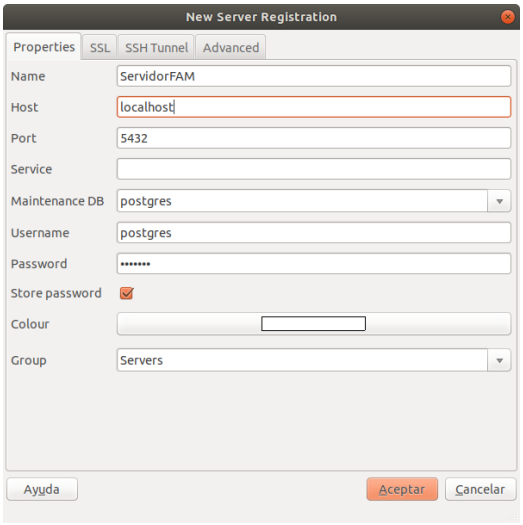
[56] G. Marcelo, «biblioteca digital,» 20 Agsoto 2015. [En línea]. Available: http://bibliotecadigital.uda.edu.ar/objetos_digitales/525/tesis-4849-confiabilidad.pdf. [Último acceso: 02 Octubre 2020].

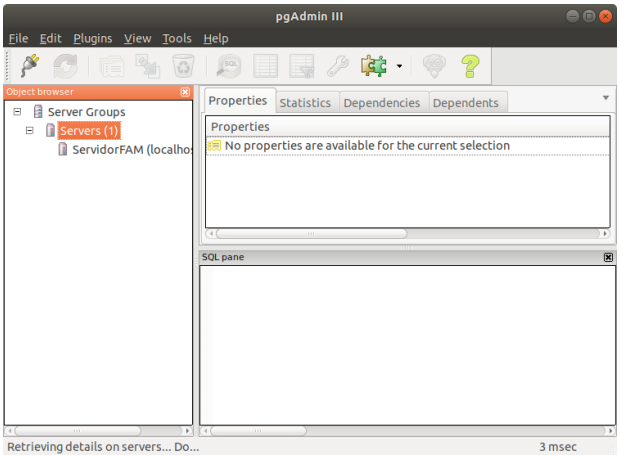
[57] «Instituto Federal de Telecomunicaciones,» 2015. [En línea]. Available: http://www.ift.org.mx/usuarios-telefonia-movil/sabias-que-la-telefonia-movil#:~:text=Las%20bandas%20de%20frecuencias%20del,%2F%202110%2D2180%20MHz).. [Último acceso: 12 12 2020].

ANEXO A. MANUAL TECNICO

Configuración del SGBD

Usamos la interfaz gráfica de *pgAdmin3* y nos conectamos dando clic en el icono “agregar una conexión a un servidor” y nos mostrará la siguiente ventana, como se muestra en la ilustración 1, donde nos pedirá un nombre para el servidor, el host que en este caso es la dirección *IP* de la máquina, el puerto que es 5432, usuario y contraseña y le daremos en el botón “Aceptar”, en la interfaz de *pgAdmin3* se podrá ver el servidor creado.





Esquema Bucardo y exención PostGIS.

Para poder instalar *Bucardo*, necesitamos primero instalar los siguientes módulos, para el funcionamiento correcto de *bucardo*, como es el módulo para el acceso a la base de datos, el módulo para obtener el nombre del host, el módulo para facilitar él envió de mensajes al demonio y el módulo para el acceso seguro y manipulación de la base de datos, como se muestra en la siguiente línea de comando.

```
sudo apt-get install libboolean-perl libdbd-mock-perl libdbd-pg-perl
libanyevent-dbd-perl libpg-hstore-perl libpgobject-perl
```

Accedemos a la carpeta *BDIx* y creamos un nuevo fichero, en *DBIx – Safe*. colocando la siguiente línea de comando, como se muestra a continuación:

```
sudo DBIx-Safe-1.2.5# perl Makefile.PL
```

Una vez instalado los módulos y creado el fichero, vamos a instalar la herramienta *bucardo*, colocando la siguiente línea de comando:

```
sudo wget http://bucardo .org/downloads/Bucardo-5.4.1.tar.gz
```

Para la instalación de *postgis*, podemos hacerlo con la siguiente línea de comando:

```
sudo apt-get install postgis
```

una vez que se haya instalado *postgis*, se deberá entrar a *postgres* y seleccionar la base de datos, donde se creara la extensión de *postgis*, como se muestra en la siguiente línea de comando:

```
sudo su - postgres
```

```
postgres=# \c DB1
```

```
DB1=# CREATE EXTENSION postgis;
```

Creación de las tablas de la base de datos

Creamos la tabla *base aéreas*, donde se almacenará el id, ciudad y estado en el que se encuentra la base aérea, el id es de tipo serial, esto quiere decir que se va incrementar automáticamente y va ser nuestra llave primaria, los otros dos campos van hacer de tipo varchar y almacenar 50 caracteres.

```
create table BasesAereas(
     id_base serial primary key,
     ciudad varchar(50) not null unique,
     estado varchar(50) not null
);
```

Creamos la tabla *tipo avión*, donde se almacenará el id de tipo serial y será nuestra llave primaria, el nombre de tipo varchar que almacenará 50 caracteres, del tipo de avión que se tenga registrado en la base.

```
create table TipoAvion(
     id_tipo serial primary key,
     nombre varchar(50)
);
```

Creamos la tabla *Avión*, que contará con su id de tipo serial y será nuestra llave primaria, matrícula de tipo varchar con la capacidad de almacenar 8 caracteres, nombre de tipo varchar con la capacidad de almacenar 50 caracteres, id de la base y el id del tipo de avión de tipo entero, en esta tabla la relacionamos con la tabla bases aéreas y tipo avión para obtener sus datos que contengan.

```
create table Avion(
     id_avion serial primary key,
     matricula varchar(8) not null unique,
     nombre varchar(50) not null,
     id_base integer not null,
     id_tipo integer not null,
     foreign key (id_base) references BasesAereas (id_base),
     foreign key (id_tipo) references TipoAvion (id_tipo)
);
```

Creamos la tabla *detalle misión*, que contendrá los campos id de tipo serial y será nuestra llave primaria, altitud de tipo entero, latitud de tipo flotante, hora de tipo tiempo y el id del mensaje para obtener los datos de la tabla mensaje, esta tabla almacena los datos que manda la aplicación de Android, como son las coordenadas y el mensaje.

```
create table DetalleMision(
     id_detalle serial primary key,
     altitud integer not null,
     latitud float not null,
     hora time not null,
     id_mensaje integer,
     foreign key (id_mensaje) references Mensaje (id_mensaje)
);
```

Creamos la tabla *Mensaje*, que contiene el id de tipo serial que será nuestra llave primaria, mensaje de tipo varchar que puede almacenar 255 caracteres y la hora en la que se recibe el mensaje, esta tabla tiene como fin almacenar los mensajes que se mandan desde la aplicación.

```
create table Mensaje(
     id_mensaje serial primary key,
     mensaje varchar(255) not null,
     hora time default current_time
);
```

Procedimientos almacenados

Creamos la función *Alta detalle*, que contendrá 3 campos, altitud, latitud y hora que son los mismos campos que tiene la tabla detalle misión y empezamos insertando los datos en los campos de la tabla detalle misión y finalizamos la función.

```
create or replace function AltaDetalle(
paltitud integer,
platitud float,
phora time)
returns void as
$$
begin
insert into DetalleMision(altitud, latitud, hora) values(paltitud, platitud, phor
a);
end
$$ language plpgsql;
```

Creamos la función Alta mensaje, declaramos el campo mensaje que es el mismo campo que tiene la tabla Mensaje y empezamos insertando los datos en los campos de la tabla Mensaje y finalizamos la función.

```
create or replace function AltaMensaje(
pmensaje varchar) la table detalle mis
returns void as
$$
begin
insert into Mensaje(mensaje) values(pmensaje);
end;
$$ language plpgsql;
```

Código de aplicación en java

Se crea un nuevo proyecto en Apache *Netbeans IDE 12.0*, la ventana principal recibe el nombre “principal”, el diseño se conforma por etiquetas, cajas de texto y botones, como se muestra en Ilustración 1.

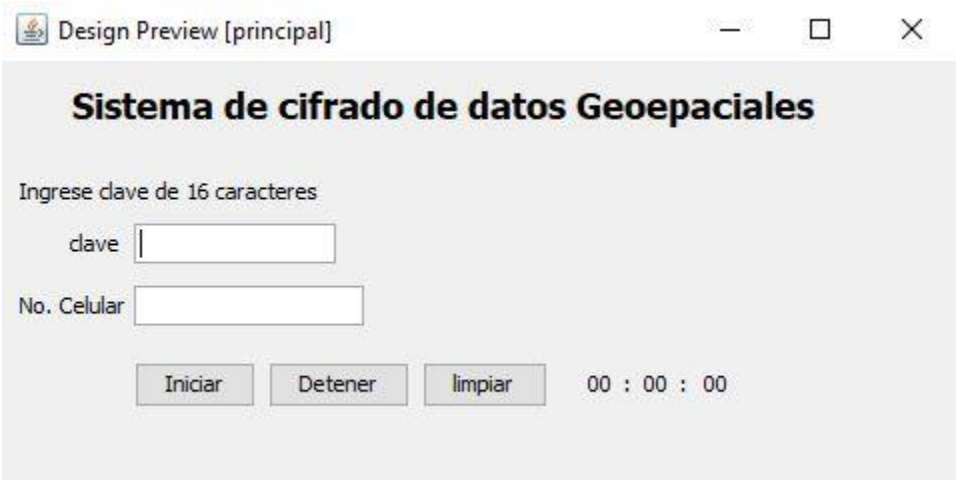


Ilustración 1 Interfaz gráfica de usuario

Posteriormente se carga la librería de *panamaHitek* al proyecto para poder hacer uso de sus clases. Una vez hecho esto se puede iniciar la codificación de la aplicación.

Se crean los objetos y variables que se utilizarán a lo largo del proyecto. El objeto *PanamHitek_Arduino* es un objeto de la librería previamente cargada y sirve para hacer la conexión entre Java y Arduino, también se crea un objeto de la clase *ArrayList* para almacenar los datos que recibe a través del transmisor, por otro lado, se definen un objeto de la clase de tipo *Hilo*, estas clases se describen más adelante. Finalmente, una variable de tipo *String* que sirve para recibir la clave de cifrado.

```
PanamaHitek_Arduino ino =  new  PanamaHitek_Arduino ();
PanamaHitek_Arduino inoSend =  new  PanamaHitek_Arduino ();

public class principal extends javax.swing.JFrame {
    public   ArrayList lista = new ArrayList();
    public   HiloTime ht;
    public   String cc="";
    public principal() {
        initComponents ();
    }
```

Ahora se crea un oyente que se encargará de escuchar los eventos que ocurran en el puerto serial, este oyente contiene un evento que muestra los datos recibidos por

el puerto serial en caso de que exista alguno, se debe incluir un try y catch para recibir los errores si estos llegasen a existir.

En caso de que el puerto serial este recibiendo datos del transmisor se valida que el contador sea menor o igual a 100, esto nos ayuda a liberar memoria ya que si el contador llega a 100 se reinicia la lista y el contador. Se agrega a la lista el mensaje que se obtiene, posteriormente se instancia el objeto de la clase AES y se le asigna la clave.

Posteriormente se crea una variable de tipo *String* la cual se va a igualar al valor que retorna el método de cifrado, a dicho método se le tiene que pasar como parámetro el valor de la lista en la posición en la que se encuentre el iterador.

Ahora se utiliza el objeto *inoSend* para invocar el método *SentData* y poder enviar el número de teléfono del destinatario y el contenido del mensaje que serán recibidos por Arduino.

```
import com.panamahitek.ArduinoException;
import com.panamahitek.PanamaHitek_Arduino;

      PanamaHitek_Arduino ino =  new  PanamaHitek_Arduino ();
      PanamaHitek_Arduino inoSend =  new  PanamaHitek_Arduino ();
      SerialPortEventListener listener =  new SerialPortEventListener () {

        public  void serialEvent ( SerialPortEvent  spe ) {
            try {
                if (ino.isMessageAvailable ()) {

                       if(i<=100){
                        lista.add(ino.printMessage());

                        AES aes1=new AES(cc);
                        String smsCifrado=aes1.cifrado(lista.get(i).toString());
                        inoSend.sendData(txtNumero.getText()+"\n");
                        inoSend.sendData(smsCifrado+"\n");

                        System.out.println(lista.get(i).toString());

                        i++;
                    }
                    else{
                        lista = new ArrayList();
                        i = 0;
                }
              }


            } catch (SerialPortException ex) {
                Logger.getLogger(principal.class.getName()).log(Level.SEVERE,
null, ex); }
```

Código de cifrado

Es necesario crear una clase llamada AES, donde se declara un array de bytes para almacenar el valor de la clave. Se crea el constructor de la clase que recibirá como parámetro una variable de tipo String, esta variable contendrá la clave que el usuario ingrese en la interfaz gráfica.

```
class AES{

        private byte[] valClave;

        public AES(String key){
            valClave= key.getBytes();
    }
}
```

Posteriormente se crea el método que va a transformar los caracteres recibidos en una clave de tipo AES, como podemos observar se declara un objeto de tipo *Key* y se utiliza la clase *SecretKeySpec*, esta clase recibe como parámetro un arreglo de bytes por lo tanto se coloca el nombre del arreglo declarado al inicio y también recibe el tipo de cifrado, en este caso es el cifrado AES, el método nos devuelve la clave. Este método va a retornar la clave de cifrafo.

```
public Key generarClave(){

    Key key = new SecretKeySpec(valClave,"AES");
    return key;
}
```

Por último, se crea un método público que devuelve una cadena y recibe de igual manera un dato de tipo String, en este caso será el mensaje que se desea cifrar. Se genera un objeto de tipo Key, dicho objeto será igual al método que genera la clave de cifrado, después se crea una instancia del tipo de cifrado que se utilizará para esto se utiliza la clase *Cipher* y el método *getInstance.* Se inicia el modo de encriptación y se asigna la clave generada para que se pueda cifrar.

Una vez hecho esto se debe crear un arreglo de bytes donde se guarda el mensaje cifrado.

Para finalizar se crea una variable de tipo string donde se guardan los datos del arreglo de bytes con el mensaje cifrado. Esta variable contendrá el valor que

devolverá el método de cifrado. Ahora es posible cifrar cualquier mensaje y guardar su contenido para un posterior uso.

```
public String cifrado(String mensaje) throws InvalidKeyException, NoSuchAlgorithm
Exception, IllegalBlockSizeException, NoSuchPaddingException,
BadPaddingException, UnsupportedEncodingException{

      Key key=generarClave();
      Cipher cipher= Cipher.getInstance("AES");
      cipher.init(Cipher.ENCRYPT_MODE, key);
      byte [] encVal=cipher.doFinal(mensaje.getBytes("UTF-8"));
      return Base64.getEncoder().encodeToString(encVal);

}
```

Este código se llama desde un hilo que se encarga de leer el archivo con los datos, cifrarlos y enviarlos en un intervalo de 15 segundos.

La clase HiloTime

Esta clase sirve para hacer funcionar el cronometro que aparece en la interfaz gráfica, el cual nos indicará cuanto tiempo se tardó en completarse la lectura del archivo de texto y el envió de los datos. Para desarrollar esta clase se define la clase llamada HiloTime que extiende de la clase Thread, se implementa el método run y se condiciona con una sentencia while, mientras el hilo este corriedo se dormirá 1 segundo, la etiqueta que lleva el registro de los segundos muestra el valor inicial que recibe al cual se le suma 1 para que pueda incrementar por cada iteración, si la etiqueta llega al numero “60” se inicia el mismo proceso para la etiqueta de los minutos y la etiqueta de segundos regresa a “0”, se utiliza el mismo proceso para la etiqueta que registra las horas.

```
class HiloTime extends Thread {
 public void run(){
 try{
  while(true){

  Thread.sleep(1000);
  lblsegundos.setText(String.valueOf(Integer.parseInt(lblsegundos.getText())+1));
  if(

    lblsegundos.getText().equals("60")==true){
    lblminutos.setText(String.valueOf(Integer.parseInt(lblminutos.getText())+1));
    lblsegundos.setText("0");

    }
  if(
```

```
    lblminutos.getText().equals("60")==true){
    lblhoras.setText(String.valueOf(Integer.parseInt(lblhoras.getText())+1));
    lblminutos.setText("0");

    }

   }
 }
 catch(Exception e){

      System.out.println("ERROR EN HILO TIMER"+ e);
  }
 }
}
```

Al tener estas clases desarrolladas se puede continuar con la explicación del funcionamiento de los botones.

El botón “iniciar” crea un nuevo objeto de tipo HiloTime, si el cuadro de texto de la clave y número telefónico es vacío, nos arroja un mensaje para llenar los campos, si la clave de cifrado no son 16 caracteres de igual forma se muestra un mensaje al usuario para que ingrese la clave correctamente para iniciar el proceso, dando inicio también al hilo del tiempo.

```
private void btnPlayActionPerformed(java.awt.event.ActionEvent evt) {

ht= new HiloTime();
cc=txtclave.getText();
            if(cc.equals("") || txtNumero.getText().equals("")){
                 JOptionPane.showMessageDialog(null,"Favor de llenar los
campos");
            }
            else {
                if(cc.length()<16){
                JOptionPane.showMessageDialog(null,"la contraseña debe ser de 16
caracteres exactos,"+
                        " Por favor intente otra vez");
                 }
                else {
                        lblclave.setText(cc);
                        ht.start();
                        arduino();
                }
            }
```

El botón “*Detener*” se detienen los procesos que se usan como es el objeto de *ino*, *ino* y *send* de *PanamaHitek*, también se detiene el objeto *ht* del hilo del tiempo.

```
private void btnStopActionPerformed(java. awt.event.ActionEvent evt) {
```

```
        try {
            ino.killArduinoConnection();
            inoSend.killArduinoConnection();
            ht.stop();
            JOptionPane.showMessageDialog(null,"Proceso Finalizado");
        } catch (ArduinoException ex) {
            Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
```

Código en Arduino

Se utilizan los siguientes comandos AT:

- AT: Este comando sirve para verificar que Arduino se logró conectar con el módulo de manera correcta, en caso de ser así, el comando devolverá OK.
- AT+CSQ: Comando que indica la intensidad de señal que recibe la tarjeta SIM.
- AT+CCID: Devuelve el nombre de la tarjeta SIM utilizada
- AT+CREG? : Indica si el moduo se ha logrado conectar dar de alta en la red, para verifiacar esto el comando devuelte 0,1.

```
#include<SoftwareSerial.h>

#define DEBUG(a) Serial.println(a);

SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2
char output;

void setup(){

  Serial.begin(9600);
  mySerial.begin(9600);

  Serial.println("iniciando");
  delay(1000);

  mySerial.println("AT");
  updateSerial();
  mySerial.println("AT+CSQ");
  updateSerial();
  mySerial.println("AT+CCID");
  updateSerial();
  mySerial.println("AT+CREG?");
  updateSerial();

}
```

Por otro lado, al código de Arduino se le agrego la función loop que sirve para enviar el SMS, donde el comando AT+CMFG=1 realiza la configuración del módulo a modo texto y el comando “AT+CMGS=\” sirve para indicar el número telefónico al cual se enviará el mensaje.

```
void loop(){
  Serial.print("Introduzca un numero de movil: ");
  char remoteNumber[20];
  readSerial(remoteNumber);
  Serial.println(remoteNumber);
  Serial.print("escriba el contenido del SMS: ");
  char txtMsg[200];
  readSerial(txtMsg);
  Serial.println(txtMsg);

   mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
   updateSerial();
   mySerial.print("AT+CMGS=\"");
   mySerial.print(remoteNumber);
   mySerial.print("\"\r");
   updateSerial();
   mySerial.print(txtMsg); //text content
   updateSerial();
   mySerial.write(26);

   Serial.println("\nCOMPLETADO!\n");

 }
```

¡Error! No se encuentra el origen de la referencia.

La función *readSerial* sirve para leer la cadena que se reciba por el puerto Serial, en este caso lo que reciba desde la aplicación en Java. La siguiente figura muestra el código de la función *readSerial.*

```
int readSerial(char result[])
 {
  int i = 0;
   while(1)
   {
  while (Serial.available() > 0)
     {
       char inChar = Serial.read();
       if (inChar == '\n')
       {
         result[i] = '\0';
         Serial.flush();
         return 0;
       }
       if(inChar!='\r')
       {
         result[i] = inChar;
         i++;
       }
     }
   }
 }
```

Código de aplicación móvil para descifrar datos y almacenarlos

Esta aplicación sirve para probar que los mensajes pueden ser descifrados y para almacenarlos en la base de datos para una replicación de estos. Para el desarrollo de esta aplicación se utilizaron cajas de texto y tres botones, en esta aplicación se implementa el algoritmo de descifrado para mostrar el contenido real de los mensajes.

Se desarrolló una clase llamada AES, esta clase contiene el método para generar la clave y también el método para el descifrado de los mensajes.

```
import android.util.Base64;

import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import static java.text.DateFormat.DEFAULT;

class AES{
    private byte[] valClave;
    public AES(String key){
        valClave= key.getBytes();
    }
    public Key generarClave(){
        Key key = new SecretKeySpec(valClave,"AES");
        return key;
    }
    public String descifrado(String mensajeEnc) throws NoSuchAlgorithmException,
NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException,
BadPaddingException, IOException {

        Key key=generarClave();
        Cipher cipher= Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte[] valordec = cipher.doFinal(Base64.decode(mensajeEnc,DEFAULT));
        return new String(valordec);

    }
}
```

Ahora esta clase debe ser llamada desde la actividad principal, primero se instancian los objetos de la interfaz, se declara una variable de tipo String que guarda el valor del mensaje descifrado.

Se crea un método que será utilizado por un botón, cada vez que el botón “descifrar” se oprima se ejecutara esta función llamada “Descifrar”, en este método se crea un objeto de la clase AES que recibe como parámetro el contenido del campo de texto que contendrá la clave de cifrado, después se llama al método de la clase a través del objeto y se guarda el resultado en la variable String, para finalizar esta función se muestra el contenido de la variable en una caja de texto.

```
public void Descrifrar(View v) throws NoSuchPaddingException, NoSuchAlgorithmExce
ption, IllegalBlockSizeException,
BadPaddingException, InvalidKeyException, IOException {

        AES aes=new AES(txtClave.getText().toString());
        smsDes=aes.descifrado(txtMensajeCifrado.getText().toString());
        txtMensajeDescifrado.setText(smsDes);
    }
```

Para guardar los datos en la base de datos, se tiene una función que será llamada desde el botó “Guardar”, en esta función se crea una instancia de la clase “*conexiónbd*”, posteriormente se guarda el contenido del cuadro de texto que contiene el mensaje descifrado en una variable de tipo *string* y se crea una condición, si el texto contiene comas significa que el mensaje contiene las coordenadas y debe crear un arreglo de tipo *string* para almacenar el contenido de la cadena de texto separado con comas, esto se logra con la ayuda de la función *split*.

Después, se crean las variables para la longitud, latitud y hora que serán igual al contenido del arreglo respectivamente. Finalmente se llama al stored procedure *AltaDetalle* que se encargara de almacenar el contenido de estas variables en la base de datos.

Si el mensaje leído no contiene comas, se llama al stored procedure *Mensaje* que se encarga de almacenar el contenido del mensaje en la base de datos y en la tabla correspondiente.

```
public void Guardar(View v){
try{
   Conexionbd con=new Conexionbd();

     String mensajeDec= txtMensajeDescifrado.getText().toString();
    if(mensajeDec.contains(",")){
     String[]  coordenadas=mensajeDec.split(",");
     String longitud = coordenadas[0]; // 123
     String latitud = coordenadas[1];
     String hora = coordenadas[2];

     String storeProcedureCall="{CALL AltaDetalle(?,?,?)}";
     CallableStatement cStmt=con.conexionBD().prepareCall(storeProcedureCall);
    //Estos dos primeros parametros son los de entrada
         cStmt.setString(1,longitud);
         cStmt.setString(2,latitud);
         cStmt.setString(3,hora);
         cStmt.executeUpdate();
     }

   else{

        String storeProcedureCall="{CALL Mensaje(?)}";
        CallableStatement cStmt=con.conexionBD().prepareCall(storeProcedureCall);
                //Estos dos primeros parametros son los de entrada
                cStmt.setString(1,txtMensajeDescifrado.getText().toString());
                cStmt.executeUpdate();}
      }
catch (SQLException e) {
          e.printStackTrace();
      }
}
```

Conexión DB

Para la conexión entre la aplicación y la base de datos, creamos la clase conexión, que tendrá una función para conectarnos a *PostgreSQL*, cargando el controlador, para juego obtener la conexión utilizando el método *DriveManger*, colocando la dirección IP de la máquina, nombre de la base de datos, usuario y contraseña y comprobamos la conexión con algunos mensajes en la consola para verificar la conexión. También tenemos la función para cerra la conexión.

```
public class Conexion {

    Connection conexion=null;

    //Creamos nuestra funcion para Conectarnos a Postgresql
```

```
    public  Connection conexionBD(){
        try{
            StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder(
).permitAll().build();
            StrictMode.setThreadPolicy(policy);
            Class.forName("org.postgresql.Driver");
            conexion = DriverManager.getConnection("jdbc:postgresql://192.168.1.7
3:5432/base1", "postgres", "123");
            System.out.println("Conectado");
        }catch (Exception er){
            System.err.println("Error en Conexion"+ er.toString());
        }
        return  conexion;
    }
    //Creamos la funcion para Cerrar la Conexion
    protected  void cerrar_conexion(Connection con)throws  Exception{
        con.close();
    }
}
```

Por último, la función *limpiar* sirve únicamente para limpiar los campos cuando se presione el botón “Limpiar”.

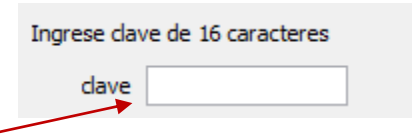
```
public void limpiar(View v){
        txtClave.setText("");
        txtMensajeCifrado.setText("");
        txtMensajeDescifrado.setText("");

    }
```

ANEXO B. MANUAL DE USUARIO

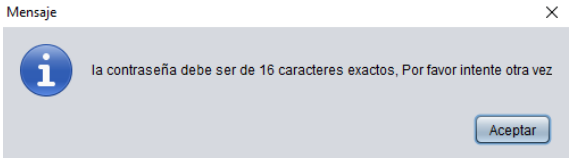
Aplicación en Java

Paso 1. Ingrese una clave de 16 caracteres exactos en el cuadro de texto que pertenece a la “clave”.

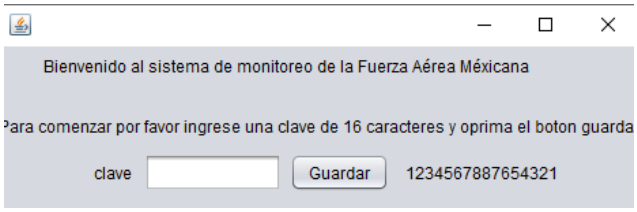


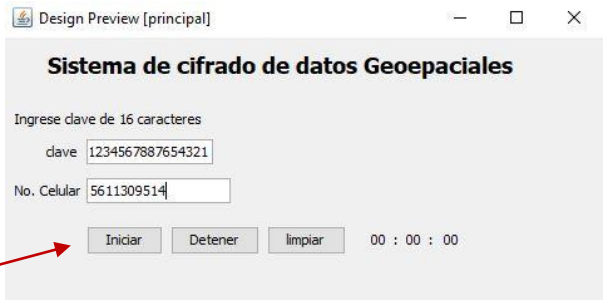
Paso 2. Ingrese algún número telefónico.

Si la clave no tiene los 16 caracteres exactos, el sistema mostrará el siguiente mensaje. En este caso, por favor de click en “aceptar” y repita el paso 1 y 2.



La clave ingresada se mostrará en la parte superior derecha, como se muestra a continuación.





Paso 3. Al ingresar la clave y el número, le damos en el botón iniciar

Al pulsar el botón iniciar se podrá observar que el reloj ubicado en el lado derecho comienza a funcionar.

Botón detener: sirve para detener el proceso.

Botón limpiar: sirve para limpiar las cajas de texto, así como poner en 0 el reloj.

Aplicación móvil