UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERIA
CARRERA TELECOMUNICACIONES

Sistema para la predicción de variables ambientales (Temperatura, humedad y velocidad del aire) basado en el modelo ARIMA para el proyecto (UNACH-CODESAN-FAO-MAATE)

Trabajo de Titulación para optar al título de:
INGINIERO EN TELECOMUNICACIONES

Autor:
López Maya, Carlos Josué

Tutor:
MgSc. Jinez Tapia, José Luis

Riobamba, Ecuador. 2025

DECLARATORIA DE AUTORÍA

Yo, **CARLOS JOSUÉ LÓPEZ MAYA,** con cédula de ciudadanía **1804345161,** autor del trabajo de investigación titulado: **Sistema para la predicción de variables ambientales (Temperatura, humedad y velocidad del aire) basado en el modelo ARIMA para el proyecto (UNACH-CODESAN-FAO-MAATE),** certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, 4 de agosto de 2025.

Carlos Josué López Maya
C.I: 180434516-1

DICTAMEN FAVORABLE DEL PROFESOR TUTOR

Quien suscribe, **José Luis Jinez Tapia** catedrático adscrito a la Facultad de Ingeniería, por medio del presente documento certifico haber asesorado y revisado el desarrollo del trabajo de investigación titulado: **Sistema para la predicción de variables ambientales (Temperatura, humedad y velocidad del aire) basado en el modelo ARIMA para el proyecto (UNACH-CODESAN-FAO-MAATE)**, bajo la autoría de Carlos Josué López Maya del estudiante; por lo que se autoriza ejecutar los trámites legales para su sustentación.

Es todo cuanto informar en honor a la verdad; en Riobamba, a los 29 días del mes de julio de 2025

Firmado electrónicamente por:
JOSE LUIS JINEZ TAPIA
Validar únicamente con FirmaEC

MgSc. José Luis Jinez Tapia
C.I: 0602899007

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación **Sistema para la predicción de variables ambientales (Temperatura, humedad y velocidad del aire) basado en el modelo ARIMA para el proyecto (UNACH-CODESAN-FAO-MAATE),** presentado por **Carlos Josué López Maya**, con cédula de identidad número 180434516-1, bajo la tutoría de MgSc. José Luis Jinez Tapia; certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba 4 de agosto del 2025.

Ing. Juan Carlos Cepeda, PhD.
PRESIDENTE DEL TRIBUNAL DE GRADO

Ing. Luis Tello, PhD.
MIEMBRO DEL TRIBUNAL DE GRADO

Ing. Alejandra Pozo, Mgs.
MIEMBRO DEL TRIBUNAL DE GRADO

C E R T I F I C A C I Ó N

Que, **LÓPEZ MAYA CARLOS JOSUÉ** con CC: **1804345161**, estudiante de la Carrera de **TELECOMUNICACIONES**, Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado "**Sistema para la predicción de variables ambientales (Temperatura, humedad y velocidad del aire) basado en el modelo ARIMA para el proyecto (UNACH-CODESAN-FAO-MAATE)**", cumple con el 2%, de acuerdo al reporte del sistema Anti plagio **COMPILATIO**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 30 de julio de 2025

Firmado electrónicamente por:
JOSE LUIS JINEZ TAPIA
Validar únicamente con FirmaEC

MgSc. José Luis Jinez Tapia
TUTOR

DEDICATORIA

Con todo mi cariño y gratitud, dedico este trabajo a las personas que han sido mi fuerza y compañía en este camino.

A mi madre, Betty, por su amor incondicional, por ser mi ejemplo de lucha y entrega constante. Su apoyo ha sido el pilar sobre el que he construido cada paso.

A Napoleón, compañero de vida de mi madre, por su respaldo firme y silencioso, siempre presente cuando más lo he necesitado.

A mi hermanita Nerea, por su ternura, su alegría y por llenar mis días de luz. Su compañía inocente y sincera ha sido un aliento especial durante este proceso.

A mis tíos Mauricio, Fernando y Silvia, por su apoyo sincero, por creer en mí y brindarme su ayuda con generosidad y afecto.

A mis amigos de la universidad, con quienes compartí risas, desvelos y aprendizajes. Aunque hoy me toque cerrar este capítulo primero, espero de corazón que cada uno de ustedes también logre cumplir su meta. Este logro también va por ustedes y para mi enamorada Flor, quien me acompañó con amor y paciencia en cada paso de este camino.

A todos ustedes, les entrego con el corazón este logro que también es suyo.

Carlos Josué López Maya.

AGRADECIMIENTO

Agradezco, en primer lugar, a Dios, por darme la vida, la fuerza y la claridad para avanzar incluso en los momentos más difíciles.

A mi madre, Betty, el corazón de este logro. Por su amor inagotable, por cada sacrificio silencioso, por creer en mí incluso cuando yo dudaba. Su entrega, su lucha diaria y su fe constante han sido mi mayor motivación. Este esfuerzo es también suyo, porque sin su mano firme y su abrazo en los días grises, este camino habría sido más duro.

A Napoleón, por su apoyo constante y silencioso que ha sido fundamental a lo largo de este proceso. A mis tíos Mauricio, Fernando y Silvia, por sus palabras de ánimo, su apoyo y por estar siempre presentes cuando más lo necesité.

A mi querida abuela Martha, por su cariño y por ser ese lazo de sabiduría y ternura que siempre me impulsa a seguir adelante.

A la Universidad Nacional de Chimborazo, por abrirme las puertas del conocimiento y brindarme las herramientas necesarias para mi formación académica y personal.

Y de manera muy especial, a mi tutor MgSc. José Luis Jinez Tapia, por su acompañamiento, orientación y paciencia a lo largo del desarrollo de este trabajo. Su guía fue clave para convertir esta investigación en una realidad.

A todos, gracias de corazón.

ÍNDICE GENERAL

ÍNDICE DE TABLAS.

ÍNDICE DE FIGURAS

RESUMEN

En este trabajo de titulación se analiza el funcionamiento del modelo de predicción ante tres variables climáticas (temperatura, humedad y velocidad del viento), como parte del proyecto UNACH-CODESAN-FAO-MAATE desarrollado en la Universidad Nacional de Chimborazo. Donde Ecuador enfrenta actualmente desafíos importantes derivados de la variabilidad climática, lo que ha incrementado la frecuencia e intensidad de eventos extremos como sequías prolongadas e inundaciones. Esta situación afecta directamente sectores como la agricultura, especialmente en la provincia de Chimborazo, donde el cambio en variables como la temperatura y la humedad influye de forma crítica en el rendimiento de los cultivos. Frente a este panorama, se hace necesaria la implementación de herramientas tecnológicas que permitan un monitoreo más preciso y la predicción anticipada de condiciones adversas.

Mediante el uso del modelo estadístico ARIMA, este estudio evalúa la precisión de las predicciones comparando los valores estimados con los datos reales obtenidos por una estación meteorológica. Para su desarrollo, se utilizó un servidor con sistema operativo Ubuntu que aloja todo el sistema, y se empleó el lenguaje de programación Python para desarrollar los scripts responsables de la recepción de datos, la predicción y el desarrollo de una página web que permite visualizar tanto los datos actuales como las predicciones semanales. Los resultados indican que la implementación de este modelo de predicción puede ser una herramienta eficaz para fortalecer la gestión del riesgo, anticiparse a eventos climáticos extremos, y mejorar la planificación agrícola mediante decisiones informadas basadas en datos. Las conclusiones demuestran que el uso de modelos estadísticos como ARIMA representa una alternativa viable para fortalecer los sistemas de monitoreo climático locales.

Palabras claves: Predicción climática, modelo ARIMA, precisión del modelo, Python, estación meteorológica, planificación de contingencias.

ABSTRACT

This thesis project analyzes the performance of a prediction model applied to three climatic variables—temperature, humidity, and wind speed—as part of the UNACH-CODESAN-FAO-MAATE initiative developed at the National University of Chimborazo. Ecuador currently faces major challenges due to climate variability, which has increased the frequency and intensity of extreme weather events such as prolonged droughts and floods. These conditions directly impact sectors such as agriculture—particularly in the province of Chimborazo—where changes in variables like temperature and humidity critically affect crop yields. In this context, the implementation of technological tools that enable more accurate monitoring and early forecasting of adverse weather conditions becomes essential. This study employs the ARIMA statistical model to evaluate forecasting accuracy by comparing predicted values with actual data collected from a weather station. The system was implemented on a server running the Ubuntu operating system, and Python was used to develop scripts for data reception, prediction, and a web interface that displays both real-time data and weekly forecasts. The results show that this prediction model can be an effective tool for strengthening risk management, anticipating extreme weather events, and enhancing agricultural planning through data-driven decision-making. The conclusions demonstrate that the use of statistical models such as ARIMA represents a viable approach to improving local climate monitoring systems.

Keywords: Climate prediction, ARIMA model, model accuracy, Python, weather station, contingency planning.

Firmado electrónicamente por:
LOURDES DEL ROCIO
QUINATA ENCARNACION
Validar únicamente con FirmaEC

Reviewed by:
Mg. Lourdes del Rocío Quinata Encarnación
ENGLISH PROFESSOR
C.C 1803476215

CAPÍTULO I

1. INTRODUCCIÓN

La predicción climática ha evolucionado con la implementación de modelos estadísticos que permiten analizar datos históricos y detectar patrones en las condiciones atmosféricas. Entre estos, el modelo Autorregresivo Integrado de Media Móvil (ARIMA) se ha consolidado como una herramienta eficaz para pronosticar variables meteorológicas como la temperatura, la humedad y la velocidad del aire. A diferencia de los modelos numéricos tradicionales, que dependen de ecuaciones físicas para describir el comportamiento atmosférico, ARIMA basa sus predicciones en el análisis de series temporales, lo que permite modelar tendencias y variaciones estacionales de manera precisa [1].

En Riobamba, ciudad situada en la provincia de Chimborazo, la variabilidad climática representa un desafío significativo, especialmente para sectores como la agricultura, el abastecimiento de agua y la gestión de desastres. Debido a su geografía y altitud, las condiciones meteorológicas de la región son altamente variables, lo que dificulta la precisión de los pronósticos. Aunque existen datos históricos y en tiempo real sobre el clima, los modelos convencionales presentan limitaciones en su capacidad para ofrecer predicciones localizadas con alta fiabilidad. En este contexto, el uso de modelos estadísticos como ARIMA ofrece una alternativa viable para mejorar la estimación de variables ambientales y optimizar la toma de decisiones en diversos sectores.
El desarrollo de estaciones meteorológicas autónomas ha permitido mejorar la recolección y análisis de datos climáticos en tiempo real. Estas estaciones, equipadas con sensores, facilitan la implementación de modelos como ARIMA para generar pronósticos más ajustados a la realidad local. Investigaciones recientes han demostrado que el uso de series temporales en la predicción climática puede mejorar la precisión de los pronósticos en comparación con métodos tradicionales, proporcionando información valiosa para la planificación y mitigación de riesgos [2].

Este estudio tiene como objetivo evaluar la precisión de un modelo ARIMA en la predicción de temperatura, humedad y velocidad del aire en una estación meteorológica autónoma. Se realizarán pruebas de campo en la Universidad Nacional de Chimborazo, comparando las predicciones del modelo con datos meteorológicos reales en períodos semanales. A partir de esta evaluación, se determinará si el modelo alcanza un umbral de precisión adecuado para su aplicación en estudios climáticos y en el sector agrícola.
La principal contribución de este estudio es la estimación de variables ambientales clave, como la temperatura, la humedad y la velocidad del aire, utilizando el modelo ARIMA. Estos resultados serán útiles para aplicaciones en investigación climática y en la toma de decisiones en el sector agrícola.

1.1 Planteamiento del problema

Ecuador enfrenta desafíos críticos derivados de la variabilidad y los cambios climáticos, lo que ha incrementado la frecuencia y severidad de fenómenos extremos como sequías prolongadas, lluvias intensas y deslizamientos de tierra. Según el Instituto Nacional de Meteorología e Hidrología (INAMHI), eventos como el fenómeno de El Niño han causado inundaciones devastadoras en las regiones costera y andina, afectando tanto la infraestructura como la seguridad de las comunidades. Por otro lado, las sequías recurrentes han generado crisis hídricas que impactan negativamente la agricultura, el suministro de agua potable y la generación hidroeléctrica [3].

Ante esta situación, la implementación de un sistema de alerta temprana (SAT) para la predicción de inundaciones, sequías y heladas es una estrategia clave para mitigar sus impactos y mejorar la resiliencia de las comunidades vulnerables.

En la provincia de Chimborazo, y específicamente en Riobamba, la actividad agrícola es una de las principales fuentes de sustento para la población rural. Sin embargo, la variabilidad climática representa un obstáculo significativo para la producción agrícola, ya que factores como la temperatura, la humedad y la velocidad del aire influyen directamente en el desarrollo y rendimiento de los cultivos. La temperatura es una variable determinante en el crecimiento de las plantas, ya que desviaciones fuera de los rangos óptimos pueden afectar su calidad y provocar pérdidas económicas. De manera similar, la humedad es esencial para el desarrollo de los cultivos, ya que un déficit o exceso de esta puede generar condiciones adversas como sequías o proliferación de plagas y enfermedades. Además, la velocidad del aire juega un papel crucial en la evapotranspiración y en la dispersión de plagas, lo que puede influir en la productividad agrícola y en la estabilidad de las cosechas.

A pesar de estos desafíos, existe una necesidad urgente de mejorar las capacidades de monitoreo y predicción meteorológica en la región. La estimación precisa de variables climáticas como la temperatura, la humedad y la velocidad del aire permitiría a los agricultores optimizar la gestión de los cultivos y reducir el impacto de fenómenos climáticos adversos. No obstante, uno de los principales problemas es la falta de acceso a datos climáticos locales en tiempo real y la ausencia de modelos predictivos adecuados que permitan anticipar variaciones en estas variables. La implementación de un SAT basado en inteligencia artificial y modelos estadísticos avanzados permitiría generar alertas tempranas y mejorar la toma de decisiones en el sector agrícola y en la gestión del riesgo de desastres.

Para abordar esta problemática, el uso de modelos estadísticos como el Autorregresivo Integrado de Media Móvil (ARIMA) se presenta como una alternativa viable. Este modelo, basado en series temporales, permite identificar patrones en los datos históricos y generar predicciones más precisas sobre la evolución de las variables climáticas. Integrar estos modelos en un SAT facilitaría la identificación temprana de condiciones meteorológicas extremas, permitiendo la implementación de medidas preventivas para reducir pérdidas económicas y mejorar la seguridad alimentaria en la región.

1.2 Justificación

La implementación de una estación meteorológica equipada con sensores y un modelo estadístico responde a la necesidad urgente de contar con herramientas tecnológicas para el monitoreo y la predicción de precipitaciones. En el contexto de la creciente variabilidad climática y los efectos del cambio climático, fenómenos como sequías prolongadas, lluvias intensas e inundaciones han aumentado en frecuencia y severidad, afectando sectores clave como la agricultura, la gestión hídrica y la generación de energía [3].

Este proyecto tiene como objetivo evaluar la precisión del modelo Autorregresivo Integrado de Media Móvil (ARIMA) en la predicción de variables meteorológicas. ARIMA, ampliamente utilizado en el análisis de series temporales, permite identificar patrones en los datos históricos y generar pronósticos más precisos sobre la evolución del clima. La implementación de este modelo contribuirá a mejorar la planificación agrícola, optimizar la gestión de los recursos hídricos y fortalecer la capacidad de respuesta ante eventos climáticos adversos.

Además, este proyecto tiene un impacto significativo en el ámbito académico y tecnológico de la Facultad de Ingeniería de la UNACH, fomentando la investigación aplicada y el desarrollo de metodologías innovadoras para el análisis de datos climáticos. Al integrar herramientas de modelado estadístico en la predicción meteorológica, se fortalece la capacidad de anticipación y toma de decisiones informadas en sectores estratégicos.

1.3 Objetivos

1.3.1 Objetivo General

- Implementar un sistema para la predicción de variables ambientales (Temperatura, humedad y velocidad del aire) basado en el modelo ARIMA para el proyecto (UNACH-CODESAN-FAO-MAATE).

1.3.2 Objetivos específicos

- Analizar las variables climáticas clave, como la temperatura, humedad y velocidad del aire, para comprender su interacción e influencia en la predicción de variables ambientales.
- Entrenar el modelo de estimación ARIMA mediante la relación de datos temperatura, humedad y velocidad del aire.
- Integrar el modelo de predicción a la estación meteorológica del proyecto (UNACH-CODESAN-FAO-MAATE).
- Evaluar la precisión del modelo de estimación mediante pruebas de campo en la Universidad Nacional de Chimborazo, comparando sus resultados con los datos reales en períodos semanales para determinar si cumple con un umbral de precisión adecuado.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 Estado del arte

En los últimos años, el análisis estadístico ha demostrado ser una herramienta eficaz para la gestión de recursos naturales y la planificación ante fenómenos climáticos extremos. En el contexto de Ecuador, un estudio realizado por la Escuela Superior Politécnica de Chimborazo (ESPOCH) se centró en la predicción de precipitaciones mediante el análisis de datos históricos climáticos. En este estudio, se emplearon técnicas estadísticas y de aprendizaje automático para establecer patrones en las precipitaciones y mejorar la precisión de los pronósticos, lo que podría resultar de gran utilidad para la agricultura y la planificación de recursos hídricos [4].

Otro estudio, llevado a cabo por la Universidad Politécnica Salesiana, se centró en el modelamiento de la temperatura y la humedad relativa, para el cumplimiento de las buenas prácticas de manufactura de los medicamentos. Los investigadores emplearon modelos matemáticos como el modelo ARIMA, el cual permitió realizar un ajuste en estas variables para la perfecta conservación de los medicamentos. A través de este enfoque, el estudio no solo mejoró la precisión de tener un buen plan de contingencia, sino que también proporcionó información valiosa a través de la predicción con el modelo ARIMA [5].

Asimismo, en la Universidad de Las Palmas de Gran Canaria, se desarrolló un modelo predictivo utilizando variables meteorológicas, para cuantificar las precipitaciones de lluvia, utilizando datos meteorológicos históricos y modelos matemáticos. Este estudio no solo se enfocó en la predicción de precipitaciones, sino también en su impacto en las telecomunicaciones terrestres y satelitales. Los resultados obtenidos demostraron que los algoritmos de aprendizaje automático pueden ser eficaces en la predicción y también en la mejora de la infraestructura de comunicaciones en zonas afectadas por precipitaciones extremas, lo que resalta la versatilidad de estos modelos [6].

En un esfuerzo similar, la investigación realizada en la Universidad Nacional de Chimborazo se dedicó a desarrollar un sistema predictivo para precipitaciones, tomando en cuenta las variables climáticas clave, como la temperatura y la humedad, y utilizando algoritmos de aprendizaje automático. Este trabajo subraya la importancia de contar con modelos precisos para la predicción de precipitaciones, los cuales podrían ser cruciales para la toma de decisiones en la gestión de recursos hídricos, la planificación agrícola y la reducción de riesgos ante desastres naturales. Además, el uso de datos históricos climáticos permitió una evaluación más robusta y realista de las predicciones [7].

Por último, se desarrolló un proyecto de investigación con el objetivo de anticiparse a eventos climáticos adversos, como las heladas y el exceso de radiación solar al mediodía. Para ello, se analizaron los registros de temperatura obtenidos mediante una estación

meteorológica, con el fin de identificar su comportamiento periódico a lo largo del tiempo. Este análisis permitió aplicar el modelo ARIMA para generar pronósticos semanales y, en función de ellos, implementar oportunamente medidas de protección en este caso en el ámbito agrícola [8]

La lectura realizada ha permitido identificar los avances más significativos en el uso de modelos estadísticos para la predicción de precipitaciones, proporcionando una base sólida para esta investigación. Los estudios revisados destacan la efectividad del modelo Autorregresivo Integrado de Media Móvil (ARIMA) en la predicción de variables meteorológicas a partir de datos históricos, lo que ha permitido una notable mejora a la precisión de los pronósticos climáticos. Este enfoque ha demostrado ser clave en la gestión de recursos naturales, la planificación agrícola y la mitigación de riesgos ante desastres naturales. Los resultados obtenidos en esta investigación servirán como referencia para optimizar la implementación del modelo ARIMA en este proyecto.

2.2 Fundamentación teórica

Las estaciones meteorológicas avanzadas constituyen un progreso significativo en el monitoreo y la predicción climática, al integrar tecnologías como el Internet de las Cosas (IoT) con modelos estadísticos para el procesamiento de datos. El modelo ARIMA se destaca por su capacidad para analizar series temporales de datos históricos e identificando patrones y tendencias para generar predicciones precisas de variables ambientales como temperatura, humedad y velocidad del aire. Estas herramientas no solo mejoran la fiabilidad en la predicción de condiciones climáticas, sino que también permiten ajustar los modelos a las características específicas de cada región así optimizando su desempeño. Hay diferentes tipos principales de estaciones meteorológicas clasificados por función, por su ubicación y su tecnología.

2.2.1 Por su función

- **Estaciones manuales** que son operadas por personas que registran datos periódicamente. Son más comunes en zonas rurales o donde no hay acceso a tecnología avanzada [9].
- **Estaciones automáticas (EMA)** que usan sensores electrónicos para recopilar y transmitir datos automáticamente muchas veces en tiempo real. Un ejemplo práctico de esta integración tecnológica es el uso de módulos como el ESP8266 en el diseño de estaciones meteorológicas portátiles. Estos dispositivos permiten recopilar datos ambientales en tiempo real y transmitirlos eficientemente a través de redes inalámbricas así mejorando el acceso a información clave para la toma de decisiones críticas [10]. De igual forma en plataformas como la Raspberry Pi se han consolidado como soluciones de bajo costo y alta efectividad en regiones remotas, donde el monitoreo continuo es importante [11].

2.2.2 Por su ubicación

- **Estaciones terrestres** son las que se ubican en tierra firme y son las más comunes. Su ubicación debe seguir estándares internacionales como distancia de obstáculos, altura de sensores, etc. [12].
- **Estaciones marinas o boyas meteorológicas** son las que flotan en el océano y miden condiciones marítimas y atmosféricas. Vitales para la navegación y monitoreo del clima global.
- **Estaciones aéreas (aeronaves o globos-sonda)** son las que miden condiciones en altura y son esenciales para estudios atmosféricos a diferentes niveles.
- **Estaciones satelitales** en si no son estaciones en sí, pero los satélites meteorológicos recopilan información global desde el espacio, complementando a las estaciones en superficie.

2.2.3 Por su tecnología

- **Estaciones meteorológicas sinópticas** son diseñadas para enviar datos estandarizados a redes meteorológicas internacionales, como las de la OMM [12].
- **Estaciones agroclimáticas o agrometeorológicas** son especializadas en recopilar datos para la agricultura como la temperatura del suelo, humedad, evapotranspiraciones relacionadas a la agricultura [13].
- **Estaciones hidrometeorológicas** se combinan variables meteorológicas con datos hidrológicos (como niveles de ríos) y se usan para alertas de inundaciones.
- **Estaciones urbanas** se enfocadas en el monitoreo del microclima urbano como islas de calor, contaminación del aire y así relacionadas al entorno urbano [14]

2.3 Climatología

El análisis de variables climáticas como la temperatura, la humedad y la velocidad del aire es fundamental para comprender el comportamiento atmosférico y mejorar la predicción de fenómenos ambientales. Estas variables no actúan de forma aislada, sino que interactúan constantemente: por ejemplo, el ascenso de temperatura puede reducir la humedad relativa si no hay aporte de vapor de agua, mientras que la circulación del viento influye en la distribución térmica y la dispersión de masas de aire húmedo o seco. Comprender estas relaciones permite interpretar y mejor la dinámica del clima local, para aplicar modelos predictivos más precisos [15].

2.3.1 Sensores ultrasónicos

Los sensores ultrasónicos son dispositivos que utilizan ondas de sonido de alta frecuencia para medir la distancia entre el sensor y un objeto. En meteorología, se emplean principalmente para medir la velocidad y dirección del viento, ya que permiten calcular el tiempo que tarda una onda sonora en viajar entre dos puntos. Al no tener partes móviles, ofrecen mayor durabilidad y precisión frente a condiciones adversas como el polvo o la lluvia, en comparación con anemómetros mecánicos.

Figura 1: Sensor ultrasónico

Fuente: [16]

2.3.2 Radiómetro ultravioleta UV

El radiómetro UV es un sensor que mide la intensidad de la radiación ultravioleta proveniente del sol. Esta radiación, aunque invisible al ojo humano, tiene efectos significativos en la salud, la agricultura y el medio ambiente. El sensor permite evaluar los niveles de exposición UV, proporcionando datos útiles para alertas sanitarias, estudios de ozono o control de cultivos sensibles a la radiación solar.

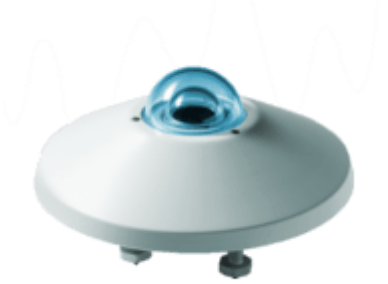


Figura 2: Radiómetro ultravioleta UV

Fuente: [17]

2.3.3 Temperatura

El sensor de temperatura mide la temperatura del aire ambiente. Generalmente utiliza termistores o sensores digitales calibrados que responden a los cambios térmicos con alta precisión. Es una variable fundamental en el monitoreo meteorológico, ya que afecta múltiples fenómenos como la evaporación, la formación de nubes y las heladas. Su medición precisa es clave para cualquier sistema de pronóstico.

Figura 3: Sensor de temperatura

Fuente: [18]

2.3.4 Pluviómetro

El pluviómetro es un instrumento diseñado para medir la cantidad de precipitación caída en un período determinado. Los modelos digitales suelen utilizar un sistema de cubeta basculante que registra electrónicamente cada incremento de lluvia. Esta variable es esencial para el estudio hidrológico, el monitoreo de sequías, el diseño agrícola y la predicción de eventos extremos como inundaciones.

Figura 4: Pluviómetro

Fuente: [19]

2.3.5 Humedad

El sensor de humedad, también conocido como higrómetro, mide la cantidad de vapor de agua presente en el aire, generalmente expresado como humedad relativa. Esta medición es crucial para la predicción del punto de rocío, la formación de nubes y la evaluación del confort térmico. Además, la humedad influye directamente en procesos agrícolas y en la propagación de enfermedades respiratorias.

Figura 5: Sensor de Humedad

Fuente: [19]

2.3.6 Veleta

La veleta es un sensor que detecta la dirección del viento. Generalmente consiste en una paleta que se alinea con el viento, y un sistema interno que convierte esa posición en una señal digital. Este dato, combinado con la velocidad del viento, permite analizar patrones de circulación atmosférica, dispersión de contaminantes y orientar adecuadamente instalaciones como paneles solares o turbinas eólicas.

Figura 6: Veleta

Fuente: [20]

2.3.7 Radiancia

El sensor de radiancia mide la cantidad total de energía radiante emitida o reflejada por una superficie en una longitud de onda determinada. En meteorología, se utiliza para evaluar la energía solar incidente y reflejada, permitiendo analizar el balance energético de la atmósfera y la superficie terrestre. También es útil en estudios climáticos, eficiencia energética y modelado de la evapotranspiración.

Figura 7: Sensor de radiancia

Fuente: [21]

2.4 Modelos de predicción de variables climáticas

Son herramientas matemáticas y estadísticas utilizadas para anticipar el comportamiento del clima mediante datos históricos observados. Estos modelos permiten estimar variables como la temperatura, la precipitación, la humedad o la velocidad del viento entre otras. Su uso es clave para la planificación agrícola, la gestión del agua, la prevención de desastres naturales y la toma de decisiones en políticas ambientales. Dependiendo de su complejidad estas pueden ir desde modelos estadísticos simples hasta simulaciones numéricas basadas en la física de la atmósfera.

2.4.1 Modelo de redes neuronales (RNA)

Estos modelos son inspirados en el funcionamiento del cerebro humano, donde aprenden patrones complejos no lineales a partir de grandes volúmenes de datos climáticos. Son especialmente útiles cuando las relaciones entre variables no son evidentes o son demasiado complejas para los modelos tradicionales. Se usan mucho en predicción de temperatura, lluvia, humedad y eventos extremos [22].

2.4.2 Modelo Autorregresivo ARIMA y SARIMA

Los modelos de autorregresivo (AR) es un modelo estadístico que utiliza series temporales para describir el comportamiento actual de una variable en función de sus valores pasados. En un modelo AR de orden p, llamado como AR (p), el valor actual y_t depende de los valores $y_t - 1, y_t - 2, \dots, y_t - p$, más un término de error aleatorio [23].

Estos modelos se emplean para realizar pronósticos basados en observaciones previas cuyos valores se conocen completamente y que están ordenadas cronológicamente. Es decir, la variable "regresa sobre sí misma", con el valor en el tiempo t que sería el resultado de sus propios valores anteriores $t - 1, t - 1, \dots, t - p$.

Los procesos autorregresivos presentan función de autocorrelación parcial ACFP con un numero finito de valores distintos de cero y los demás que son nulos. Un valor se considera casi cero cuando su modulo es inferior a $2/t$. Los softwares constituyen la franja -$2/t$; $2/t$ y detectan los valores de la ACFP que caen fuera de ella. Los procesos de medias móviles presentan función de autocorrelación con un numero finito de valores distintos de cero. Un proceso MAQ donde q tiene los primeros términos de la función de autocorrelación distintos de cero y los demás son nulos. Las dos propiedades mencionadas son muy importantes con vistas a la identificación de un proceso mediante la autocorrelación y autocorrelación parcial [24]. A continuación, se detallan las características y sus aplicaciones:

- **Características:** Los modelos autorregresivos se caracterizan por utilizar la propia historia de la variable como base para realizar sus predicciones, es decir, regresan sobre sí mismos. Este tipo de modelos se denota como AR(p), donde p representa el orden del modelo y corresponde al número de periodos anteriores que se consideran para estimar el valor futuro de la variable. Una de sus principales particularidades es que su especificación se fundamenta

principalmente en los datos observados, más que en teorías subyacentes, lo que les permite capturar de forma efectiva los mecanismos dinámicos del proceso siempre que se cumpla la condición de estacionalidad [25].

- **Aplicación:** La aplicación de los modelos autorregresivos se basa en su capacidad para predecir valores futuros a partir de datos pasados, lo que los convierte en herramientas fundamentales para el análisis de series temporales en diversas disciplinas. En el ámbito económico y financiero, se utilizan para estimar variables como la inflación, el tipo de cambio o los precios de activos, aprovechando la dependencia temporal inherente a estas series. En meteorología, permiten anticipar el comportamiento de variables climáticas como la temperatura, la humedad o la velocidad del viento, a partir de registros históricos[8]. Asimismo, en la ingeniería de señales se aplican para modelar ruidos y patrones repetitivos en sistemas dinámicos contribuyendo al procesamiento y análisis de señales con componentes estocásticos.

Ahora se va a analizar dos modelos que están dentro del modelo autorregresivo que son el modelo ARIMA y SARIMA.

Modelo ARIMA

El modelo Autoregressive Integrated Moving Average (ARIMA) es una técnica ampliamente utilizada en el análisis de series temporales debido a su capacidad para modelar datos dependientes del tiempo. Permite describir un valor como una función lineal de datos anteriores y errores debidos al azar, donde incluye un componente cíclico o estacional.

La metodología de Box y Jenkins se resume en cuatro fases. Primero consiste en transformar la serie original para hacerla estacionaria e identificar los posibles órdenes del modelo autorregresivo y de media móvil (p y q). En la segunda fase, se estiman los parámetros del modelo mediante el método de máxima verosimilitud y se calculan los errores estándar y los residuos. La tercera fase corresponde al diagnóstico, en la que se evalúa si los residuos se comportan como ruido blanco; si no es así, el modelo se ajusta y se repiten las fases anteriores. Finalmente, en la cuarta fase, una vez validado el modelo, se procede a realizar las predicciones.

Matemáticamente se define el modelo según esta ecuación [26].

$$\emptyset p(L)(1-L)^d Y_t = \delta + \theta_q(L)a_t$$

Los modelos con periodicidad o cíclicos en el tiempo como es el caso de los parámetros climáticos, presenta la forma general ARIMA (p,d,q) y su ecuación [26].

$$\emptyset_p(L)\emptyset(L^S)(1-l)^d(1-L^S)^D Y_t = \delta + \theta_q(L)\sigma(L^S)a_t$$

Donde:

$\emptyset_p$= Coeficiente Autorregresivo

$p = Orden\ Autorregresivo$
L= Operador de Retardo
$\theta = Coeficiente\ de\ Autoregresivo\ estacionario$
s = Periodo Estacional
d = Orden de las diferencias
D = Orden de las diferencias Estacionales
δ = Media del proceso
θ_q= Coeficiente de media móvil
q = Orden de media móvil
σ= Coeficiente de media móvil estacional
a_t= Función de innovación o perturbaciones

Modelo SARIMA

El modelo SARIMA (AutoRegresivo Integrado de Media Móvil Estacional) es una extensión del modelo ARIMA que incorpora componentes estacionales esto lo hace ideal para predecir series temporales que presentan patrones repetitivos a lo largo del tiempo como las variables climáticas mensuales o anuales. Este modelo combina términos autorregresivos como es la diferenciación y de media móvil tanto en su forma regular como estacional. Esto permite capturar tendencias a largo plazo y ciclos recurrentes (por ejemplo, estaciones del año). Es ampliamente utilizado en climatología para anticipar variables como la temperatura o la precipitación que muestran un comportamiento cíclico bien definido [26].

2.5 Pronóstico

Los pronósticos se elaboran a partir de datos históricos utilizando modelos matemáticos, técnicas estadísticas y otras herramientas analíticas que permiten anticipar el comportamiento futuro de una variable. Aunque no garantizan una predicción exacta, los pronósticos son fundamentales para tomar decisiones informadas y planificar estrategias basadas en tendencias y patrones previamente observados. En este contexto, los modelos autorregresivos (AR) constituyen una metodología ampliamente utilizada en el análisis de series temporales, al permitir estimar valores futuros en función de valores pasados de la misma serie. Estos modelos se basan en la premisa de que los valores presentes dependen linealmente de una cantidad limitada de observaciones anteriores. Un modelo autorregresivo de orden p, denominado como AR(p), utiliza los últimos p valores para predecir el siguiente. La estimación de los coeficientes del modelo se realiza mediante métodos estadísticos como los mínimos cuadrados o la máxima verosimilitud. Una vez ajustado el modelo, se pueden generar pronósticos para varios periodos hacia adelante. Es importante tener en cuenta que los modelos AR requieren que la serie sea estacionaria, es decir, que su media y varianza se mantengan constantes a lo largo del tiempo. La

precisión del pronóstico dependerá tanto de la calidad de los datos utilizados como de la adecuación del modelo a la estructura real de la serie temporal analizada.

2.6 VPS

Un VPS (Servidor Privado Virtual, por sus siglas en inglés) es una máquina virtual que se ejecuta en un servidor físico compartido, pero que funciona de manera independiente con su propio sistema operativo como Windows o Linux. Tiene recursos asignados como CPU, memoria y almacenamiento controlado por parte del usuario. Lo que permite tener un entorno aislado y personalizable para alojar aplicaciones, sitios web, bases de datos o servicios, con mayor flexibilidad y seguridad que un hosting compartido tradicional.

Figura 8: VPS

Fuente:[27]

2.7 Lenguajes

2.7.1 Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, diseñado para ser fácil de leer y escribir gracias a su sintaxis clara y estructurada. Fue creado por Guido van Rossum y lanzado en 1991, y desde entonces se ha convertido en una herramienta ampliamente utilizada en diversos campos como desarrollo web, automatización, análisis de datos, inteligencia artificial y ciencia. Su gran comunidad y extenso ecosistema de bibliotecas lo hacen ideal para proyectos científicos y técnicos, como el procesamiento de datos meteorológicos y la implementación de modelos de predicción [28].

2.7.2 Flask

Flask es un microframework web escrito en Python que permite desarrollar aplicaciones web de manera rápida y sencilla. Fue diseñado para ser ligero, flexible y modular. Flask es ampliamente utilizado en aplicaciones donde se requiere construir interfaces o servicios web, como en sistemas de monitoreo en tiempo real o servidores API REST [29].

CAPÍTULO III

3. METODOLOGÍA

3.1 Tipo de Investigación.

La presente investigación tiene como objetivo desarrollar un sistema de predicción para tres variables climáticas (temperatura, humedad y velocidad del aire) para una estación meteorológica. El enfoque principal consiste en la implementación del modelo Autorregresivo Integrado de Media Móvil (ARIMA) para el análisis de la predicción temporal de las variables climáticas. Este estudio tiene un carácter experimental, ya que busca validar la precisión del modelo mediante pruebas controladas y el análisis de datos recolectados en un entorno real.

Un aspecto clave de esta investigación es el análisis de variables climáticas fundamentales, como la temperatura, la humedad y la velocidad del aire, con el fin de comprender su interacción y su impacto en el entorno. Estas variables influyen directamente en la dinámica atmosférica, por lo que su monitoreo y modelado preciso son esenciales para mejorar la fiabilidad de los pronósticos. Los datos climáticos provendrán de dos fuentes principales: información en tiempo real capturada por la estación meteorológica y registros históricos suministrados por la API Mateum, que incluyen variables como temperatura, humedad, presión atmosférica, dirección y velocidad del viento, punto de rocío y precipitación acumulada [30]. Para la implementación y evaluación del modelo, se utilizará el lenguaje de programación Python, una herramienta clave en el análisis de series temporales y modelado estadístico.

El desempeño del modelo ARIMA se evaluará mediante pruebas de campo en la Universidad Nacional de Chimborazo, comparando los resultados de las predicciones con los datos reales obtenidos en períodos semanales. Este análisis permitirá determinar si el modelo cumple con un umbral de precisión adecuado para su aplicación en la predicción de precipitaciones.

3.2 Método de investigación

3.2.1 Cualitativo

En el presente proyecto de titulación se realizó una revisión exhaustiva de documentos técnicos y científicos con el fin de identificar patrones climáticos relevantes y comprender las características principales de los datos meteorológicos utilizados en el entrenamiento del modelo estadístico ARIMA. Esto permitió analizar de manera detallada las variables más importantes, como temperatura, humedad y velocidad del aire, asegurando que se configure adecuadamente el modelo en función de los objetivos planteados.

3.2.2 Cuantitativo

El método cuantitativo se empleó para el diseño mediante la recolección de datos climáticos y la aplicación del modelo ARIMA. Se utilizó la base de datos histórica

proporcionada por Mateum junto con los datos recolectados por la estación meteorológica, con el fin de predecir el comportamiento semanal de variables como temperatura, humedad y velocidad del viento. Permitiendo así realizar la comparación de los resultados con los datos reales para evaluar la precisión del modelo.

3.3 Técnicas de recolección de datos

Los instrumentos empleados para la recolección de datos fueron la estación meteorológica Ecowitt GW3000 con sus sensores ambientales, que permiten medir variables como temperatura, humedad y velocidad del aire en tiempo real. Esta información se transmite automáticamente al servidor mediante una API REST, donde se almacena en archivos CSV para su posterior análisis. Además, se complementó esta recolección con una base de datos histórica proporcionada por Mateum, la cual sirvió como insumo para el entrenamiento del modelo de predicción.

3.4 Población y muestra

3.4.1 Población

La población de esta investigación estará constituida por los datos históricos de temperatura, humedad y velocidad del aire registrados. Estos datos servirán como base para entrenar y evaluar el modelo ARIMA, permitiendo analizar su desempeño en la predicción de variables ambientales. Dado que la recolección de datos es continua en el tiempo, se considera que la población tiene un carácter infinito.

3.4.2 Muestra

La muestra es el historial climático de 10 años que contiene 350.788 datos (los valores son time_iso,humidity,temperature,wind_speed) que sirve para el entrenamiento del modelo ARIMA.

3.5 Operacionalización de las variables

Tabla 1:Operacionalización de las variables

Variable Independiente	**Concepto**	**Indicadores**	**Técnicas e Instrumentación**
Variables Climáticas Temperatura, humedad y velocidad del aire.	Son los datos climáticos utilizados por el modelo para predecir temperatura, humedad y velocidad del viento.	Temperatura, humedad y velocidad del viento.	Adquisición de datos mediante sensores de temperatura (termómetro), sensores de humedad y anemómetro.
Variable dependiente	**Concepto**	**Indicadores**	**Técnicas e Instrumentación**
Predicciones del modelo ARIMA	Se refiere a la capacidad del modelo estadístico ARIMA para estimar con precisión la temperatura, humedad y velocidad del viento en un periodo determinado.	Error de predicción y precisión del modelo en porcentaje.	La precisión del modelo se evaluará mediante el Error Relativo Medio (MRE) entre los valores predichos y los reales

Fuente: Autor

3.6 Diseño de investigación

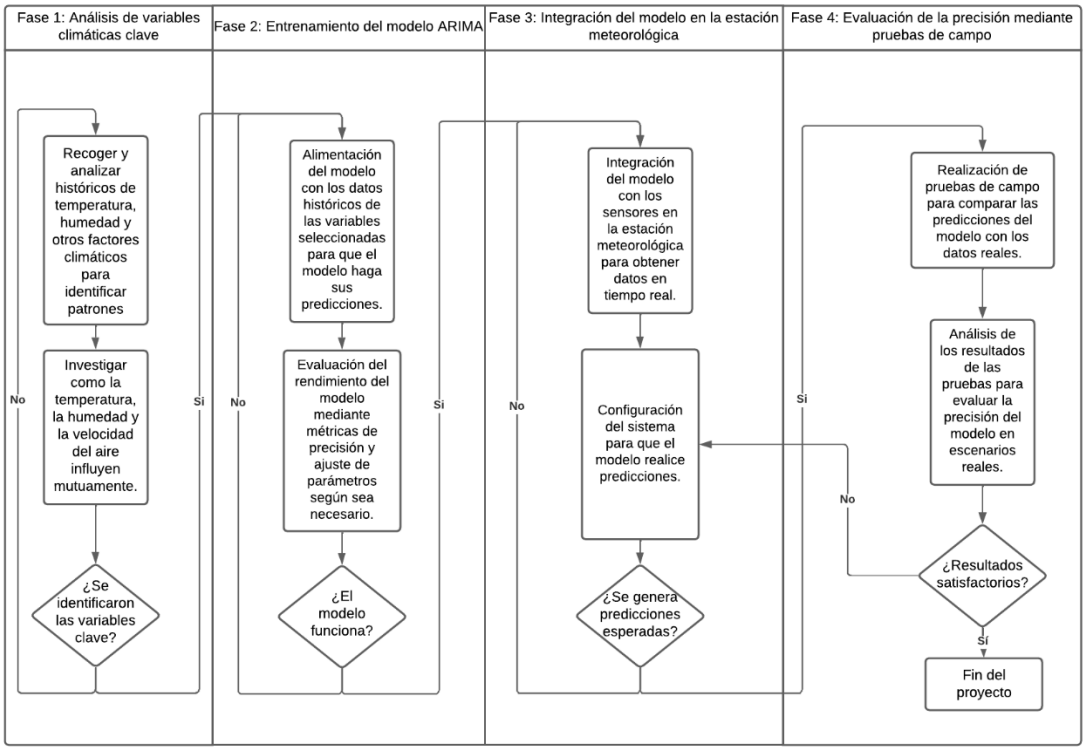


Figura 9: Diagrama de flujo del proyecto

Fuente: Autor

Para asegurar el desarrollo adecuado de este trabajo de titulación y el cumplimiento de los objetivos planteados para esta investigación se organizaron en cuatro etapas principales, descritas a continuación:

Etapa 1: Análisis de variables climáticas clave

En esta primera etapa, se llevó a cabo un análisis detallado de las variables climáticas más relevantes para la predicción ambiental: temperatura, humedad y velocidad del aire. Se estudio la interacción entre estas variables para comprender su impacto en los cambios climáticos y su relación con las condiciones meteorológicas. Se recopilaron datos históricos de estas variables con el fin de identificar patrones y tendencias para luego entrenar al modelo ARIMA.

Etapa 2: Entrenamiento del modelo ARIMA

En esta fase, se empleó el modelo estadístico ARIMA para la estimación y predicción de variables climáticas. Se ajustaron los parámetros del modelo utilizando los datos históricos de temperatura, humedad y velocidad del aire, aplicando técnicas como el análisis de autocorrelación y autocorrelación parcial (ACF y PACF), así como la diferenciación de series para garantizar la estacionariedad de los datos. El proceso se realizó mediante un script en Python que carga los datos históricos desde un archivo CSV,

identifica automáticamente la estructura de la serie temporal, ajusta un modelo ARIMA para cada variable y genera predicciones semanales. Este código utiliza la librería statsmodels para construir el modelo, y pandas para el manejo y preprocesamiento de datos. Las predicciones resultantes se almacenan en un nuevo archivo CSV para su posterior análisis y visualización. Durante esta etapa se evaluaron distintos órdenes del modelo ARIMA, con el fin de encontrar la configuración óptima que brinde las mejores predicciones posibles para cada variable climática

Etapa 3: Integración del modelo en la estación meteorológica

Una vez entrenado el modelo ARIMA, se procedió a su implementación en la estación meteorológica del proyecto (UNACH-CODESAN-FAO-MAATE). El modelo fue integrado dentro del entorno de ejecución del servidor VPS, automatizando su ejecución mediante tareas programadas (cron) que permiten generar predicciones de forma semanal sin intervención manual. Los datos actuales se reciben en tiempo real desde la estación meteorológica Ecowitt GW3000 a través de una API REST desarrollada en Flask, la cual guarda automáticamente la información en archivos CSV estructurados. El script de predicción, también desarrollado en Python, accede a estos datos, realiza el análisis correspondiente y actualiza los archivos de salida con los valores pronosticados. Además, se desarrolló una interfaz web que muestra tanto los datos en tiempo real como los resultados del modelo, permitiendo un monitoreo continuo con datos actuales. Esta fase garantiza que el sistema funcione de manera autónoma, confiable y accesible, generando pronósticos continuos que pueden respaldar la toma de decisiones ambientales.

Etapa 4: Evaluación de la precisión mediante pruebas de campo

Para validar la efectividad del modelo, se realizaron pruebas de campo en la Universidad Nacional de Chimborazo. Donde se comparó los valores predichos por ARIMA con los datos reales recopilados por la estación meteorológica en un periodo de tres semanas. La evaluación se basó en métricas estadísticas para determinar si el modelo cumple con un umbral de precisión adecuado.

3.7 Métodos de análisis, y procesamiento de datos

En el método de análisis y procesamiento de datos se desarrolló en 5 puntos que se desarrollaran a continuación:

3.7.1 Preprocesamiento de datos

Los datos antes de ser utilizados para el entrenamiento del modelo ARIMA fueron sometidos a un proceso de preprocesamiento, para garantizar la calidad y la coherencia de la información utilizada. Se aplicaron técnicas de limpieza y transformación de datos tanto a la base histórica adquirida (con 10 años de registros climáticos) como los datos actuales recolectados en tiempo real por la estación meteorológica. Mediante scripts desarrollados en Python se eliminaron registros incompletos, duplicados o valores atípicos y se homogeneizaron con el formato de fecha, hora y unidades de media de las

diferentes variables climáticas. también se estandarizo los nombres de las columnas para asegurar la compatibilidad entre ambos conjuntos de datos. Dando como resultado construir series temporales consistentes y sin interrupciones, para el entrenamiento del modelo ARIMA.

3.7.2 Organización y almacenamiento de los datos

Los datos climáticos utilizados en este proyecto de titulación provinieron de dos fuentes principales: una base de datos histórica con 10 años de registros adquirida previamente (Mateum), y los datos actuales recolectados en tiempo real por la estación meteorológica. Ambos conjuntos de datos fueron organizados en archivos con formato CSV, estructurados en columnas que representan variables clave como temperatura, humedad y velocidad del aire, junto con la marca de tiempo correspondiente. Para facilitar el análisis los datos históricos fueron almacenados en un archivo principal, mientras que los datos diarios actuales se guardan automáticamente en archivos separados, que luego son integrados periódicamente mediante un script en Python. Este proceso asegura la continuidad temporal de las series, permitiendo que el modelo ARIMA trabaje con información actualizada y consolidada, mejorando así la calidad de las predicciones.

3.7.3 Análisis exploratorio

Una vez preprocesados y organizados los datos, se realizó un análisis exploratorio para comprender el comportamiento de las variables climáticas clave: temperatura, humedad y velocidad del aire. Este análisis permitió identificar tendencias, estacionalidades, posibles anomalías y patrones temporales dentro de las series de datos. Además, se generaron visualizaciones gráficas mediante matplotlib y seaborn, lo que facilitó la interpretación visual de los cambios diarios y estacionales. Para evaluar la viabilidad del modelado con ARIMA, se aplicaron funciones de autocorrelación (ACF) y autocorrelación parcial (PACF), lo que ayudó a determinar el orden de los modelos a entrenar. Este análisis exploratorio fue clave para comprender la dinámica climática del entorno y tomar decisiones fundamentadas en el diseño del modelo predictivo.

3.7.4 Procesamiento para modelo

Para preparar los datos para el entrenamiento del modelo ARIMA, se realizó un procesamiento específico orientado a cumplir con el método estadístico. En primer lugar, se verificó la estacionariedad de las series temporales mediante pruebas visuales y estadísticas, y se aplicaron diferenciaciones a las variables que presentaron tendencias o estacionalidad. A continuación, se seleccionaron los parámetros del modelo ARIMA (p, d, q) basados en el análisis de las funciones de autocorrelación (ACF) y autocorrelación parcial (PACF), así como en criterios de información como el AIC (Criterio de Información de Akaike). El ajuste del modelo se realizó con la librería statsmodels en Python, que permitió estimar los coeficientes y validar su significancia estadística. Finalmente, el modelo entrenado fue utilizado para generar predicciones semanales de las variables climáticas, las cuales se almacenaron en archivos CSV para su posterior evaluación y visualización.

3.7.5 Generación de predicciones

Una vez entrenado y validado el modelo ARIMA, se procedió a la generación automática de predicciones semanales para las variables climáticas clave: temperatura, humedad y velocidad del aire. Este proceso se implementó mediante un script en Python que se ejecuta de forma programada en el servidor (VPS), garantizando la actualización continua de los pronósticos sin intervención manual. Las predicciones obtenidas se almacenan en archivos CSV, que posteriormente son accesibles para su visualización en la interfaz web desarrollada con Flask. Esta automatización permite un monitoreo en tiempo real de las condiciones climáticas previstas, facilitando la toma de decisiones y oportunas en función de los resultados.

3.8 Desarrollo del diseño e implementación del modelo predictivo ARIMA

En este apartado se va a desarrollar paso a paso como se diseñó y se implementó este trabajo de titulación.

3.8.1 Estación meteorológica

La estación meteorológica utilizada en este proyecto está compuesta por diversos sensores que permiten la medición y monitoreo en tiempo real de variables climáticas esenciales. Su componente principal es la WS90 de Ecowitt, un sensor compacto e integrado que se complementa con instrumentos adicionales como un pluviómetro y un anemómetro, donde todos trabajando en conjunto para enviar datos a un servidor local por una Gateway y una ESP32.

Figura 10: Estación meteorológica del proyecto de investigación

Fuente: Autor

Como se puede observar en la Figura 10 el **WS90** es un sensor meteorológico muy completo que integra en un solo módulo la medición de temperatura, humedad, presión barométrica, radiación solar e índice UV. Incorpora un sensor ultrasónico para detectar la velocidad y dirección del viento sin partes móviles. Luego hay también un sensor de precipitación basado en tecnología piezoeléctrica de vibración. Funciona con energía solar y cuenta con batería de respaldo, además de conectividad inalámbrica (WiFi) mediante un gateway que le permite enviar los datos a servicios en la nube o a un servidor local. Su diseño compacto sin piezas mecánicas expuestas facilita la instalación y reduce significativamente el mantenimiento, lo que lo hace ideal para estaciones meteorológicas modernas.

El pluviómetro que se puede ver en la Figura 10 que está a la derecha de la imagen es un instrumento diseñado para medir la cantidad de lluvia caída en un intervalo determinado. En este proyecto se utiliza un modelo tradicional de cubeta basculante, el cual registra cada 0.2 mm de precipitación mediante un sistema de inclinación mecánica. Entre sus principales características destacan su precisión, su alta sensibilidad y su estructura resistente a la intemperie así facilita su instalación en exteriores. Este sensor está conectado mediante un cable de datos a un microcontrolador ESP32, el cual interpreta los pulsos generados por el mecanismo y transmite la información a un servidor local, donde se almacena y visualiza en tiempo real.

El anemómetro que se puede observar en la Figura 10 a la izquierda es el sensor encargado de medir velocidad del viento, utilizando un sistema mecánico compuesto por cazoletas giratorias. Este diseño permite obtener lecturas continuas de la intensidad del viento en unidades como km/h o m/s. Podemos destacar en sus características que está fabricado con materiales resistentes, para soportar condiciones climáticas adversas en exteriores. Su funcionamiento está vinculado directamente a una ESP32 que se conecta mediante un cable de datos. Desde allí, los valores capturados son procesados y enviados a un servidor local.

3.8.2 Configuración y puesta en marcha de la estación meteorológica

La estación meteorológica WS90 junto al pluviómetro y anemómetro se comunica por radiofrecuencia (915 MHz) con la puerta de enlace GW3000, la cual cuenta con conectividad WiFi. Esto permite transmitir los datos en tiempo real a través de la red local hacia servidores externos o servicios en la nube. La comunicación se realiza mediante protocolos como HTTP POST o MQTT, a través de una API configurable por el usuario.

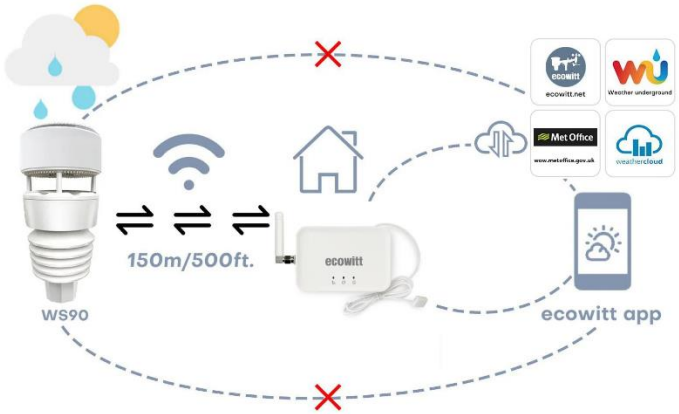


Figura 11: Diagrama de conexión WS90 al GW3000

Fuente: Autor

Para la puesta en marcha del sistema, se realizó en primer lugar la conexión física y eléctrica tanto de la estación meteorológica como del módulo GW3000 y el ESP32 a la red de alimentación. Luego, el GW3000 fue conectado mediante un cable Ethernet al router de internet que permitió una conexión estable sin necesidad de configurar acceso por Wi-Fi. Accediendo desde un navegador web a la dirección IP predeterminada 192.168.4.1, se ingresó a la interfaz de configuración del dispositivo desde donde es posible establecer parámetros de red. Posteriormente, se configuraron las unidades de medida de los sensores, seleccionando las unidades que el usuario quiera en este caso se seleccionaron las siguientes que se puede ver en la Figura 12.

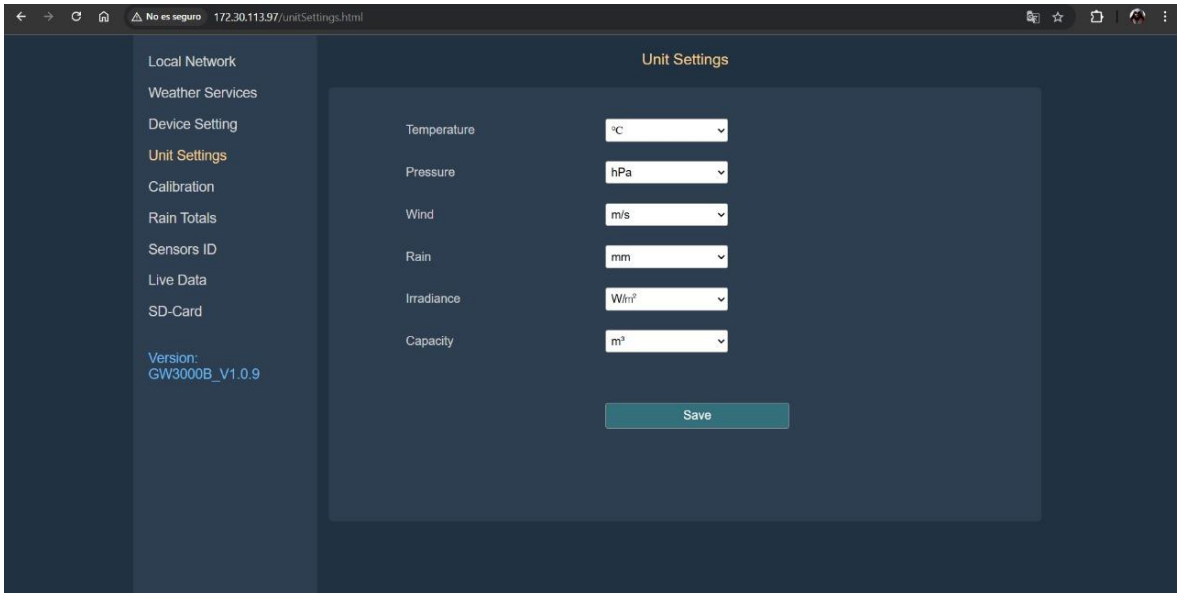


Figura 12: Configuración de las unidades

Fuente: Autor

Finalmente, en la sección Weather Services de dicha interfaz, se habilitó el protocolo Ecowitt para el envío de datos hacia un servidor personalizado, como se observa en la Figura 13.

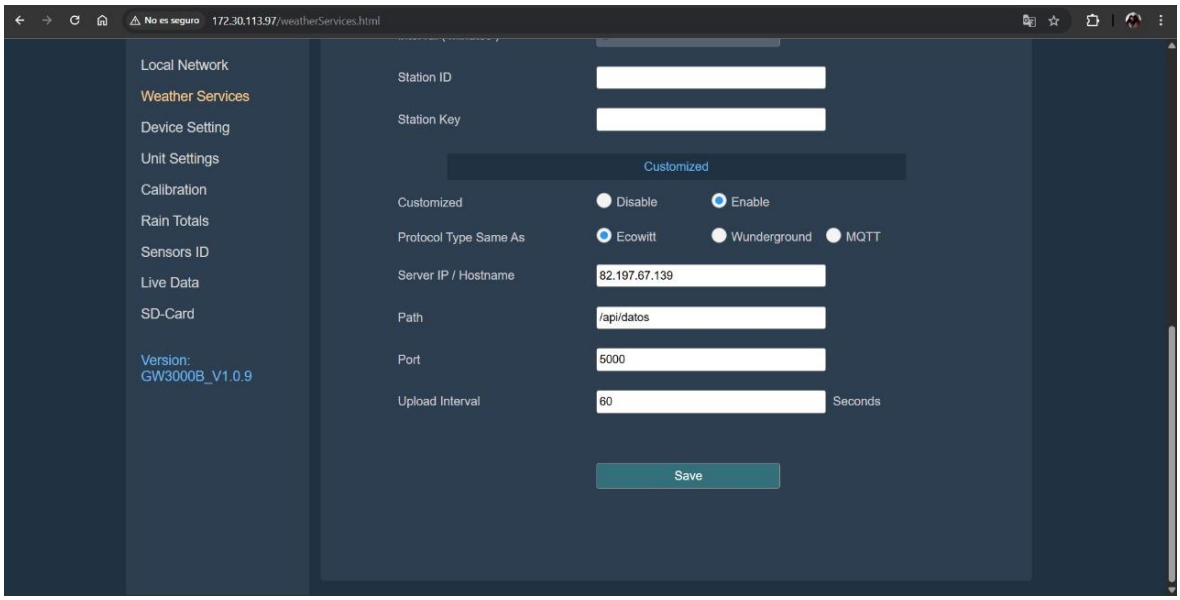


Figura 13: Configuración del servidor personalizado

Fuente: Autor

En esta configuración se especificaron el protocolo, la dirección IP del servidor, el puerto, el path de la API y el intervalo de envío, que en este caso fue establecido en 1 minuto para asegurar una actualización frecuente de los datos en tiempo real de igual manera se hizo con la ESP32 con el script que se puede ver en el Anexo 10.

3.8.3 Validación de recepción de datos a tiempo real

Para verificar la correcta recepción de los datos, se utilizó inicialmente un script como se puede ver en la Figura 14 el cual imprime en la consola todos los campos recibidos de la estación meteorológica. Se observó la llegada periódica de los registros de temperatura, humedad, viento, presión, radiación y lluvia. Además, se revisó el archivo de almacenamiento temporal (CSV) generado por el script receptor, corroborando que los datos se almacenaban de manera continua y sin pérdidas como se observa en la Figura 15.

```python
from flask import Flask, request
from datetime import datetime, timedelta
import csv
import os
import json

app = Flask(__name__)

# Ruta absoluta hacia ../datos
BASE_DATOS = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'datos'))

def grados_a_cardinal(grados):
    direcciones = [
        "N", "NNE", "NE", "ENE", "E", "ESE", "SE", "SSE",
        "S", "SSO", "SO", "OSO", "O", "ONO", "NO", "NNO"
    ]
    idx = int((grados + 11.25) / 22.5) % 16
    return direcciones[idx]

@app.route('/api/datos', methods=['POST'])
def recibir_datos():
    data = request.form.to_dict()

    # Obtener fecha UTC desde la estación
    fecha_utc_str = data.get("dateutc")
    if fecha_utc_str:
        fecha_utc = datetime.strptime(fecha_utc_str, "%Y-%m-%d %H:%M:%S")
        fecha_local = fecha_utc - timedelta(hours=5)  # Ecuador
    else:
        fecha_local = datetime.now()

    # === Extracción y conversión de datos ===
    temperatura_f = float(data.get("tempf", 0))
    humedad = float(data.get("humidity", 0))
    velocidad_mph = float(data.get("windspeedmph", 0))
    direccion_viento = float(data.get("winddir", 0))
    lluvia_hora = float(data.get("rainin", 0))
    lluvia_dia = float(data.get("dailyrainin", 0))
    presion_rel = float(data.get("baromrelin", 0))
```

Figura 14: Fragmento del código en Python para la recepción de datos de la estación meteorológica

Fuente: Autor

Como se ve en la Figura 15 este código recibe datos meteorológicos por el método POST desde una estación (como el Ecowitt GW3000) los procesa (convirtiendo unidades y calculando fecha local) los guarda en un archivo CSV y también genera un archivo JSON con el último registro para su visualización en una interfaz web.

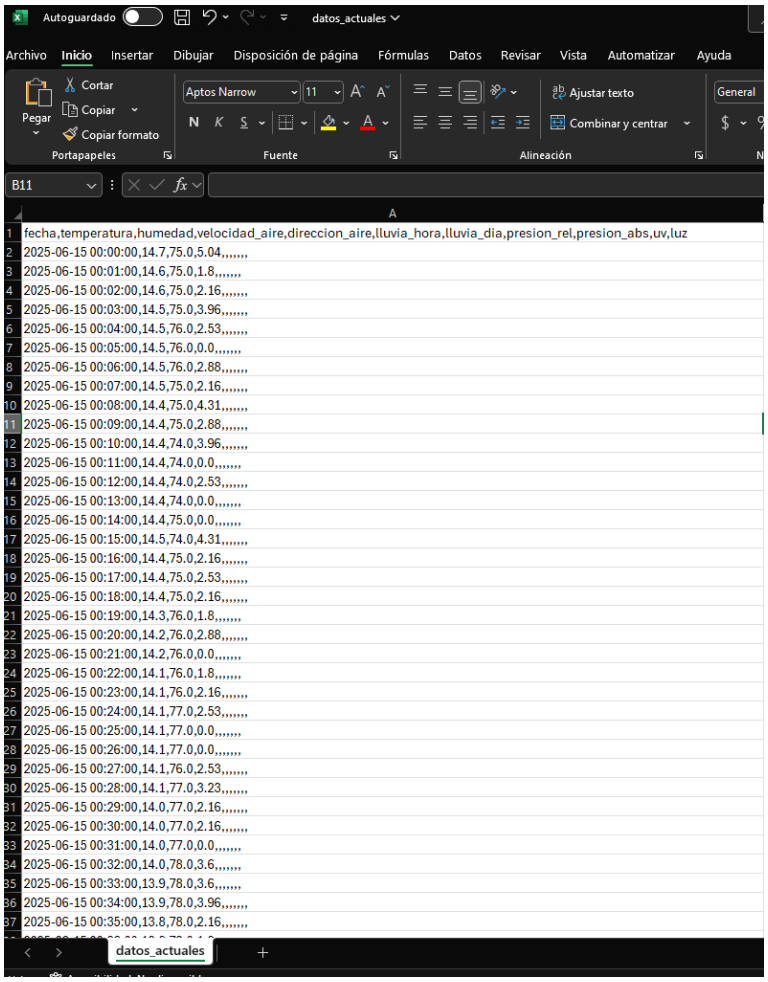


Figura 15: Datos actuales guardados en CSV

Fuente: Autor

3.8.4 Adquisición del historial climático

En este punto con el fin de realizar predicciones más precisas y evaluar el comportamiento climático de la zona, se adquirió una base de datos histórica que contiene registros detallados de variables meteorológicas durante un período de 10 años. Esta información es fundamental para entrenar y validar modelos de predicción basados en series temporales.

Esta base de datos histórica contiene registros diarios de variables meteorológicas como temperatura, humedad relativa y velocidad del viento. Estos datos fueron recopilados de una fuente confiable especializada en información climática (Mateum) con cobertura completa y sin grandes vacíos temporales. La frecuencia de muestreo es diaria, y cada registro incluye una marca temporal en formato ISO 8601 como se puede observar en la Figura 16.

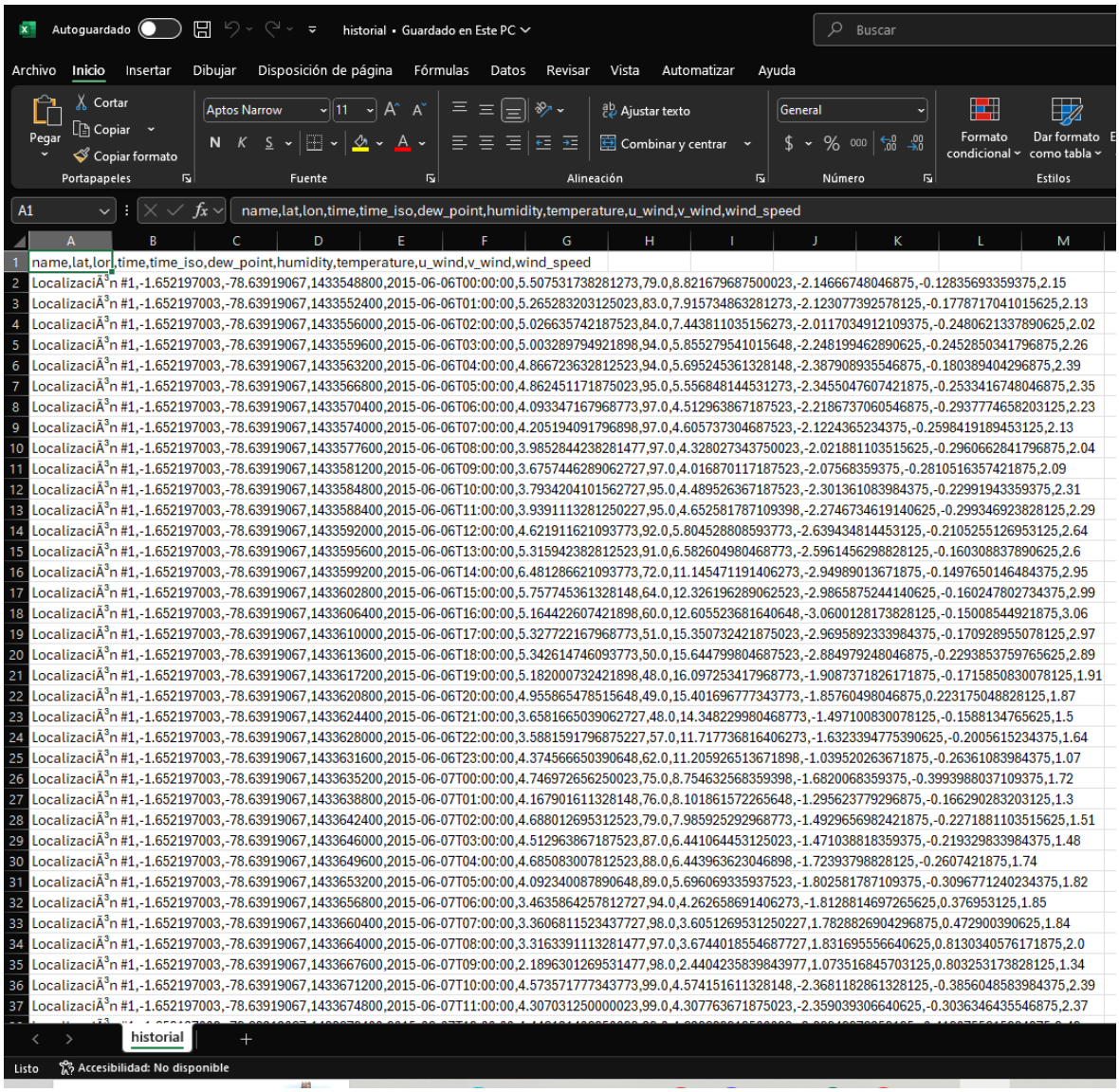


Figura 16: Historial climático de 10 años

Fuente: Autor

Los datos fueron entregados en formato CSV, ampliamente compatible con herramientas de análisis de datos. Cada archivo contiene las siguientes columnas: time_iso, temperature, humidity, wind_speed que son las variables que se van a utilizar en este proyecto. Las unidades están expresadas en grados Celsius (°C) para la temperatura, porcentaje (%) para la humedad y kilómetros por hora (km/h) para el viento. Este formato facilita su procesamiento automatizado mediante scripts en Python.

Por último, en esta parte para llevar a cabo un análisis conjunto y mantener la continuidad de los datos, la información histórica se integró con los datos actuales recolectados por la estación meteorológica Ecowitt GW3000. Este proceso implicó la unificación de formatos, normalización de unidades y alineación temporal. Los datos actuales se

almacenan en archivos CSV con la misma estructura que los históricos, lo que permitió su fusión sin complicaciones.

3.8.5 Preprocesamiento y limpieza de datos

Antes de entrenar el modelo de predicción climática, fue necesario realizar un proceso de preprocesamiento y limpieza de los datos históricos y actuales. Esta etapa garantiza la calidad, coherencia y utilidad de los datos utilizados, eliminando errores, valores atípicos o registros incompletos que podrían afectar el rendimiento del modelo. Para la limpieza del historial climático de 10 años se utilizó el siguiente script que se puede ver en la Figura 17.

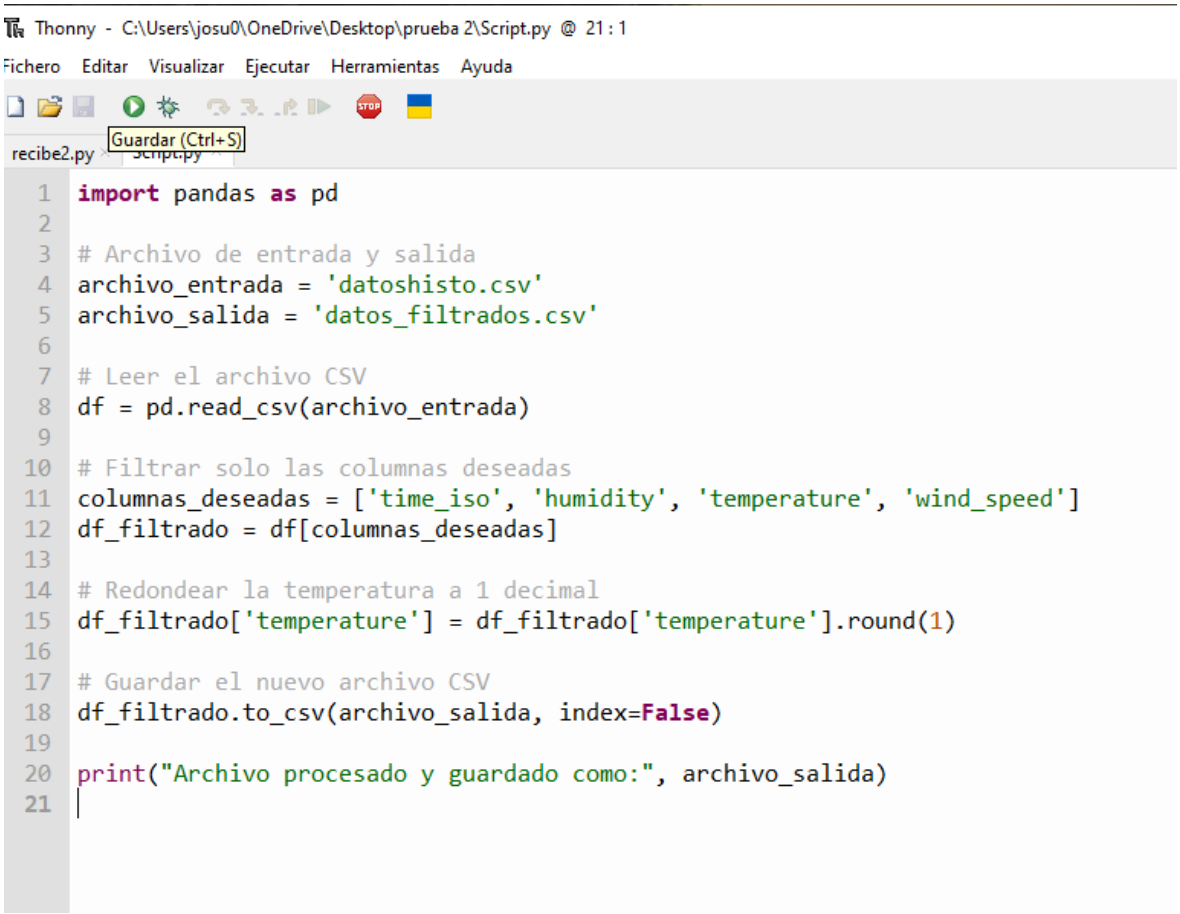


Figura 17: Script de limpieza del historial de 10 años

Fuente: Autor

Este código en Python lee historial de 10 años, selecciona solo las columnas time_iso, humidity, temperature y wind_speed, redondea los valores de la columna de temperatura a un decimal, y guarda el resultado en un nuevo archivo llamado datos_filtrados.csv como se muestra en la Figura 18 que ya tiene el formato adecuado ya para el siguiente paso.

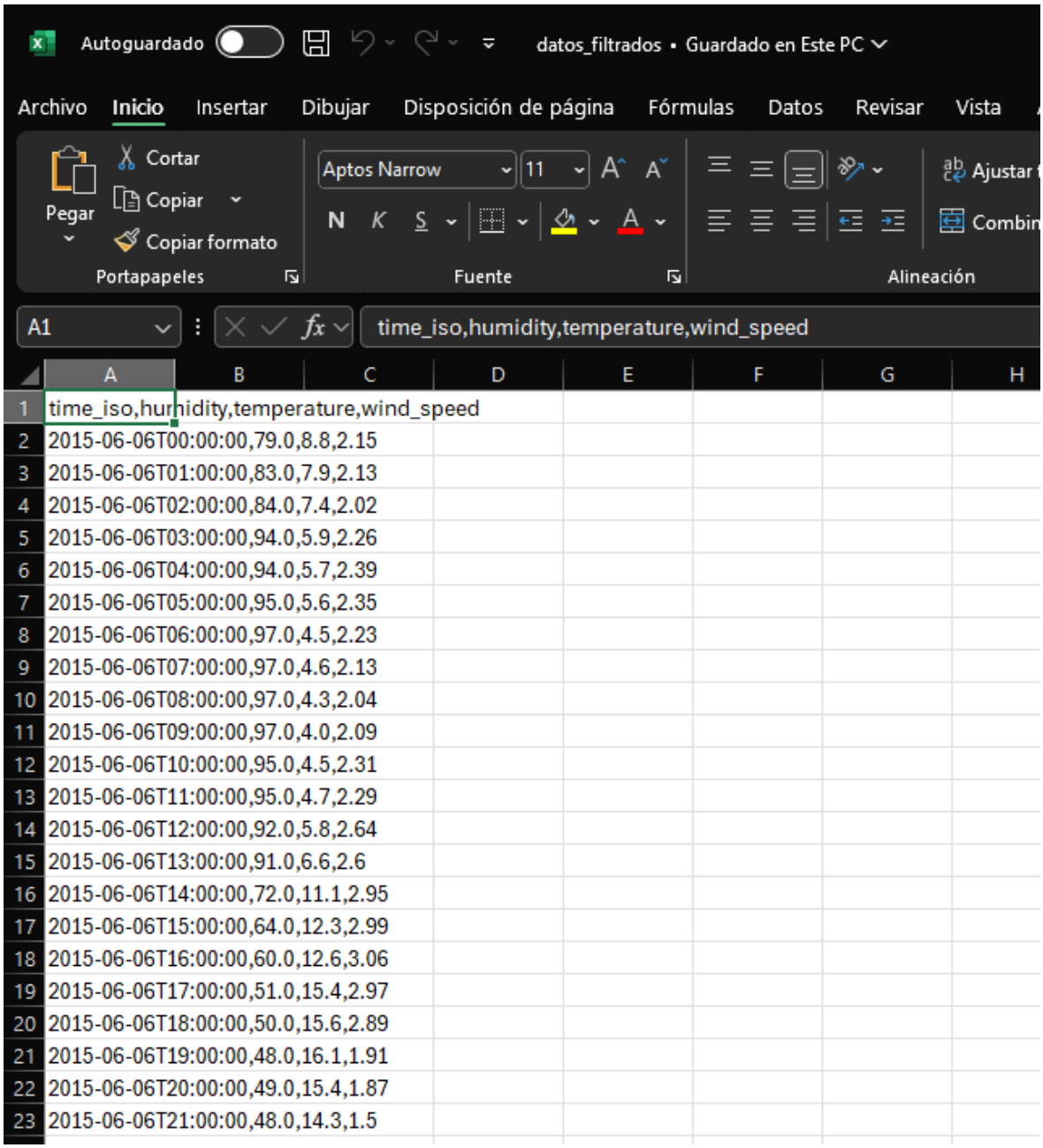


Figura 18: Datos filtrados y arreglados del historial de 10 años

Fuente: Autor

3.8.6 Entrenamiento del modelo ARIMA

Con el objetivo de realizar predicciones semanales de las variables climáticas, se implementó y evaluó un modelo ARIMA (AutoRegressive Integrated Moving Average) en un entorno local utilizando Python. Esta etapa incluyó el entrenamiento del modelo, pruebas con datos reales y ajustes necesarios para mejorar la precisión de las predicciones.

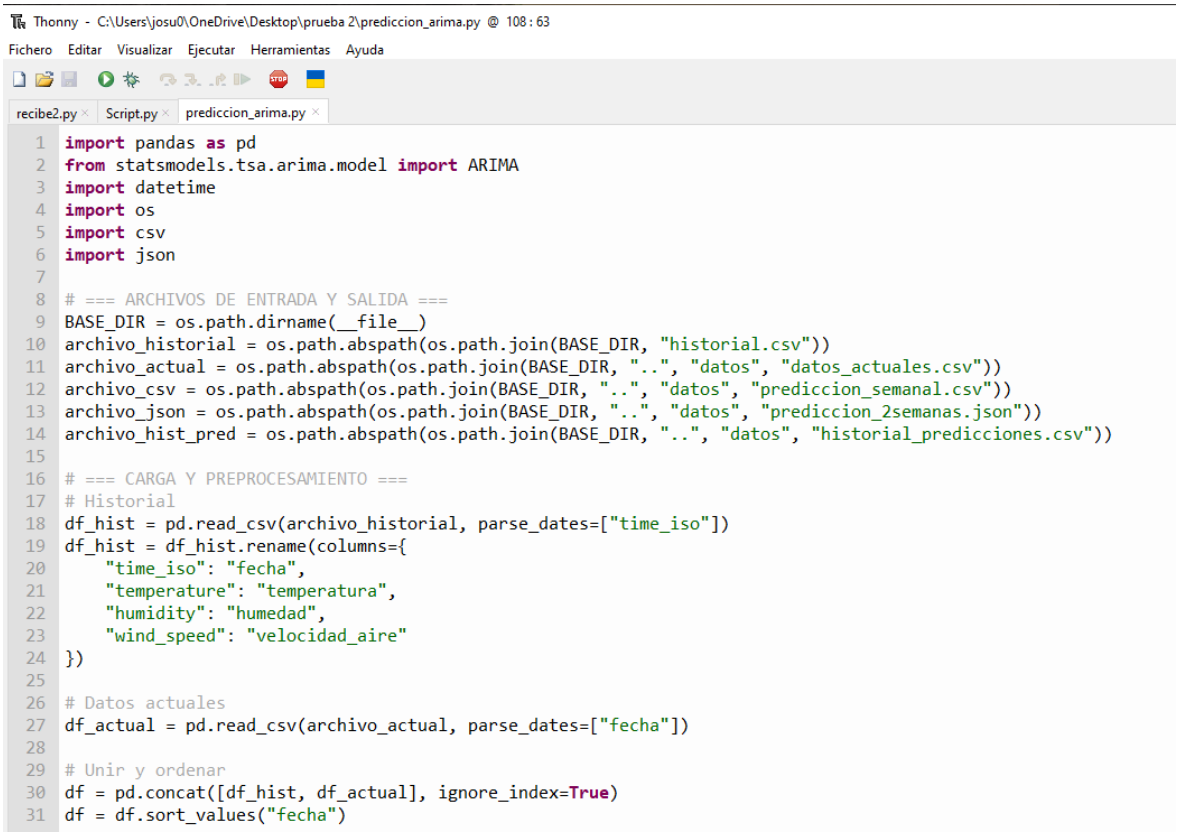


Figura 19: Fragmento del codigo de prediccion modelo ARIMA

Fuente: Autor

El script que se puede ver en la Figura 19 para realizar el entrenamiento y predicción fue desarrollado en Python utilizando Thonny y emplea las librerías statsmodels, pandas y matplotlib para procesar datos históricos y actuales desde archivos CSV, ajustar un modelo ARIMA y generar predicciones semanales. De las variables como temperatura, humedad o velocidad del viento mandando los resultados en formatos CSV y JSON. La elección de los parámetros del modelo (p, d, q) se basó en análisis ACF y PACF. El script que se puede ver en la Figura 20 fue probado con distintos subconjuntos de datos para verificar su precisión, estabilidad y rendimiento.

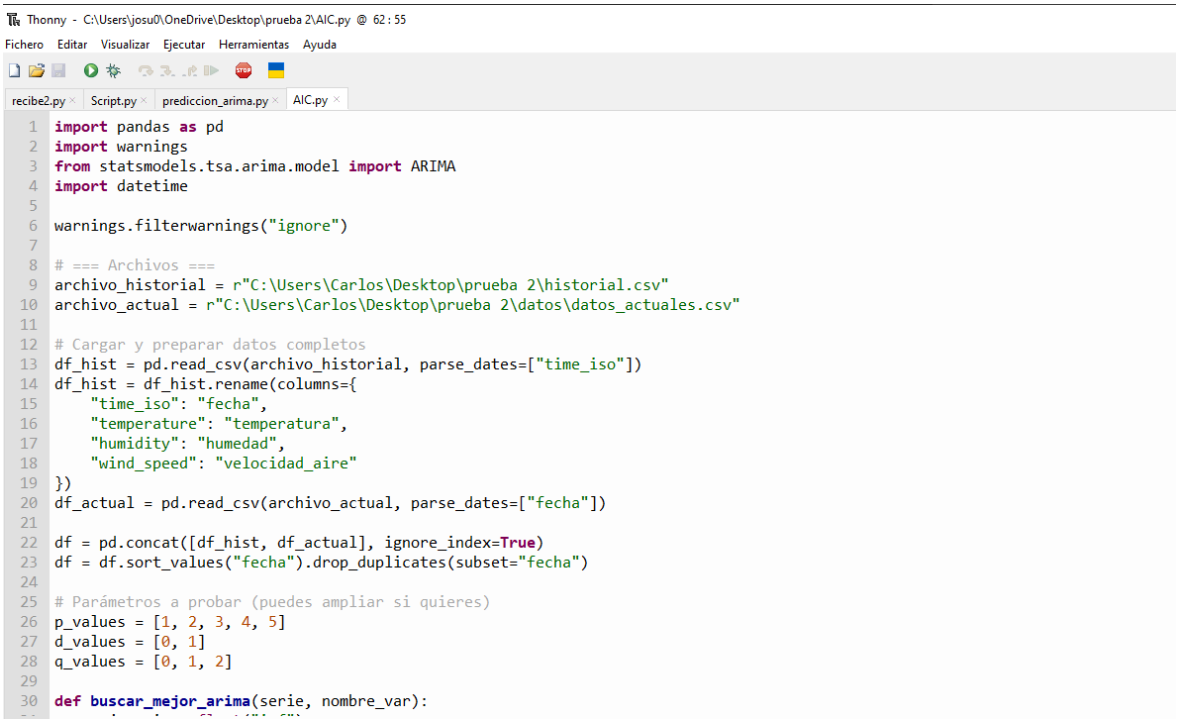


Figura 20: Fragmento del script AIC

Fuente: Autor

Este código en Python busca el mejor modelo ARIMA para predecir temperatura, humedad y velocidad del viento. Combina datos históricos y actuales para luego carga los datos desde archivos CSV, los unifica y ordena, y luego prueba distintas combinaciones de parámetros (p, d, q) para cada variable, evaluando cada modelo según el criterio AIC (Akaike Information Criterion es una medida estadística que evalúa la calidad de un modelo considerando su precisión y complejidad, donde si es menor el valor de AIC se ajusta el modelo a los datos). Al final la combinación se determinó que el modelo ARIMA (5, 1, 2) proporciona los mejores resultados por ello fue utilizado en el script de predicción final para generar pronósticos semanales.

3.8.7 Migración e implementación en el servidor VPS

Ya con la comprobación de que va bien en el entorno local (Pc personal) se pasó al siguiente paso que es ofrecer un sistema de predicción climática accesible desde cualquier red, se migró el proyecto a un servidor VPS (Servidor Privado Virtual), lo que permite ejecutar los scripts de forma automatizada y mantener actualizada la información en una página web. Se contrató un servidor VPS con sistema operativo Ubuntu, acceso completo vía SSH en mi caso me conecte por PuTTY que ofrece la vía SSH como se ve en la Figura 21.

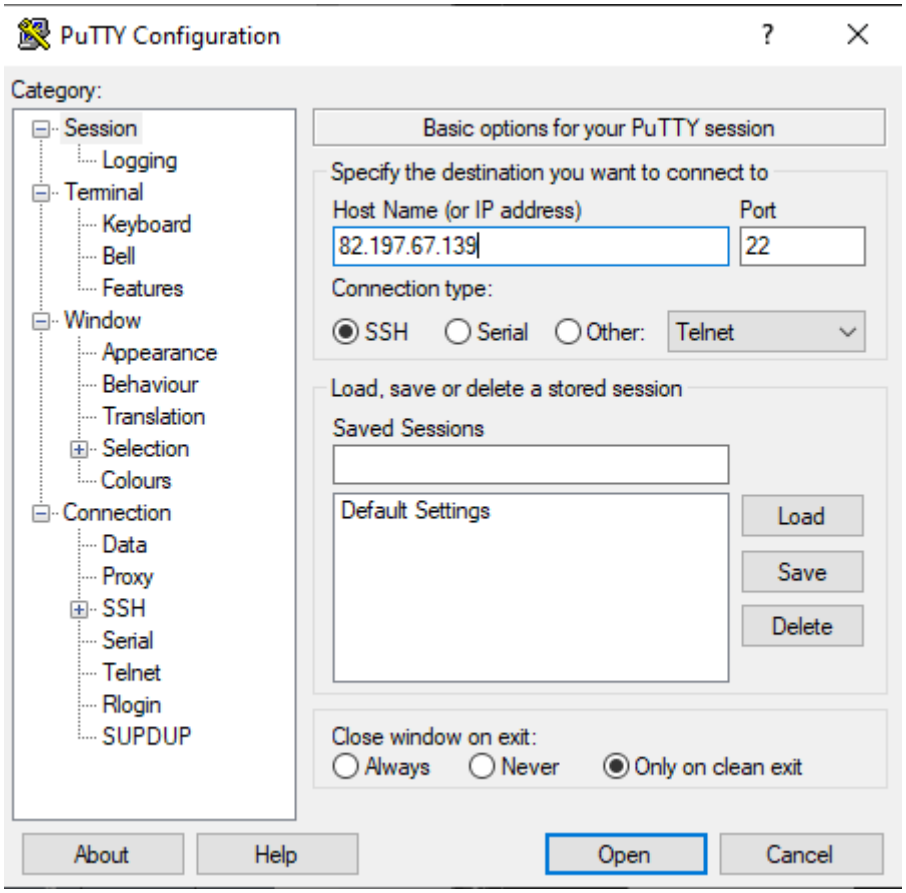


Figura 21: Accediendo a la VPS por PuTTY

Fuente: Autor

El VPS fue configurado para operar sin entorno gráfico así está garantizando eficiencia y control total desde la terminal. Se asignaron permisos, se creó una estructura de carpetas que se puede ver en la Figura 23 para el proyecto y se configuraron reglas básicas de firewall y seguridad.

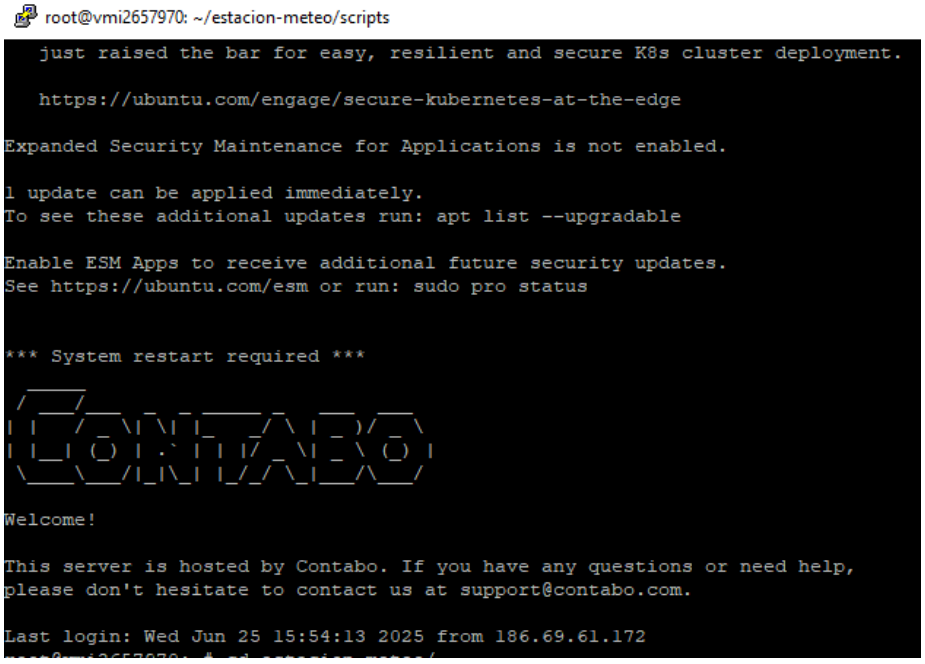


Figura 22: VPS controlado por PuTTY

Fuente: Autor

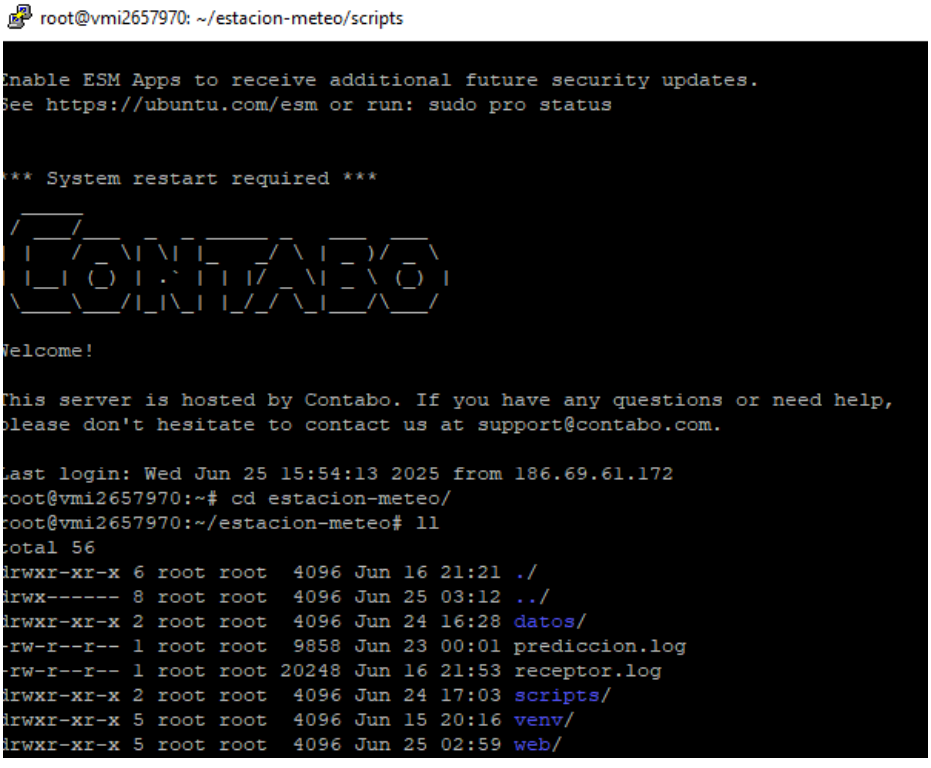


Figura 23: Directorios creados para este proyecto en la VPS

Fuente: Autor

En la Figura 23 se muestra la estructura del proyecto implementado en el servidor VPS. El directorio principal es estacion-meteo/ que dentro del cual se organizaron cuatro

subdirectorios clave. El primero, datos/, donde se almacena los archivos generados en formato CSV y JSON por los scripts de recepción y predicción ARIMA; estos archivos son posteriormente leídos por la página web para su visualización. El directorio scripts/ contiene los scripts principales, donde uno que recibe los datos enviados por la estación meteorológica y los guarda en datos/ y otro que realiza la predicción ARIMA utilizando el historial climático de 10 años (está dentro del directorio scipts/) junto con los datos actuales (directorio /datos). Este segundo script genera una predicción de 14 días y también guarda los resultados en el directorio datos/. El directorio venv/ corresponde al entorno virtual de Python, donde se almacenan todas las librerías necesarias para la correcta ejecución de los scripts.
Adicionalmente, los archivos .log contienen registros y alertas del sistema, útiles para el monitoreo.
Finalmente, el directorio web/ alberga el servidor web, cuyo archivo principal es app.py. Dentro de web/templates/ se encuentra el archivo index.html, que representa la interfaz principal del sitio, y en web/static/ se almacenan los recursos estáticos como imágenes y elementos decorativos utilizados en el diseño de la página.

Volviendo a la implantación y para asegurar un entorno controlado y reproducible, se creó un entorno virtual de Python en el VPS, donde se instalaron todas las dependencias necesarias como Flask, pandas, statsmodels, numpy y matplotlib, garantizando la compatibilidad con los scripts desarrollados localmente. Luego, mediante el uso de scp, se subieron al servidor los scripts encargados de recibir datos, realizar predicciones y generar salidas, junto con los archivos CSV de datos históricos y actuales ya preprocesados, organizando todo en una estructura de carpetas clara y funcional como se puede observar en la Figura 24 y 25. Finalmente, se automatizó la ejecución de los scripts mediante cron, que permite programar tareas como la predicción semanal.

Figura 24: Extensión de PuTTY (pscp) que se utilizó para migrar los archivos locales a la VPS

Fuente: Autor

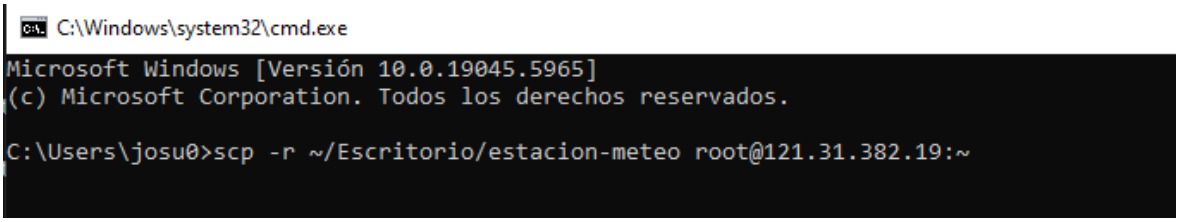


Figura 25: Ejemplo de cómo pasar mediante pscp del pc local Windows al servidor VPS (Ubuntu)

Fuente: Autor

Resumiendo lo que se ve en la Figura 25 el comando scp permite copiar archivos de forma segura entre máquinas. La opción -r se utiliza para transferir de forma recursiva todo el contenido de un directorio. En el ejemplo, ~/Escritorio/estacion-meteo representa la ruta local donde se encuentra el proyecto en mi caso es ese, mientras que root@121.31.382.19 indica la dirección IP del servidor y el nombre de usuario remoto. Finalmente, "**~**" especifica que los archivos se copiarán al directorio home del usuario en el servidor.

3.8.8 Desarrollo del backend de recepción y predicción

En el tema del desarrollo yo hice el backend del sistema con dos scripts principales que permiten la recolección de datos meteorológicos en tiempo real y la generación automática de predicciones. Estos scripts fueron desarrollados en Python, probados localmente y luego implementados en el servidor VPS para su ejecución continua como ya se comentaron en los anteriores puntos.

El script **receptor.py** utiliza Flask para crear una API REST que recibe datos desde la estación meteorológica Ecowitt GW3000 mediante el protocolo HTTP POST. Una vez que los datos llegan el script los procesa y convierte las unidades necesarias (como temperatura en °C o velocidad del viento en km/h) y los guarda en formato CSV y JSON dentro del directorio datos/. Esta información queda lista para ser utilizada por el sistema de predicción y visualizada en la página web. El segundo componente es el **script de predicción ARIMA**, que lee los datos históricos y actuales, aplica un modelo previamente ajustado y genera predicciones semanales de temperatura, humedad y velocidad del viento. Estas predicciones se exportan en archivos CSV y JSON, que son luego consultados por el frontend web para su presentación. El script se ejecuta automáticamente cada semana gracias a la configuración de tareas programadas con cron que en este caso le puse que lo haga todos los domingos a las 23:59 para que el lunes comienzo de semana ya tenga su predicción.

Finalmente, se validó el funcionamiento automático de ambos scripts en el VPS, comprobando que el receptor funciona de manera continua y sin interrupciones, y que el script de predicción se ejecuta correctamente según lo programado, generando los archivos de salida en la ubicación esperada y sin errores. Esto confirma que el backend

opera de forma autónoma, garantizando la actualización constante del sistema de monitoreo y predicción climática.

3.8.9 Integración de la aplicación web

La aplicación web integra un backend en Python con Flask y un frontend responsivo usando TailwindCSS y Chart.js. El script principal (app.py) procesa y expone automáticamente tanto los datos actuales como las predicciones ARIMA, visualizándolos en tiempo real mediante tablas y gráficas históricas diarias de temperatura, humedad y viento en la plantilla index.html. Para actualizar la interfaz, basta editar los archivos fuente y reiniciar el servidor con comandos como ps aux | grep gunicorn para ubicar procesos activos y kill <pid> para detenerlos, seguido de /root/venv/bin/gunicorn -w 4 -b 127.0.0.1:8000 app:app --daemon para relanzar el backend de forma automatizada hay que tener en cuenta que hay que estar en el modo virtual de Python sino da error.
El despliegue y la publicación segura se realizan empleando Gunicorn como servidor WSGI (es un programa que conecta aplicaciones web en Python con Internet.), operando detrás de un proxy inverso Nginx que filtra y redirige el tráfico HTTP/HTTPS, exponiendo únicamente el puerto público necesario y mejorando la seguridad frente a accesos directos o ataques. Así, Gunicorn y Flask quedan protegidos tras Nginx, y la aplicación se mantiene disponible y robusta. Este esquema permite una integración automática de datos y predicciones, asegurando la continuidad y confiabilidad del monitoreo climático en tiempo real. A continuación, en la Figura 26 una parte del código del diseño de la página web para este trabajo de titulación.

index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Estación Meteorológica</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">
    <style>
        body {
            background-color: #0f172a; /* Azul oscuro */
            color: white;
        }
        .cuadro {
            background-color: rgba(255, 255, 255, 0.1);
            padding: 1rem;
            border-radius: 0.5rem;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
        }
        .flecha {
            width: 100px;
            height: 100px;
            position: absolute;
            top: 0;
            left: 0;
            transform-origin: 50% 50%;
        }
    </style>
</head>
<body class="p-6 font-sans">

    <!-- ENCABEZADO CENTRADO -->
    <div class="flex flex-col md:flex-row justify-between items-center mb-8 text-center md:text-left">
        <div class="flex-1">
            <h1 class="text-3xl font-bold text-center">Universidad Nacional de Chimborazo</h1>
            <p class="text-xl text-center">Facultad de Ingeniería</p>
            <p class="text-lg text-center">Carrera de Telecomunicaciones</p>
            <p class="text-sm text-center mt-2">Autor: CARLOS LOPEZ</p>
        </div>
        <div class="w-24 h-24 mt-4 md:mt-0">
```

Figura 26: Fragmento del script del diseño de la página index.html

Fuente: Autor

3.8.10 Visualización de los datos en la página WEB

En este apartado, una vez completados los pasos anteriores se mostrará todo lo programado en la página web. A continuación, se explicará qué información se visualizará y cómo estará organizada como se puede ver en la Figura 27 y 28.

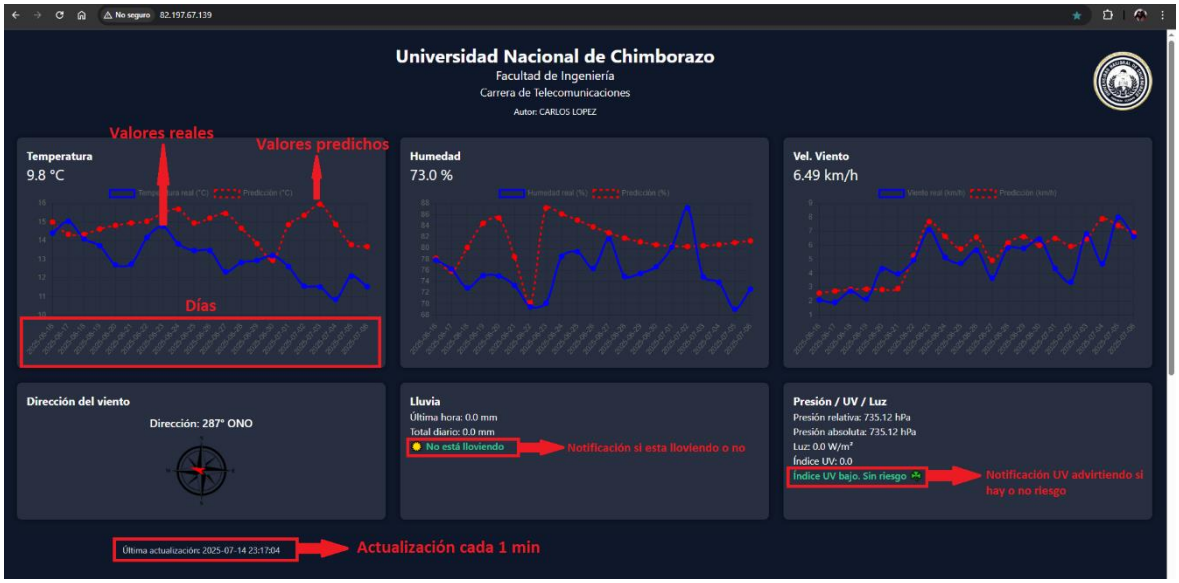


Figura 27: Página WEB parte 1

Fuente: Autor

En la Figura 27 se observan seis cuadros informativos. Los tres cuadros superiores muestran gráficamente los datos recogidos por la estación meteorológica (línea azul) y los valores predichos (línea roja) para tres variables: temperatura, humedad relativa y velocidad del viento. En todos ellos, el eje X representa los días, mientras que el eje Y varía según la variable: en el caso de la temperatura, se muestra en grados Celsius (°C); para la humedad, en porcentaje (%); y en el gráfico de viento, la velocidad se representa en kilómetros por hora (km/h).

Los tres cuadros inferiores muestran otras variables clave. El primero representa la dirección del viento mediante una brújula animada que indica hacia dónde sopla en tiempo real. El segundo cuadro muestra los datos de precipitación, incluyendo la lluvia acumulada en la última hora, el total diario (en mm), y una notificación que indica si está lloviendo en ese momento. Finalmente, el último cuadro combina presión atmosférica, radiación solar y índice UV: se presentan tanto la presión relativa como absoluta en hPa, la intensidad de luz solar en W/m², y el índice UV, acompañado de una alerta visual si se detecta un nivel de radiación potencialmente riesgoso. Todo esto es lo que se visualiza en esta primera figura.

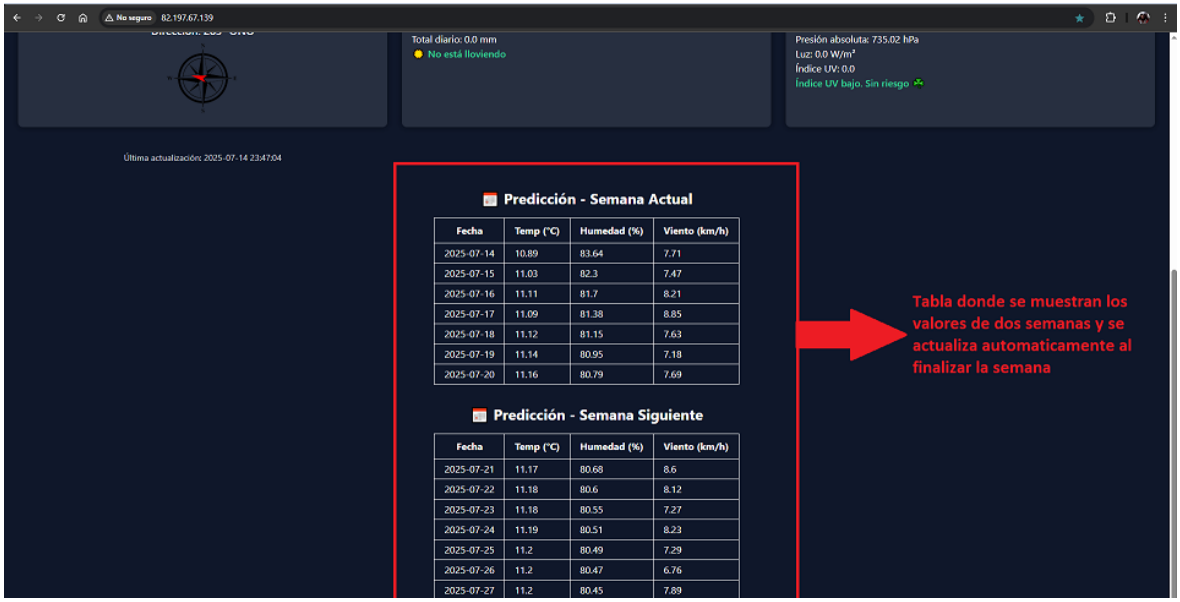


Figura 28: Página WEB parte 2

Fuente: Autor

Por último, en la Figura 28 se muestran las tablas de predicción correspondientes a la semana actual y la siguiente. Estas tablas se actualizan automáticamente cada domingo, proporcionando una proyección semanal actualizada de las variables climáticas.

CAPÍTULO IV

4. RESULTADOS Y DISCUSIÓN

Los resultados recolectados en este proyecto de titulación corresponden a los datos obtenidos de la estación meteorológica y del software de predicción con el modelo ARIMA. Los resultados de la tesis, se estableció una cantidad de 63 datos por las tres variables ambientales (temperatura, humedad y velocidad del viento) en un periodo de tiempo de 3 semanas.

4.1 Precisión del modelo de predicción ARIMA

La tabla 2 presenta un resumen de las características principales del modelo ARIMA que fue desarrollado para la predicción de las variables ambientales temperatura, humedad y velocidad del viento.

Tabla 2: Detalles del modelo ARIMA

Características	Descripción
Tiempo de desarrollo	8 semanas
Número de líneas del código	115 líneas de código
Datos del historial climático	263091 registros (10 años)
Variable Pronosticada	Temperatura, Humedad, Velocidad del viento
Frecuencia de datos históricos	1 hora
Frecuencia de datos actuales	1 minuto
Lenguaje de programación	Python
Tamaño promedio de predicción	14 días (2 semanas)

4.1.1 Error relativo medio (ERM)

Para evaluar la precisión del modelo ARIMA de las predicciones generadas, se calculó el error cuadrático medio (ECM) entre los valores reales obtenidos por la estación meteorológica y los valores predichos por el modelo para los días siguientes con la siguiente fórmula.

$$ERM = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)$$

Donde y_i es el valor real y $\tilde{y}_i$ es el valor predicho. Los resultados obtenidos fueron los siguientes:

Tabla 3: Resultados obtenidos al aplicar ERM

Variable	Porcentaje (Error Relativo)
Temperatura	14.63 %
Humedad	11.49 %
Viento	24.83 %

Fuente: Autor

Como se puede ver en la Tabla 3 el modelo de predicción demostró una precisión moderada en la variable temperatura, con un ERM del 14.63 %, lo que resulta aceptable para aplicaciones generales. En el caso de la humedad salió el ERM de 11.49 % que refleja un buen comportamiento del modelo, permitiendo anticipar sus valores con buena precisión. Por último, la predicción del viento resultó la más desafiante con un ERM de 24.83%, pero como está cercano a un valor del 20 % se estima que según se vaya incrementado los datos actuales el modelo vaya acercándose más a los valores reales recogidos por la estación meteorológica.

4.1.2 Comparación gráfica de datos reales y predichos

Se generaron gráficas para visualizar la evolución diaria de temperatura, humedad y velocidad del viento, mostrando tanto los valores reales obtenidos por la estación como los valores predichos por el modelo ARIMA en un periodo de tres semanas.

En la temperatura se observa que el modelo sigue adecuadamente la tendencia general de la temperatura, aunque tiende a sobreestimar los valores en la mayoría de los días. Si bien la curva predicha captura la forma global, existen discrepancias notorias en los extremos y variaciones bruscas, lo que sugiere que el modelo es menos sensible a los cambios abruptos que sí ocurren en los datos reales como se observa en la Figura 29.

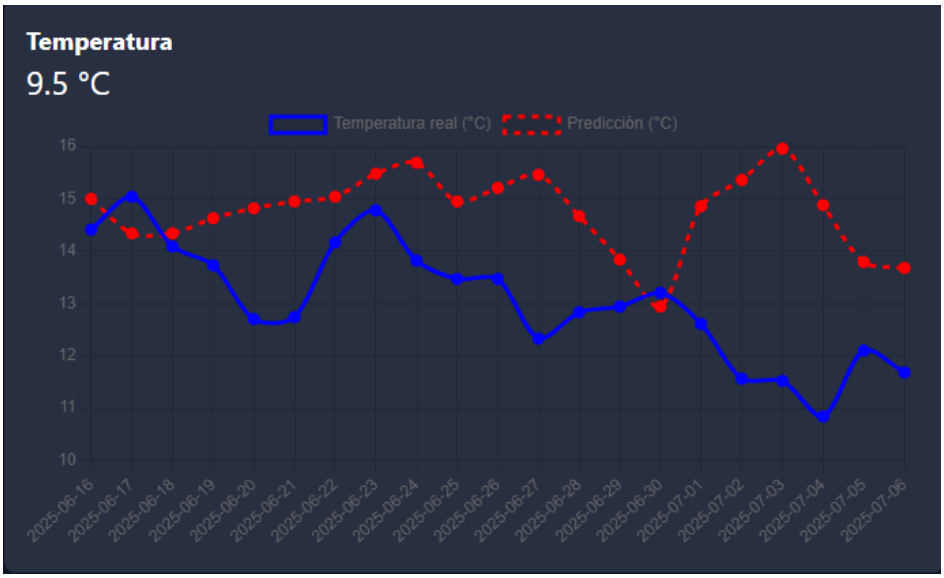


Figura 29: Gráfica de temperatura datos reales y datos predichos

Fuente: Autor

En la humedad el modelo se observa de manera aceptable el patrón general, aunque la serie predicha es visiblemente más suave y estable en los datos reales. Esto indica que el modelo ARIMA prioriza la tendencia sobre la adaptación a los valores puntuales como se ve en la Figura 30.

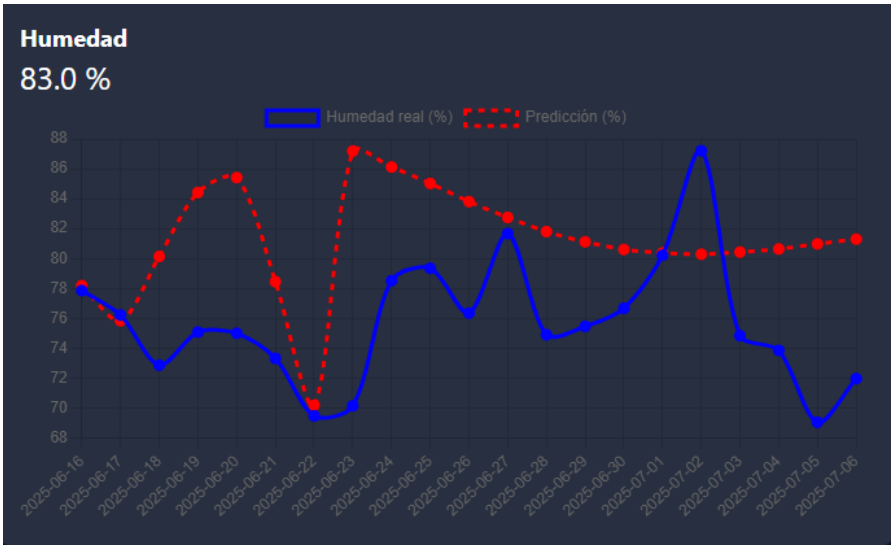


Figura 30: Gráfica de humedad datos reales y datos predichos

Fuente: Autor

Por último, en el viento la predicción coincide en términos generales, reflejando los principales picos y valles del periodo. No obstante, el modelo tiende a suavizar las variaciones más extremas, logrando un buen ajuste de la tendencia, pero sin captar completamente los episodios de mayor variedad como se puede ver en la Figura 31.

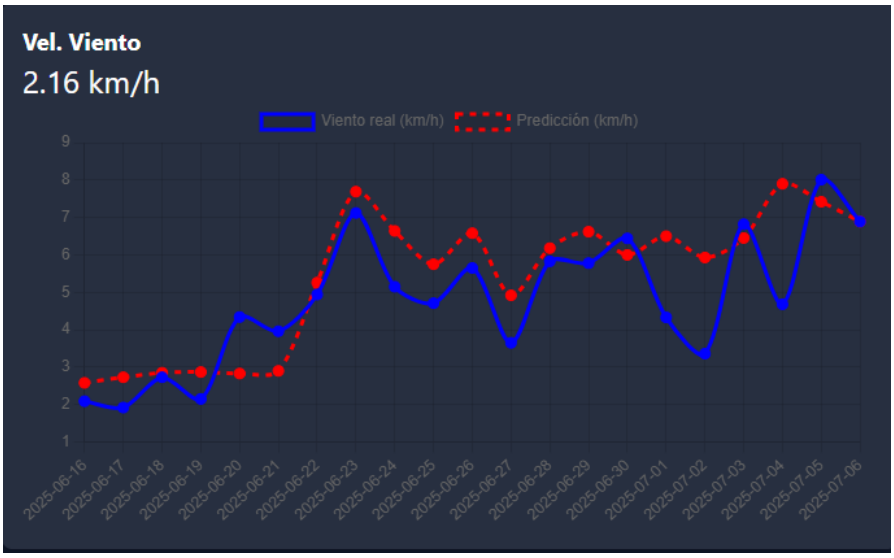


Figura 31: Gráfica de velocidad del viento datos reales y datos predichos

Fuente: Autor

4.2 Discusión de los Resultados

Los resultados obtenidos ponen en evidencia tanto las fortalezas como las limitaciones del modelo ARIMA implementado para la predicción meteorológica en la Universidad Nacional de Chimborazo. En términos generales, se observa que el modelo es capaz de capturar la tendencia promedio de las variables, aunque con diferencias en algunos puntos, pero a medida que se vaya recolectando datos diariamente el modelo se adaptara mejor.

Para la temperatura como se observó en el apartado anterior observando la Tabla 3 este valor indica un desempeño aceptable ya que está por debajo del 20 %. En la gráfica que muestra la Figura 29 tiene una tendencia a sobreestimar los valores reales y a no capturar con precisión los descensos o aumentos bruscos de temperatura que se presentan en el sitio en el lugar de medición. Esto puede atribuirse a la influencia de microclimas locales y a la propia naturaleza del modelo ARIMA que tienden a suavizar las series.

En el caso de la humedad como se observó en el apartado anterior como la temperatura observando la Tabla 3. Esto sugiere que la humedad en la zona presenta patrones más estables y menos sujetos a variaciones extremas, lo que facilita la tarea del modelo ARIMA. La gráfica como se ve en la Figura 30 muestra que la curva predicha sigue de cerca la evolución de los datos reales, con errores de menor magnitud que en el caso de la temperatura.

Respecto a la velocidad del viento según lo que muestra la Tabla 3 la variable de viento esta levemente por encima del 20 %, pero como ya se analizó se espera que haya mejoría con el paso del tiempo. En la gráfica como se vio en la Figura 31, el modelo logra anticipar las tendencias generales, pero le resulta más difícil predecir los picos y caídas súbitas propios de la variable velocidad del viento. Este resultado es esperable, dado que el viento es una variable muy sensible a factores de la zona y a eventos esporádicos.

Por último, para ver en temas generales cómo funciona el modelo de predicción ARIMA se hará con esta fórmula, donde se sacará el promedio general del ERM que los valores los tenemos en la Tabla 3.

$$ERM_{general} = \frac{14.63 + 11.49 + 24.83}{3} = \frac{50.95}{3} = 16.98\%$$

Con esto se puede decir que estos resultados muestran que el modelo ARIMA es una herramienta útil para pronosticar tendencias generales y obtener una referencia rápida sobre el comportamiento futuro de las variables meteorológicas en una zona determinada ya que da un valor del 16.98 % que se por debajo del 20% reflejando que se encuentra en un buen rango de precisión. Sin embargo, para aplicaciones donde se requiera anticipar con precisión eventos extremos o cambios repentinos es necesario complementar otros modelos con otros enfoques. Por ejemplo, utilizar modelos con redes neuronales o modelos estadísticos como SARIMA que puede entrenarse especificando la estacionalidad y tendencia en determinados periodos del año, lo que permite capturar de forma más adecuada las variaciones propias de las estaciones

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- El modelo ARIMA logra capturar la tendencia general de las variables meteorológicas analizadas, especialmente en temperatura y humedad, mostrando que es adecuado para predicciones a corto plazo en la estación local.
- La predicción de la humedad es la más precisa, con el menor error relativo (11.49%), lo que sugiere que esta variable presenta un comportamiento más predecible y estable en la zona estudiada.
- La predicción de la velocidad del viento presenta la mayor dificultad, alcanzando un error relativo del 24.83%. Esto se debe a la alta variabilidad e imprevisibilidad del viento, influenciado por factores locales y picos esporádicos.
- El modelo muestra limitaciones al anticipar cambios bruscos o eventos extremos en las variables particularmente en la temperatura y el viento, reflejando la naturaleza suavizadora de los modelos ARIMA sobre series temporales complejas.

5.2 Recomendaciones

- Complementar el modelo con técnicas adicionales como redes neuronales, modelos híbridos o variables exógenas, para mejorar la precisión en la predicción de eventos extremos y variabilidad diaria.
- Actualizar periódicamente el historial de datos para que el modelo se mantenga adaptado a los cambios climáticos recientes y evite la obsolescencia de patrones pasados.
- Incorporar métricas adicionales de evaluación (como MAE, MAPE o análisis de residuales) y realizar validaciones cruzadas para obtener una visión más completa del desempeño del modelo.
- Realizar estudios específicos sobre microclimas locales y factores externos que puedan estar afectando las mediciones con el fin de ajustar el modelo y aumentar la precisión en predicciones puntuales**.**

BIBLIOGRAFÍA

[1] M. Pilar, G. Casimiro, y P. G. Casimiro, *Análisis de series temporales: Modelos ARIMA*. 2023.

[2] Alvarez Corral Alejandro y Noguera González Daniel Francisco, «Monitorización y análisis de variables ambientales en tiempo real», jun. 2023.

[3] N. Proaño, «La transición entre la época seca y lluviosa se retrasó un mes en la Sierra, dice el Inamhi», https://www.metroecuador.com.ec/noticias/2024/10/22/la-transicion-entre-la-epoca-seca-y-lluviosa-se-retraso-un-mes-en-la-sierra-dice-el-inamhi/.

[4] MONTENEGRO AMBROSI LOURDES VALENTINA, «MODELO DE PREDICCIÓN DE PRECIPITACIÓN PARA ECUADOR CONTINENTAL BASADO EN ÍNDICES CLIMÁTICOS», 2024.

[5] Paulina Elizabeth López De La Vega, «Modelamiento espacial de la temperatura y humedad como parámetros de producción en el control de almacenaje del área de bodega en una industria farmacéutica para la identificación de puntos críticos», 2022.

[6] Gabriel Ojeda Suárez, «CUANTIFICACIÓN Y PREDICCIÓN DE LLUVIA A PARTIR DE INFORMACIÓN OBTENIDA DE RADIOENLACES DE MICROONDAS Y COMUNICACIONES SATELITALES APLICANDO MACHINE LEARNING», 2023.

[7] Garcés Palacios Luis Enrique y Jarrin Buenaño Carlos Fernando, «DISEÑO E IMPLEMENTACIÓN DE UNA ESTACIÓN METEOROLÓGICA MÓVIL PARA LA OBTENCIÓN DE DATOS MEDIOAMBIENTALES EN ESCENARIOS CONTROLADOS EN LA PROVINCIA DE CHIMBORAZO», 2019.

[8] Sosa Jose, Eusebio Roque, y Palomino Rubén, «Modelo de pronóstico ARIMA en el monitoreo de parametros ambientales», *Cienc Desarro*, vol. 21, n.º 2, p. 49, dic. 2018, doi: 10.21503/cyd.v21i2.1631.

[9] C. Campella, B. Cerne, y P. Salio, «Estación meteorológica», 2020.

[10] J. A. Simmonds, «Diseño, Construcción e Implementación De Una Estación Meteorológica Portátil IoT Usando ESP8266», 2020.

[11] A. Tobajas Garcia, «Diseño e implementación de una estación meteorológica con Raspberry Pi», 2021.

[12] Trewin. Blair, «Directrices de la Organización Meteorológica Mundial sobre el cálculo de las normales climáticas», 2017.

[13] A. Acevado, S. Torrent, S. Alvear, C. González, y M. A. Bustos, «MANUAL DE AGROMETEOROLOGÍA», 2022. [En línea]. Disponible en: https://climatologia.meteochile.gob.cl/

[14] A. S. B. Campoverde, «Análisis de la isla de calor urbana en el entorno andino de Cuenca-Ecuador», *Investigaciones Geograficas*, n.º 70, pp. 167-179, jul. 2018, doi: 10.14198/INGEO2018.70.08.

[15] S. Horacio E., «INTRODUCCIÓN a la METEOROLOGÍA GENERAL», 2022.

[16] «Anemómetro ultrasónico WUS4423 IED», https://www.meteo-shopping.com/es/viento/780-anemometro-ultrasonico-wus4423-ied.html.

[17] «Radiómetro Ultravioleta», https://www.quimisur.com/es/sensores/radiacion/radiometro-uv.html.

[18] «Sensor de temperatura y de humedad para estación meteorológica by METEO OMNIUM | DirectIndustry», https://www.directindustry.es/prod/meteo-omnium/product-72296-1647325.html.

[19] Spoch, «Red Meteorológica», https://ceaa.espoch.edu.ec/redEstaciones/index.php/informacion/sensores.

[20] «Anemómetro/veleta fijo para meteorologia LSI LASTEM», https://proviento.com.ec/anemometros/95-anemometroveleta-fijo-para-meteorologia-lsi-lastem.html.

[21] «Sensor De Radiación Solar » COTECNO | Equipamiento Científico | Prospecciones, Auscultación, Geofísica, Ingeniería», https://www.cotecno.cl/sensor-de-radiacion-solar/.

[22] Z. M. Mohammed Amin y S. A. Khaleefa Ali, «Climate change prediction using artificial neural network», 10 de enero de 2022, *Institute of Physics*. doi: 10.1088/1755-1315/961/1/012003.

[23] Tibshirani Ryan, «Lecture 6: Autoregressive Integrated Moving Average Models», 2023.

[24] M. Zhang, «Time Series: Autoregressive models AR, MA, ARMA, ARIMA», 2018.

[25] R. Graf y P. Aghelpour, «Daily river water temperature prediction: A comparison between neural network and stochastic techniques», *Atmosphere (Basel)*, vol. 12, n.º 9, sep. 2021, doi: 10.3390/atmos12091154.

[26] J. A. Mauricio, «Análisis de Series Temporales», 2007.

[27] «Contabo VPS», https://contabo.com/es/.

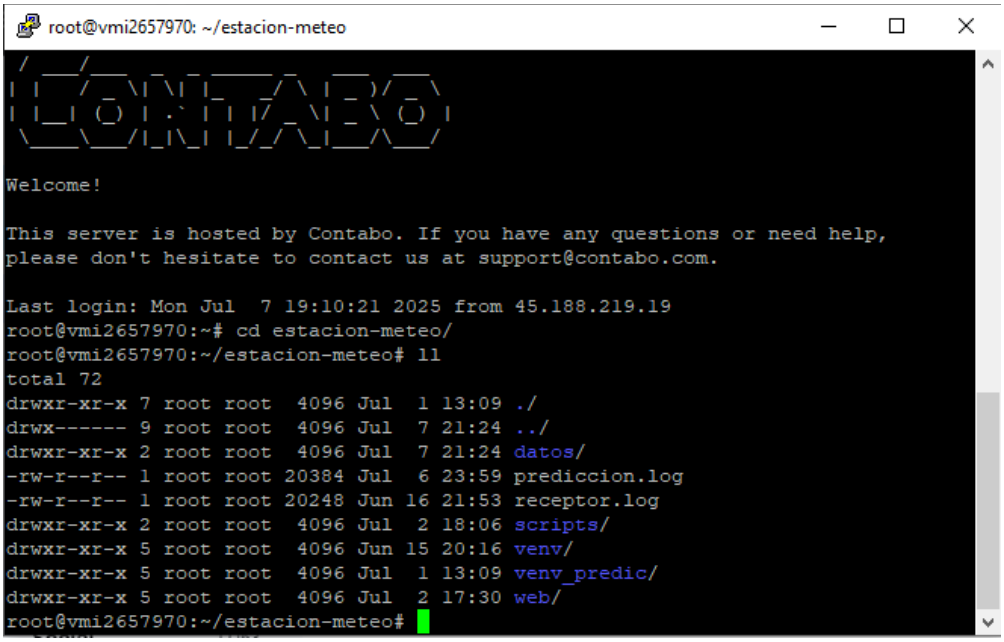
[28] A. Downey y J. Elkner, *Aprenda a Pensar Como un Programador con Python*. Green Tea Press, 2022.

[29] J. A. Fisteus, «Desarrollo Web con Flask», 2023.

[30] «Meteum », https://meteum.ai/meteum.

ANEXOS

Anexo 1. Estructura del servidor Ubuntu



Anexo 2. Script de la recepción de datos

```python
from flask import Flask, request
from datetime import datetime, timedelta
import csv
import os
import json

app = Flask(__name__)

# Ruta absoluta hacia ../datos
BASE_DATOS = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'datos'))

def grados_a_cardinal(grados):
    direcciones = [
        "N", "NNE", "NE", "ENE", "E", "ESE", "SE", "SSE",
        "S", "SSO", "SO", "OSO", "O", "ONO", "NO", "NNO"
    ]
    idx = int((grados + 11.25) / 22.5) % 16
    return direcciones[idx]

@app.route('/api/datos', methods=['POST'])
def recibir_datos():
    data = request.form.to_dict()

    # Obtener fecha UTC desde la estación
    fecha_utc_str = data.get("dateutc")
    if fecha_utc_str:
        fecha_utc = datetime.strptime(fecha_utc_str, "%Y-%m-%d %H:%M:%S")
        fecha_local = fecha_utc - timedelta(hours=5)  # Ecuador
    else:
        fecha_local = datetime.now()

    # === Extracción y conversión de datos ===
    temperatura_f = float(data.get("tempf", 0))
    humedad = float(data.get("humidity", 0))
    velocidad_mph = float(data.get("windspeedmph", 0))
    direccion_viento = float(data.get("winddir", 0))

    # Extraer lluvia desde los campos piezoeléctricos
    lluvia_hora_pulg = float(data.get("hrain_piezo", 0))
```

```
    lluvia_dia_pulg = float(data.get("drain_piezo", 0))
    lluvia_instantanea_pulg = float(data.get("rrain_piezo",0))
    lluvia_hora = round(lluvia_hora_pulg * 25.4, 2)  # mm
    lluvia_dia = round(lluvia_dia_pulg * 25.4, 2)    # mm
    lluvia_instantanea = round(lluvia_instantanea_pulg * 25.4, 2)

    presion_rel = float(data.get("baromrelin", 0))
    presion_abs = float(data.get("baromabsin", 0))
    uv = float(data.get("uv", 0))
    luz = float(data.get("solarradiation", 0))

    # Conversión de unidades
    temperatura_c = round((temperatura_f - 32) * 5/9, 1)
    velocidad_kmh = round(velocidad_mph * 1.60934, 2)

    # Mostrar en consola
    print(f"[{fecha_local.strftime('%Y-%m-%d %H:%M:%S')}] T={temperatura_c}°C, H={humedad}%, V={velocidad_kmh}km/h,

    # === Guardar en CSV ===
    os.makedirs(BASE_DATOS, exist_ok=True)
    archivo_csv = os.path.join(BASE_DATOS, "datos_actuales.csv")
    archivo_nuevo = not os.path.isfile(archivo_csv)
     with open(archivo_csv, "a", newline="") as f:
        writer = csv.writer(f)
        if archivo_nuevo:
            writer.writerow([
                "fecha", "temperatura", "humedad", "velocidad_aire", "direccion_aire",
                "lluvia_hora", "lluvia_dia", "presion_rel", "presion_abs", "uv", "luz"
            ])
        writer.writerow([
            fecha_local.strftime("%Y-%m-%d %H:%M:%S"),
            temperatura_c, humedad, velocidad_kmh, direccion_viento,
            lluvia_hora, lluvia_dia, presion_rel, presion_abs, uv, luz
        ])

    # === Guardar JSON para frontend ===
    ultimo_dato = {
        "fecha": fecha_local.strftime("%Y-%m-%d %H:%M:%S"),
        "temperatura": temperatura_c,
```

```
        "humedad": humedad,
        "viento_vel": velocidad_kmh,
        "viento_dir": f"{direccion_viento:.0f}° {grados_a_cardinal(direccion_viento)}",
        "lluvia_instantanea": lluvia_instantanea,
        "lluvia_hora": lluvia_hora,
        "lluvia_dia": lluvia_dia,
        "presion_rel": presion_rel,
        "presion_abs": presion_abs,
        "uv": uv,
        "luz": luz
    }

    archivo_json = os.path.join(BASE_DATOS, "ultimos_datos.json")
    with open(archivo_json, "w") as jsonfile:
        json.dump(ultimo_dato, jsonfile)

    return "OK", 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Anexo 3. Script de predicción del modelo ARIMA

```python
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
import datetime
import os
import csv
import json
import random

# ARCHIVOS DE ENTRADA Y SALIDA
BASE_DIR = os.path.dirname(__file__)
archivo_historial = os.path.abspath(os.path.join(BASE_DIR, "historial.csv"))
archivo_actual = os.path.abspath(os.path.join(BASE_DIR, "..", "datos", "datos_actuales.csv"))
archivo_csv = os.path.abspath(os.path.join(BASE_DIR, "..", "datos", "prediccion_semanal.csv"))
archivo_json = os.path.abspath(os.path.join(BASE_DIR, "..", "datos", "prediccion_2semanas.json"))
archivo_hist_pred = os.path.abspath(os.path.join(BASE_DIR, "..", "datos", "historial_predicciones.csv"))

# CARGA Y PREPROCESAMIENTO
# Historial
df_hist = pd.read_csv(archivo_historial, parse_dates=["time_iso"])
df_hist = df_hist.rename(columns={
    "time_iso": "fecha",
    "temperature": "temperatura",
    "humidity": "humedad",
    "wind_speed": "velocidad_aire"
})

# Datos actuales
df_actual = pd.read_csv(archivo_actual, parse_dates=["fecha"])

# Unir y limpiar
df = pd.concat([df_hist, df_actual], ignore_index=True)
df = df.sort_values("fecha")
df = df.drop_duplicates(subset="fecha")
df = df[["fecha", "temperatura", "humedad", "velocidad_aire"]].dropna()

# Agrupar por día y obtener promedios diarios
df['solo_dia'] = pd.to_datetime(df['fecha']).dt.date
df_dia = df.groupby('solo_dia')[['temperatura', 'humedad', 'velocidad_aire']].mean().reset_index()
df_dia = df_dia.sort_values("solo_dia")
```

```python
df_dia = df_dia.set_index("solo_dia")

# FUNCIÓN DE PREDICCIÓN ARIMA GENERAL
def predecir_arima(serie, pasos=14, orden=(5, 1, 2)):
    modelo = ARIMA(serie, order=orden)
    modelo_entrenado = modelo.fit()
    pred = modelo_entrenado.forecast(steps=pasos)
    return [round(p, 2) for p in pred]

#  FECHAS DE PREDICCIÓN
pasos = 14
fecha_inicio = df_dia.index[-1] + datetime.timedelta(days=1)
fechas_pred = [fecha_inicio + datetime.timedelta(days=i) for i in range(pasos)]

# PREDICCIONES
pred_temp = predecir_arima(df_dia["temperatura"], pasos, orden=(5, 1, 2))
pred_hum = predecir_arima(df_dia["humedad"], pasos, orden=(5, 1, 2))
pred_viento = predecir_arima(df_dia["velocidad_aire"], pasos, orden=(3, 0, 2))
pred_viento = [round(max(x * 2.0 + random.uniform(-1, 1), 3.0), 2) for x in pred_viento]
#  MOSTRAR EN CONSOLA 7 días
print("\n Predicción semanal:\n")
print("Fecha       | Temp (°C) | Humedad (%) | Viento (km/h)")
print("-" * 50)
for i in range(7):
    fecha = fechas_pred[i].strftime("%d/%m/%y")
    print(f"{fecha:11} | {pred_temp[i]:9} | {pred_hum[i]:12} | {pred_viento[i]:13}")

# GUARDAR EN CSV 7 días
os.makedirs(os.path.dirname(archivo_csv), exist_ok=True)
with open(archivo_csv, "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["fecha", "temperatura", "humedad", "velocidad_aire"])
    for i in range(7):
        writer.writerow([
            fechas_pred[i].strftime("%Y-%m-%d"),
            pred_temp[i],
            pred_hum[i],
            pred_viento[i]
```

```python

# GUARDAR EN 14 días
datos_json = []
for i in range(pasos):
    datos_json.append({
        "fecha": fechas_pred[i].strftime("%Y-%m-%d"),
        "temperatura": pred_temp[i],
        "humedad": pred_hum[i],
        "velocidad_aire": pred_viento[i]
    })

with open(archivo_json, "w", encoding="utf-8") as f_json:
    json.dump(datos_json, f_json, indent=2)

# GUARDAR EN HISTORIAL PREDICCIONES
fecha_generacion = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
nueva_entrada = []
for i in range(pasos):
    nueva_entrada.append([
        fecha_generacion,
        fechas_pred[i].strftime("%Y-%m-%d"),
        pred_temp[i],
        pred_hum[i],
        pred_viento[i]
    ])
archivo_nuevo = not os.path.exists(archivo_hist_pred) or os.stat(archivo_hist_pred).st_size == 0
with open(archivo_hist_pred, "a", newline="") as f_hist:
    writer = csv.writer(f_hist)
    if archivo_nuevo:
        writer.writerow(["generado", "fecha", "temperatura", "humedad", "velocidad_aire"])
    writer.writerows(nueva_entrada)

# RESUMEN FINAL
print("\n✅ Archivos guardados en:")
print(f"   - CSV (7 días): {archivo_csv}")
print(f"   - JSON (14 días): {archivo_json}")
print(f"   - Historial acumulativo: {archivo_hist_pred}")
```

Anexo 4. Script de AIC

AIC.py * ×

```python
import pandas as pd
import warnings
from statsmodels.tsa.arima.model import ARIMA
import datetime

warnings.filterwarnings("ignore")

archivo_historial = r"[redacted]\historial.csv"
archivo_actual = r"C:[redacted]\datos_actuales.csv"

# Cargar y preparar datos completos
df_hist = pd.read_csv(archivo_historial, parse_dates=["time_iso"])
df_hist = df_hist.rename(columns={
    "time_iso": "fecha",
    "temperature": "temperatura",
    "humidity": "humedad",
    "wind_speed": "velocidad_aire"
})
df_actual = pd.read_csv(archivo_actual, parse_dates=["fecha"])

df = pd.concat([df_hist, df_actual], ignore_index=True)
df = df.sort_values("fecha").drop_duplicates(subset="fecha")

# Parámetros a probar
p_values = [1, 2, 3, 4, 5]
d_values = [0, 1]
q_values = [0, 1, 2]

def buscar_mejor_arima(serie, nombre_var):
    mejor_aic = float("inf")
    mejor_orden = None
    resultados = []

    print(f"\nIniciando búsqueda ARIMA para {nombre_var} con {len(serie)} datos...")

    for p in p_values:
        for d in d_values:
            for q in q_values:
                try:
                    print(f"Probando ARIMA({p},{d},{q}) para {nombre_var}...")
```

Anexo 5. Visualización de los resultados en la página web

Anexo 6. Script para hacer Error Relativo Medio (Temperatura)

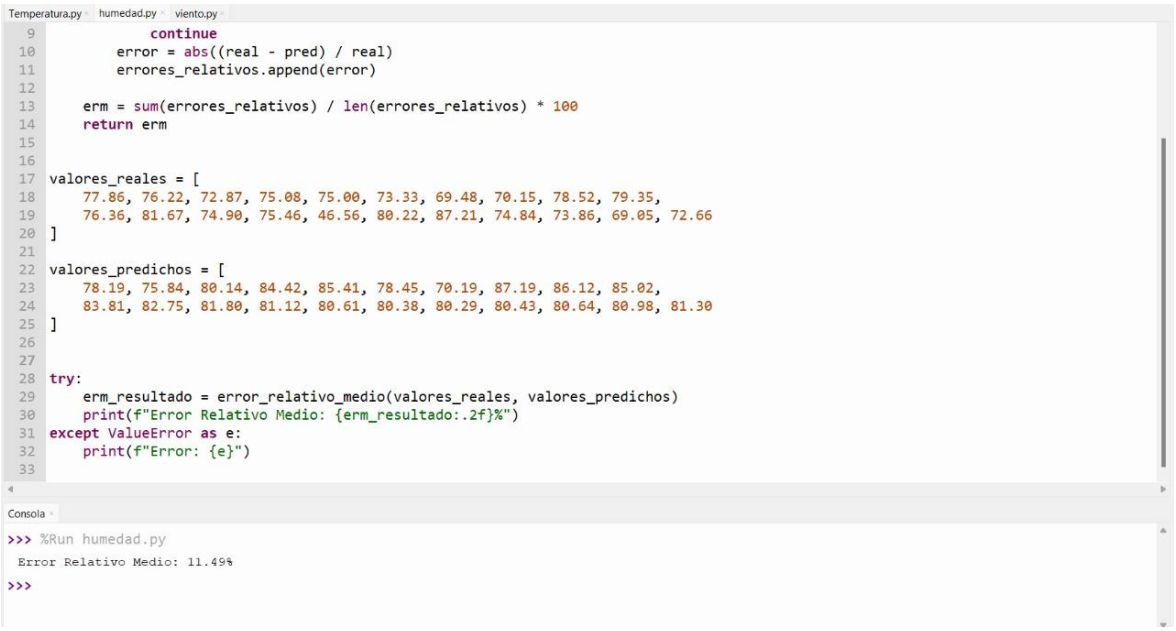
Temperatura.py humedad.py viento.py

```python
def error_relativo_medio(valores_reales, valores_predichos):
    if len(valores_reales) != len(valores_predichos):
        raise ValueError("Las listas deben tener la misma longitud.")

    errores_relativos = []
    for real, pred in zip(valores_reales, valores_predichos):
        if real == 0:
            print("Advertencia: se encontró un valor real igual a cero. Saltando...")
            continue
        error = abs((real - pred) / real)
        errores_relativos.append(error)

    erm = sum(errores_relativos) / len(errores_relativos) * 100
    return erm

valores_reales = [
    14.41, 15.04, 14.09, 13.73, 12.7, 12.74, 14.17, 14.78, 13.82, 13.47,
    13.47, 12.33, 12.83, 12.94, 13.2, 12.61, 11.56, 11.52, 10.84, 12.1, 11.53
]

valores_predichos = [
    15.00, 14.34, 14.34, 14.63, 14.82, 14.95, 15.04, 15.48, 15.69, 14.95,
    15.21, 15.46, 14.67, 13.84, 12.94, 14.86, 15.36, 15.96, 14.88, 13.79, 13.68
]
```

Consola

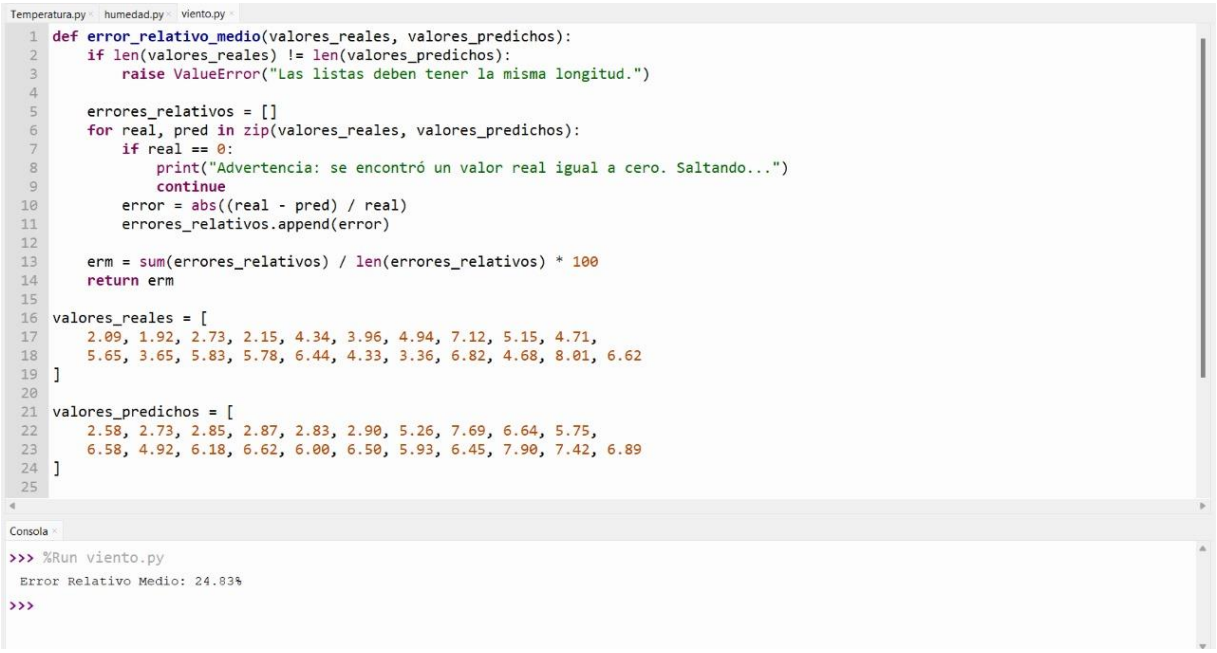
```
>>> %Run Temperatura.py
 Error Relativo Medio: 14.63%
>>>
```

Anexo 7. Script para hacer Error Relativo Medio (Humedad)



```
Temperatura.py  humedad.py  viento.py
            continue
        error = abs((real - pred) / real)
        errores_relativos.append(error)

    erm = sum(errores_relativos) / len(errores_relativos) * 100
    return erm


valores_reales = [
    77.86, 76.22, 72.87, 75.08, 75.00, 73.33, 69.48, 70.15, 78.52, 79.35,
    76.36, 81.67, 74.90, 75.46, 46.56, 80.22, 87.21, 74.84, 73.86, 69.05, 72.66
]

valores_predichos = [
    78.19, 75.84, 80.14, 84.42, 85.41, 78.45, 70.19, 87.19, 86.12, 85.02,
    83.81, 82.75, 81.80, 81.12, 80.61, 80.38, 80.29, 80.43, 80.64, 80.98, 81.30
]


try:
    erm_resultado = error_relativo_medio(valores_reales, valores_predichos)
    print(f"Error Relativo Medio: {erm_resultado:.2f}%")
except ValueError as e:
    print(f"Error: {e}")

Consola
>>> %Run humedad.py
 Error Relativo Medio: 11.49%
>>>
```

Anexo 8. Script para hacer Error Relativo Medio (Velocidad del viento)



```
Temperatura.py  humedad.py  viento.py
def error_relativo_medio(valores_reales, valores_predichos):
    if len(valores_reales) != len(valores_predichos):
        raise ValueError("Las listas deben tener la misma longitud.")

    errores_relativos = []
    for real, pred in zip(valores_reales, valores_predichos):
        if real == 0:
            print("Advertencia: se encontró un valor real igual a cero. Saltando...")
            continue
        error = abs((real - pred) / real)
        errores_relativos.append(error)

    erm = sum(errores_relativos) / len(errores_relativos) * 100
    return erm

valores_reales = [
    2.09, 1.92, 2.73, 2.15, 4.34, 3.96, 4.94, 7.12, 5.15, 4.71,
    5.65, 3.65, 5.83, 5.78, 6.44, 4.33, 3.36, 6.82, 4.68, 8.01, 6.62
]

valores_predichos = [
    2.58, 2.73, 2.85, 2.87, 2.83, 2.90, 5.26, 7.69, 6.64, 5.75,
    6.58, 4.92, 6.18, 6.62, 6.00, 6.50, 5.93, 6.45, 7.90, 7.42, 6.89
]

Consola
>>> %Run viento.py
 Error Relativo Medio: 24.83%
>>>
```

Anexo 9. Script del diseño de la página WEB index.hmtl

index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Estación Meteorológica</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">
    <style>
        body {
            background-color: #0f172a; /* Azul oscuro */
            color: white;
        }
        .cuadro {
            background-color: rgba(255, 255, 255, 0.1);
            padding: 1rem;
            border-radius: 0.5rem;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
        }
        .flecha {
            width: 100px;
            height: 100px;
            position: absolute;
            top: 0;
            left: 0;
            transform-origin: 50% 50%;
        }
    </style>
</head>
<body class="p-6 font-sans">

    <!-- ENCABEZADO CENTRADO -->
    <div class="flex flex-col md:flex-row justify-between items-center mb-8 text-center md:text-left">
        <div class="flex-1">
            <h1 class="text-3xl font-bold text-center">Universidad Nacional de Chimborazo</h1>
            <p class="text-xl text-center">Facultad de Ingeniería</p>
            <p class="text-lg text-center">Carrera de Telecomunicaciones</p>
            <p class="text-sm text-center mt-2">Autor: CARLOS LOPEZ</p>
        </div>
        <div class="w-24 h-24 mt-4 md:mt-0">
```

```
            <img src="{{ url_for('static', filename='logo2.png') }}" alt="Logo UNACH" class="w-full">
        </div>
    </div>

    <!-- DATOS ACTUALES Y GRÁFICAS -->
    <div class="grid grid-cols-1 md:grid-cols-3 gap-6">
        <div class="cuadro">
            <h2 class="text-lg font-bold">Temperatura</h2>
            <p class="text-2xl">{{ datos_actuales.temperatura }} °C</p>
            <canvas id="tempChart"></canvas>
        </div>
        <div class="cuadro">
            <h2 class="text-lg font-bold">Humedad</h2>
            <p class="text-2xl">{{ datos_actuales.humedad }} %</p>
            <canvas id="humChart"></canvas>
        </div>
        <div class="cuadro">
            <h2 class="text-lg font-bold">Vel. Viento</h2>
            <p class="text-2xl">{{ datos_actuales.viento_vel }} km/h</p>
            <canvas id="vientoChart"></canvas>
        </div>
    </div>
    <div class="mt-6 grid grid-cols-1 md:grid-cols-3 gap-6">

        <div class="cuadro">
            <h2 class="text-lg font-bold">Dirección del viento</h2>
            <p class="text-center mt-2 text-lg font-semibold">
                Dirección: {{ datos_actuales.viento_dir }}
            </p>
            <div class="relative w-32 h-32 mx-auto">
                <img src="{{ url_for('static', filename='brujula.png') }}" alt="Brújula"
                     class="absolute inset-0 w-full h-full pointer-events-none select-none">
                <div class="absolute inset-0 flex items-center justify-center"
                     style="transform: rotate({{ datos_actuales.viento_dir.split('°')[0] }}deg);">
                    <svg viewBox="0 0 100 100" width="56" height="56">
                        <polygon points="50,15 60,60 50,50 40,60" fill="red"/>
                    </svg>
                </div>
            </div>
        </div>

        <div class="cuadro">
            <h2 class="text-lg font-bold">Lluvia</h2>
            <p>Última hora: {{ datos_actuales.get('lluvia_hora', 0) }} mm</p>
            <p>Total diario: {{ datos_actuales.get('lluvia_dia', 0) }} mm</p>
            {% if datos_actuales.get('lluvia_hora', 0) > 0 %}
                <p class="text-red-400 font-semibold">      Está lloviendo</p>
            {% else %}
                <p class="text-green-400 font-semibold">☀ No está lloviendo</p>
            {% endif %}
        </div>

        <div class="cuadro">
            <h2 class="text-lg font-bold">Presión / UV / Luz</h2>
            <p>Presión relativa: {{ (datos_actuales.presion_rel * 33.8639) | round(2) }} hPa</p>
            <p>Presión absoluta: {{ (datos_actuales.presion_abs * 33.8639) | round(2) }} hPa</p>
            <p>Luz: {{ datos_actuales.luz }} W/m²</p>
            <p>Índice UV: {{ datos_actuales.uv }}</p>
            {% if datos_actuales.uv >= 8 %}
                <p class="text-red-400 font-semibold">¡Peligro! Índice UV muy alto. Protégete ◉</p>
            {% elif datos_actuales.uv >= 6 %}
                <p class="text-yellow-400 font-semibold">Precaución: Índice UV alto. Usa protección ☀</p>
            {% else %}
                <p class="text-green-400 font-semibold">Índice UV bajo. Sin riesgo ❀</p>
            {% endif %}
        </div>

        <div class="text-center mt-4">
            <p class="text-sm text-gray-300">Última actualización: {{ datos_actuales.fecha }}</p>
        </div>

    </div>

    <!-- TABLAS DE PREDICCIÓN -->
    <div class="mt-10">
        <h2 class="text-2xl font-bold text-center mb-4">▤ Predicción - Semana Actual</h2>
```

```
        <table class="table-auto mx-auto border-collapse border border-white">
            <thead>
                <tr>
                    <th class="border border-white px-4 py-2">Fecha</th>
                    <th class="border border-white px-4 py-2">Temp (°C)</th>
                    <th class="border border-white px-4 py-2">Humedad (%)</th>
                    <th class="border border-white px-4 py-2">Viento (km/h)</th>
                </tr>
            </thead>
            <tbody>
                {% for fila in semana_actual %}
                <tr>
                    <td class="border border-white px-4 py-1">{{ fila.fecha }}</td>
                    <td class="border border-white px-4 py-1">{{ fila.temperatura }}</td>
                    <td class="border border-white px-4 py-1">{{ fila.humedad }}</td>
                    <td class="border border-white px-4 py-1">{{ fila.velocidad_aire }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>

        <h2 class="text-2xl font-bold text-center mt-8 mb-4">📅 Predicción - Semana Siguiente</h2>
        <table class="table-auto mx-auto border-collapse border border-white">
            <thead>
                <tr>
                    <th class="border border-white px-4 py-2">Fecha</th>
                    <th class="border border-white px-4 py-2">Temp (°C)</th>
                    <th class="border border-white px-4 py-2">Humedad (%)</th>
                    <th class="border border-white px-4 py-2">Viento (km/h)</th>
                </tr>
            </thead>
            <tbody>
                {% for fila in semana_siguiente %}
                <tr>
                    <td class="border border-white px-4 py-1">{{ fila.fecha }}</td>
                    <td class="border border-white px-4 py-1">{{ fila.temperatura }}</td>
                    <td class="border border-white px-4 py-1">{{ fila.humedad }}</td>
                    <td class="border border-white px-4 py-1">{{ fila.velocidad_aire }}</td>
```

```
            data: {
                labels: labels,
                datasets: [
                    {
                        label: 'Viento real (km/h)',
                        data: vientoReal,
                        borderColor: 'blue',
                        fill: false,
                        tension: 0.3,
                        pointRadius: 4,
                        pointBackgroundColor: 'blue'
                    },
                    {
                        label: 'Predicción (km/h)',
                        data: vientoPred,
                        borderColor: 'red',
                        fill: false,
                        borderDash: [5,5],
                        tension: 0.3,
                        pointRadius: 4,
                        pointBackgroundColor: 'red'
                    }
                ]
            },
            options: {
                scales: { y: { beginAtZero: false } }
            }
        });
    </script>

    <!-- AUTO REFRESH -->
    <script>
        setInterval(() => {
            location.reload();
        }, 60000); // Recargar cada 60 segundos
    </script>

</body>
</html>
```

Anexo 10. Script ESP32 para mandar datos

index.html × esp32.py * ×

```python
import network
import time
import urequests
from machine import Pin

SSID = 'INGENIERIA'
PASSWORD = 'INGENIERIA2025'

SERVER_URL = 'http://82.197.67.139/datos'

# === CONFIGURACIÓN DE SENSORES ===
PLUVIOMETRO_PIN = 14   # GPIO donde está conectado el pluviómetro
ANEMOMETRO_PIN = 27    # GPIO del anemómetro
MM_POR_PULSO = 0.2      # Valor de lluvia por pulso del pluviómetro
VIENTO_FACTOR = 2.4     # Ejemplo: 2.4 km/h por pulso (ajustar según el sensor)

# === VARIABLES GLOBALES ===
conteo_lluvia = 0
conteo_viento = 0

# === FUNCIONES DE INTERRUPCIÓN ===
def contar_lluvia(pin):
    global conteo_lluvia
    conteo_lluvia += 1

def contar_viento(pin):
    global conteo_viento
    conteo_viento += 1

# === CONECTAR A WIFI ===
def conectar_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(SSID, PASSWORD)

    print("Conectando a WiFi...", end='')
    while not wlan.isconnected():
        print(".", end='')
        time.sleep(1)
    print("\nConectado:", wlan.ifconfig())
```

```
        print(".", end='')
        time.sleep(1)
    print("\nConectado:", wlan.ifconfig())

# === ENVIAR DATOS AL SERVIDOR ===
def enviar_datos(lluvia_mm, viento_kmh):
    try:
        datos = {
            'lluvia': lluvia_mm,
            'viento': viento_kmh
        }
        respuesta = urequests.post(SERVER_URL, json=datos)
        print("Datos enviados:", datos)
        print("Respuesta del servidor:", respuesta.text)
        respuesta.close()
    except Exception as e:
        print("Error al enviar datos:", e)

# === CONFIGURAR PINES ===
pluvio_pin = Pin(PLUVIOMETRO_PIN, Pin.IN, Pin.PULL_UP)
pluvio_pin.irq(trigger=Pin.IRQ_FALLING, handler=contar_lluvia)

anemo_pin = Pin(ANEMOMETRO_PIN, Pin.IN, Pin.PULL_UP)
anemo_pin.irq(trigger=Pin.IRQ_FALLING, handler=contar_viento)

# === PROGRAMA PRINCIPAL ===
conectar_wifi()

while True:
    conteo_lluvia = 0
    conteo_viento = 0
    time.sleep(60)  # Espera 60 segundos

    # Calcular valores
    lluvia_mm = conteo_lluvia * MM_POR_PULSO
    viento_kmh = conteo_viento * VIENTO_FACTOR

    # Enviar al servidor
    enviar_datos(lluvia_mm, viento_kmh)

```