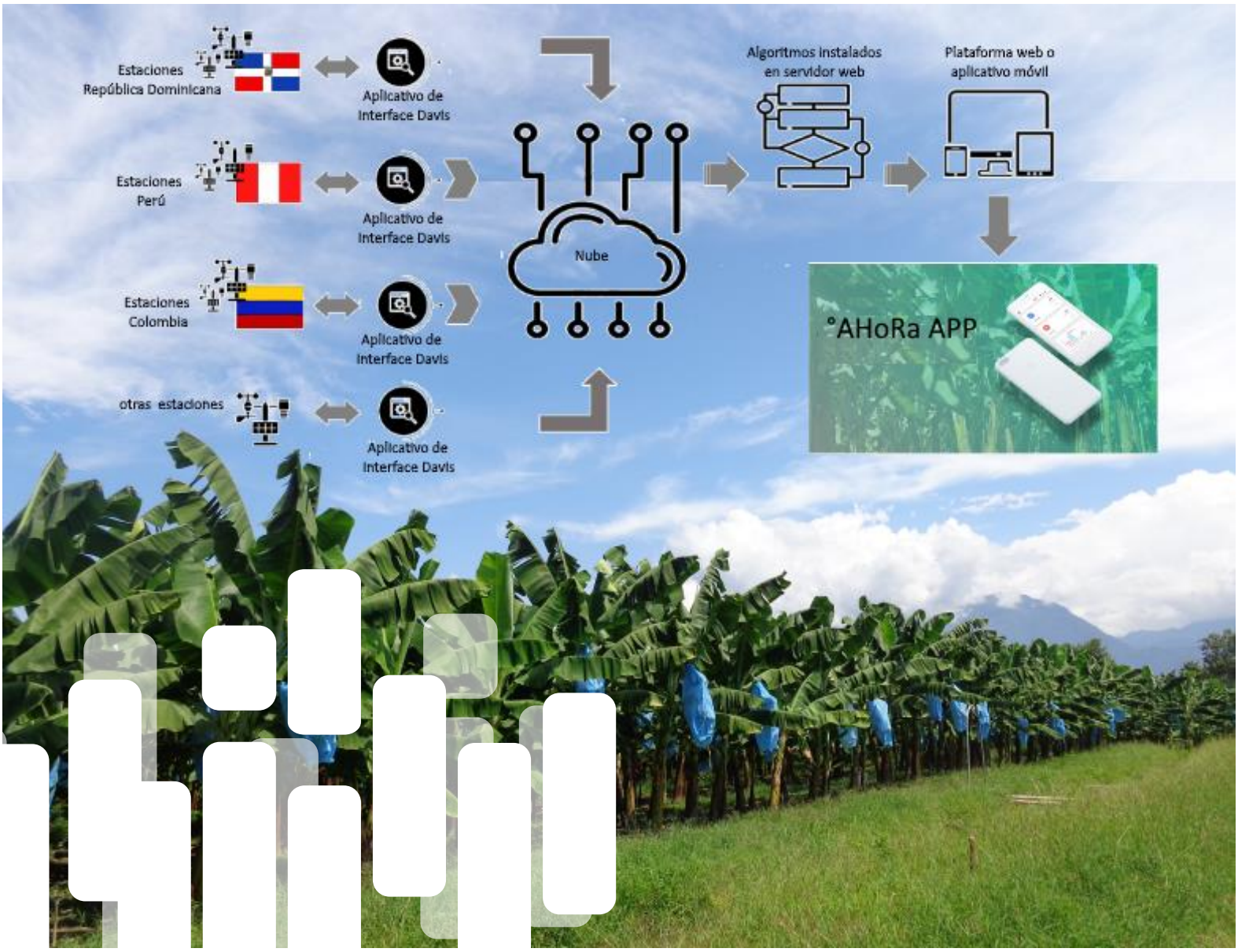


Consultoría para desarrollar un estudio de un aplicativo para productores familiares de musáceas

Producto 6. Monografía de diseño técnico de la aplicación

William Ipanaqué Alama, Iván Belupú, Carlos Estrada, Jorge Neyra Hau Yon, Jean Carlos Alexander Campos Cercado – **Universidad de Piura**

2021

FONTAGRO

Códigos JEL: Q16
ISBN: en trámite

FONTAGRO (Fondo Regional de Tecnología Agropecuaria) es un programa de cooperación administrado por el Banco Interamericano de Desarrollo (BID), pero con su propia membresía, estructura de gobernabilidad y activos. Las opiniones expresadas en esta publicación son de los autores y no necesariamente reflejan el punto de vista del Banco Interamericano de Desarrollo, FONTAGRO, de sus Directores Ejecutivos, ni de los países que representan.

El presente documento ha sido preparado por William Ipanaqué Alama, Iván Belupú, Carlos Estrada, Jorge Neyra Hau Yon, Jean Carlos Alexander Campos Cercado – Universidad de Piura.

Esta publicación puede solicitarse a:
FONTAGRO
Correo electrónico: fontagro@fontagro.org
www.fontagro.org

Tabla de Contenidos

Contenido

Resumen **5**

Introducción **6**

Objetivos **7**

Metodologia **8**

Resultados y Discusión **10**

I. Algoritmos de adquisición de datos **12**

1. Estación Caribia (Colombia) y Estación Asprobo (Perú) 12

1.1. Objetivos del desarrollo del algoritmo 12

1.2. Tipos de estaciones 12

1.3. Características estación Caribia - Colombia 12

1.4. Características estación ASPROBO – PERÚ 14

1.5. Desarrollo del algoritmo 15

2. Estaciones de Republica Dominicana 19

2.1. Objetivos del desarrollo del algoritmo 19

2.2. Características estaciones de Republica Dominicana 19

2.3. Desarrollo del algoritmo 21

II. Algoritmos de funcionamiento basados en indicadores atmosféricos **22**

3. Librerías pandas 22

3.1 Tipos de datos de Pandas 23

4. Algoritmos de funcionamiento 26

4.1. Tratamiento de Datos 26

4.2. Funcionalidades 30

4.2.1. Tasa potencial de crecimiento de hojas y gráficas. 30

4.2.2. Grados día para cosecha – Estimación de fecha de floración y gráfica 33

4.2.3. Grados día para cosecha – Estimación de fecha de cosecha y gráfica 34

4.2.4. Cálculo de biomasa – Estimación de la masa producida por racimo y por parcela 36

4.2.5. Cálculo del potencial de nutriente a reponer – Estimación de la masa nutrientes que se deben reponer por planta y parcela 38

4.2.6. Cálculo de la necesidad hídrica – Estimación del volumen de agua que se deben reponer por hectárea 40

III. Desarrollo de la aplicación web ahora app 43

5. ¿Qué es Flask? 43

6. Esquema de desarrollo 43

6.1. Vistas 44

6.2. Controlador 47

6.3. Modelo 48

7. Base de datos de la aplicación Web 51

7.1. Definición de base de datos 51

7.2. Ventajas del uso de base de datos 51

7.3. Diagrama entidad – relación para el sistema de la plataforma virtual 52

7.4. Descripción de las tablas de la base de datos diseñada 52

8. SERVIDOR WEB 53

Conclusiones: 54

Referencias Bibliográficas 55

Resumen

El aplicativo °AHoRa, que se está desarrollando, busca facilitar la incorporación de la variabilidad de las condiciones abióticas en la planificación y toma de decisiones de productores, técnicos y asociaciones de banano y plátano, permitiendo el ajuste en prácticas de manejo como el desmane, la proyección del periodo óptimo a cosecha de racimos y la identificación de condiciones particulares que causan el rechazo de fruta por defectos fisiológicos. Para el desarrollo de dicho aplicativo se han implementado cinco funciones (Tasa potencial de emisión de hojas, tasa potencial de desarrollo de racimo, peso potencial de racimo, demanda potencial de nutrientes y estimación de necesidad hídrica), cuya información debe ser mostrada en un entorno amigable y fácil de utilizar. Detrás de todo lo que se pueda visualizar del aplicativo, existen muchos algoritmos que hacen posible su funcionamiento sin necesidad de la intervención humana y es acerca de estos algoritmos lo que principalmente se desarrolla en este documento, desde la extracción de los datos de los servidores de Davis, su almacenamiento en una base de datos propia, el desarrollo de las funciones en Python hasta el desarrollo de la interfaz web.

Palabras Clave: Aplicativo web, algoritmos, adquisición de datos, servidores Davis, Smart farming.

Introducción

El cambio climático es uno de los problemas de mayor preocupación a nivel mundial que tiene efectos tanto directos como indirectos en la productividad agrícola, entre ellos, cambios en los regímenes pluviométricos, sequías, inundaciones y la redistribución geográfica de plagas y enfermedades (FAO, 2020). En particular en el cultivo de banano, la variación de los niveles óptimos de temperatura, precipitación y humedad relativa afectan el desarrollo del cultivo; alteraciones que impactan directamente en la productividad del cultivo, al afectar el desarrollo de la planta y del fruto (Távara, 2020).

Eventos como inundaciones y vientos huracanados dañan y destruyen plantaciones, mientras que eventos moderados como las olas de frio y calor, sequias fuera de temporada y lluvias excesivas atrasan o adelanta el ritmo productivo. Aunque los eventos extremos como inundaciones y vientos huracanados son incontrolables, la variabilidad moderada exige un ajuste en las prácticas de manejo. Sin embargo, muchos productores y técnicos de campo siguen las rutinas de prácticas culturales sin considerar la variabilidad climática.

Con el creciente conocimiento científico y los estudios ecofisiológicos se han obtenido modelos matemáticos en diferentes ambientes, los cuales describen la dinámica del crecimiento, desarrollo foliar y producción de racimos, siendo varios de ellos validados en nuevos ambientes. Algunos de los autores que han estudiado el efecto del cambio climático en el cultivo de banano son Guarín (2011), Higuera (2015), Yela et al. (2016) y Távara (2020). No obstante, a pesar de lo anterior, hasta la fecha no se evidencia una integración de estos conocimientos en una herramienta tecnológica útil para la toma de decisiones y en un insumo para los procesos de planificación del cultivo de banano y plátano, direccionado hacia la adaptación y la mitigación de los efectos de la creciente variabilidad climática.

Es por ello que con el proyecto “°AHoRa: Aplicativo para productores familiares de musáceas”, el cual se desarrolla en las regiones productoras de banano y plátano de Colombia, República Dominicana y Perú, se pretende diseñar un aplicativo que facilite la incorporación de la variabilidad de las condiciones abióticas en la planificación y toma de decisiones de los productores, técnicos y sus asociaciones de banano y plátano, mediante la implementación de cinco funciones (Tasa potencial de emisión de hojas, tasa potencial de desarrollo de racimo, peso potencial de racimo, demanda potencial de nutrientes y estimación de necesidad hídrica), permitiendo así el ajuste en prácticas de manejo como el desmane, la proyección del periodo óptimo para cosechar el racimo y la identificación de condiciones particulares que causan el rechazo de fruta por defectos fisiológicos. La institución ejecutora del proyecto es la Corporación Colombiana de Investigación Agropecuaria- Agrosavia, y las instituciones co-ejecutoras son la Universidad de PIURA, el Instituto Nacional de Innovación Agraria - INIA de Perú, y el Instituto Dominicano de Investigaciones Agropecuarias y Forestales - IDIAF.

El aplicativo denominado °AHoRa, operará con datos locales de estaciones meteorológicas

y de parcelas de productores, para ofrecer servicios y cubrir 20% del área de banano de exportación y plátano de la zona de prueba. El uso amplio del aplicativo por los productores familiares impulsará las siguientes mejoras en la gerencia del cultivo: 1) Mejor identificación de variaciones en comportamiento del cultivo frente a su potencial productivo; 2) Identificación de factores limitantes en tiempo real sujeto a cambios en gerencia; 3) Recomendaciones de prácticas de manejo como el desmane, a través de información en tiempo real de condiciones agroclimáticas y de crecimiento de fruto; 4) Proyecciones del volumen de fruta cosechada según fluctuación de variables meteorológicos; y 5) Identificación de condiciones abióticas especiales correlacionadas con defectos fisiológicos.

Con el fin de especificar y documentar la información técnica que sustenta el diseño y desarrollo de la aplicación °AHoRa, el presente documento describe los pasos y los algoritmos necesarios para la construcción, funcionamiento y programación de la aplicación en la interfaz web. Se describe el proceso desde la extracción de los datos meteorológicos de los servidores de Davis, su almacenamiento en una base de datos, la implementación de las cinco funciones en Python, hasta el desarrollo de la interfaz web. Esta información a futuro permitirá soportar mejoras, modificaciones o actualizaciones al sistema en general de la aplicación.

El presente diseño o manual técnico de la aplicación es complemento del documento titulado “Nota técnica sobre la plataforma de cálculos diseñada para generar indicadores de comportamiento de banano”, en el cual se detallan las cinco ecuaciones matemáticas que cuantifican la relación entre factores abióticos y el comportamiento del cultivo de banano – elaborado previamente a este-. Del mismo modo, es complemento de este documento el “Manual operativo de la aplicación en su versión Demo”, cuyo objetivo es instruir al usuario en el uso de la aplicación con elaboración posterior al presente documento. Teniendo en cuenta lo anterior, y con la finalidad de guardar consistencia y evitar redundar en los diferentes temas, las referencias o explicaciones someras serán suficientes cuando se requiera exponer o dilucidar conceptos o temas ya ilustrados previamente o que se van a abarcar en documentos posteriores.

Objetivos

Objetivo principal:

El objetivo de este documento es ilustrar e informar a las personas encargadas de mantener la prestación del servicio de la aplicación °AHoRa y en general al público interesado, acerca de los aspectos técnicos involucrados en el desarrollo de la aplicación, es decir, la estructura y conformación del sistema, con el fin de asegurar la transferencia de conocimiento y servir de soporte o insumo para realizar modificaciones o actualizaciones al sistema en general.

Objetivos específicos:

- Diseño y desarrollo de los algoritmos que actualizarán la data en tiempo real.
- Diseño y desarrollo de los algoritmos de pretratamiento.
- Diseño de la base de datos del aplicativo.
- Programación de las cinco ecuaciones o funciones que conforman el aplicativo.
- Diseño del Frond-end del aplicativo web.

Metodología

La construcción de la aplicación °AHoRa depende de dos insumos principales: 1. Los datos meteorológicos reportados por las estaciones ubicadas en cada uno de los países participantes en el proyecto, y 2. Las ecuaciones matemáticas que cuantifican la relación entre factores abióticos y el comportamiento de aspectos claves del cultivo de banano. Una vez se cuenta con estos dos aspectos, lo cuales se detallan a continuación, se procede a diseñar y programar el aplicativo mediante algoritmos – El detalle de este proceso se encuentra en la sección de Resultados y Discusión.

Estaciones meteorológicas

Cada uno de los países que integra el proyecto °AHoRa tiene estaciones meteorológicas instaladas en áreas específicas y de interés para este proyecto. Hasta la fecha todas son estaciones meteorológicas Davis, por lo que las estructuras de los algoritmos son similares.

Los datos son recopilados de cada estación por medio de solicitudes a los servidores de Davis (API REST) para posteriormente ser almacenados en una base de datos de la Universidad de Piura. Los datos que se obtienen del API REST de Davis están en formato JSON, pero el algoritmo es capaz de separar los datos e insertarlos en la base de datos (BD). Hasta la redacción de este documento se ha trabajado con una estación de Colombia (Estación Caribia), una estación de Perú (Estación ubicada en la asociación ASPROBO), y tres estaciones de Republica Dominicana (Estación Amina, Hatillo Palma y Hato al medio), no obstante, se está trabajando en la inclusión total de cinco estaciones meteorológicas más que han sido designadas para este proyecto.

Funciones del aplicativo

En la presente monografía se describe el diseño técnico general que se ha optado para la aplicación, en relación con el proyecto °AHoRa. Las ecuaciones o funciones que integra el aplicativo se muestran en la *Tabla 1*.

Tabla 1: Funciones que integran el aplicativo y que permiten cuantificar la relación entre factores abióticos y el comportamiento del cultivo de banano.

Indicador	Importancia	Descripción de Indicadores a desarrollar	Periodo de cálculo
Tasa potencial de emisión de hojas	Aspecto de crecimiento más sensible a condiciones abióticas (temperatura y agua) a corto plazo.	GDD= (temperatura media diaria del día - temperatura base de 13°C) 0 si valor es negativo o por temperatura >35°C. Potencial emisión de hojas =suma GDD del periodo/108 GDD para la emisión de una hoja.	14 o 28 días previo a la fecha del cálculo.
Tasa de desarrollo de racimo	Cosechar en un momento adecuado es importante en asegurar calidad de fruta durante el transporte al mercado	GDD= (temperatura media diaria del día - temperatura base de 13°C) 0 si valor es negativo o por temperatura >35°C. Cuantos días de hoy para atrás para acumular 900 GDD de encinte a cosecha	Calculado para periodo de encinte a momento óptimo de cosecha
Peso potencial de racimo	Integración de factores abióticos de floración a cosecha	GDD para atrás- fecha floración (900 GDD), Radiación acumulada de floración a presente y captada basado en ecuación Beer Lambert con coeficiente de extinción de 0.7 y IAF de 3.5, conversión de luz a biomasa 1,5 g MJ captada Convertir a biomasa/mata multiplicando por 5m2/mata (densidad de 2000 matas/ha) Toda biomasa acumulada de floración a cosecha es racimo	Periodo de desarrollo de racimo finalizando en cosecha
Demanda potencial de nutrientes	Plantación establecida representa un capital de nutrientes acumulados, pero cada semana nutriente son extraídos en racimos	El mismo cálculo de biomasa usando un periodo fijo de últimos dos meses para llegar a biomasa/mata: Para calcular nutrientes a compensar (sin tomar en cuenta lixiviación o otras perdidas), biomasa total*0.5 (% en racimo) *contenido de nutriente en biomasa (1% N y x% K)	Dos meses previos a la fecha actual

Balance hídrico básico	Asegurar humedad adecuada es elemento de manejo importante para productividad de campo de banano	Sumar EVTP diario para bloque 7 días = demanda agua de la planta; periodo de cálculo; Restar precipitación efectiva = déficit a cubrir; Incorporar eficiencia del sistema de riego	Calculo por semana en base a EVPo con opción de sumar por periodo deseado - 7, 14, 28 días

Resultados y Discusión

Para el cálculo de las cinco ecuaciones se requiere información que proviene de las diferentes centrales meteorológicas y sensores que se tienen en Colombia, Perú y República Dominicana, cercanas a las regiones productoras de banano de estos países. Se ha procurado que el sistema tenga la mínima intervención humana. Por lo que se evitará la digitalización por teclado. La idea es que la plataforma a diseñar tenga un programa que se comunique en forma automática con cada estación meteorológica, suba los datos a la nube y los preprocese. Con esta información otro algoritmo implementado en un programa calcula y los resultados son mostrados en la aplicación al usuario.

En la figura 1 se muestra cómo se ha dividido el primer diseño de la aplicación.

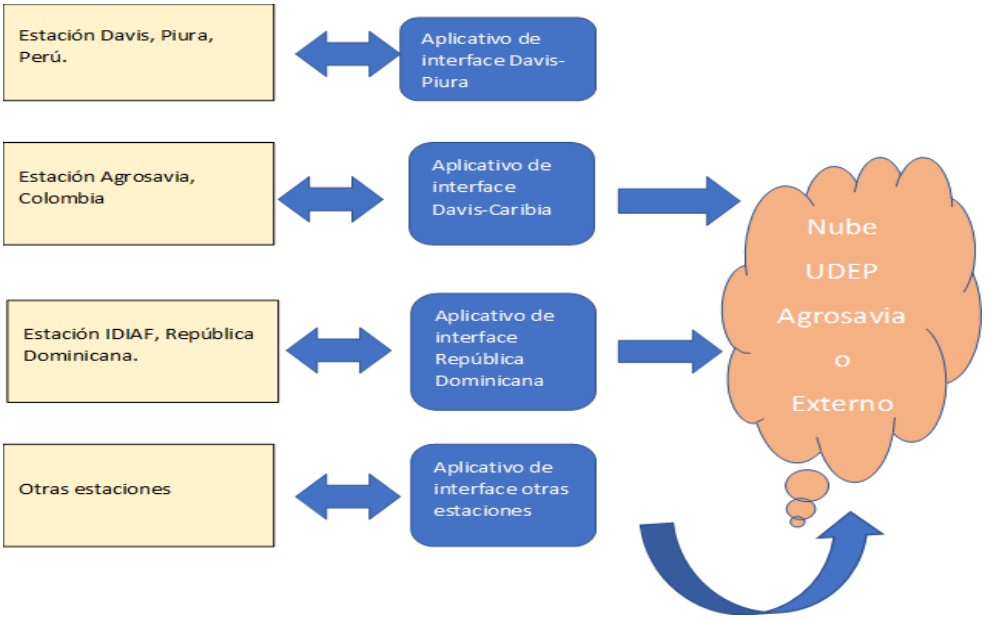


Figura 1: Esquema aplicativos de interfaces y estaciones meteorológicas

Esta primera parte tiene el objetivo de subir los datos de las diferentes estaciones mediante un aplicativo denominado “aplicativo de interface”. Según el tipo de estación se diseña un tipo de aplicativo, ya que éste deberá tomar en cuenta las características técnicas y de comunicación de cada estación en particular. Además, este aplicativo evitará el tener una intervención manual en este proceso, es decir se automatizará el subir los datos de las diferentes estaciones, hacer un pre-procesamiento y ponerlo en un formato adecuado, el cual se aloja en la nube. De esta manera la nube contiene la información necesaria para realizar todos los algoritmos o cálculos para que los productores tengan la información adecuada. La Figura 1 muestra, por ejemplo, un aplicativo de interface que se comunica con la correspondiente estación y esos datos los sube a la nube. Los algoritmos de estos aplicativos se describen en la Sección **Error! Reference source not found.** “**Error! Reference source not found.**”.

Una vez los datos están en la nube, se tienen otros algoritmos que realizan los cálculos y provee información en un dispositivo informático, tal como lo esquematiza la siguiente Figura 2:

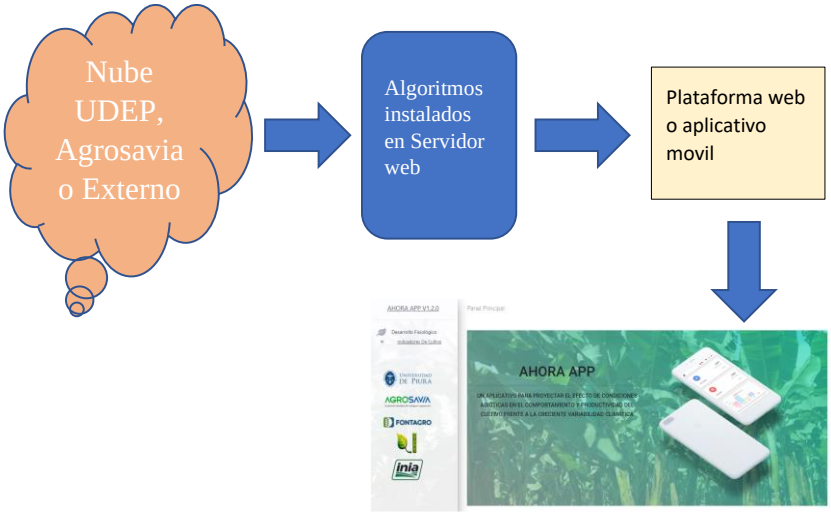


Figura 2: Esquema algoritmos de cálculo en servidor web

Los algoritmos de funcionamiento basado en indicadores atmosféricos se detallan en la Sección “**Error! Reference source not found.**”.

La aplicación web se ha desarrollado bajo el framework Flask de Python. Flask es un módulo de Python que permite desarrollar aplicaciones web de manera intuitiva. Tiene un núcleo pequeño y fácil de ampliar: es un microframework que no incluye un ORM (Object Relational Manager) o características similares. Entre sus características más importantes están el enrutamiento de URL, motor de plantillas y la aplicación web WSGI. Está basado en el patrón MVC (Modelo Vista Controlador) que es una manera de trabajar que permite diferenciar y separar lo que es el modelo de datos (los datos que van a tener la App que normalmente están guardados en la base de datos), la vista (páginas HTML) y el controlador

(donde se gestiona las peticiones de la app web). Los elementos que componen el backend y frontend de la aplicación web se detallan en la Sección "**Error! Reference source not found.**".

I. Algoritmos de adquisición de datos

1. Estación Caribia (Colombia) y Estación Asprobo (Perú)

1.1. Objetivos del desarrollo del algoritmo

- Solicitar datos a los servidores de Davis (API REST - Es una forma de que dos sistemas informáticos se comuniquen a través de HTTP).
- Extracción de todos los datos históricos.
- Almacenamiento de los datos extraídos en una base de datos (BD) propia.
- Funcionamiento en tiempo real sin necesidad de intervención humana.
- Capacidad de recuperar los datos cuando la estación suba la data varias horas después de su última actualización.

1.2. Tipos de estaciones

Son estaciones Davis que actualizan los datos cada hora en Colombia y cada 15 minutos en Perú. La estación de Colombia no cuenta con sensores de suelo por lo que variables como electro-conductividad, temperatura de suelo, y humedad del suelo no son tomadas en cuenta.

1.3. Características estación Caribia - Colombia

La estación mide los datos cada hora y por medio de un chip celular se conecta a internet y se suben los datos a los servidores de Davis. Los datos registrados son almacenados y mostrados en una página web (www.weatherlink.com) (Figura 3). Para acceder a los datos históricos se paga una cierta cantidad de dinero de manera anual. En total la estación mide 27 variables, que se muestran en la siguiente *Tabla 2*.

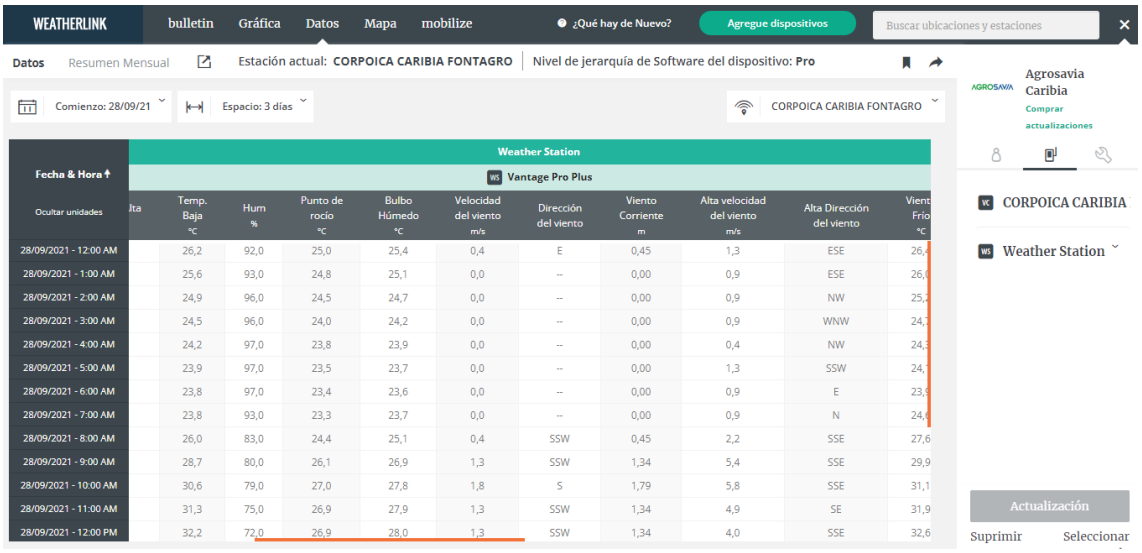


Figura 3: Visualización de algunas variables en Weatherlink

Tabla 2: Variables que se miden en la estación de Caribia – Colombia

	Variable	Unidades
1	Presión barométrica	mb
2	Temperatura	°C
3	Temperatura Alta	°C
4	Temperatura Baja	°C
5	Humedad	%
6	Punto de rocío	°C
7	Bulbo húmedo	°C
8	Velocidad del viento	m/s
9	Dirección del viento	--
10	Viento corriente	m
11	Alta velocidad del viento	m/s
12	Alta dirección del viento	--
13	Viento frio	°C
14	Índice de calor	°C
15	THW	°C
16	Lluvia	mm
17	Tasa de lluvia	mm/h
18	Evapotranspiración	mm
19	Grados día de calentamiento	--
20	Grados día de enfriamiento	--
21	THWS	°C
22	Radiación solar	W/m2

23	Energía solar	Ly
24	Radiación solar alta	W/m2
25	UV	--
26	Dosis UV	--
27	Alto índice de UV	--

1.4. Características estación ASPROBO – PERÚ

Los registros de los datos en la nube son en intervalos de 15 minutos y además de las variables de la tabla anterior se suman dos nodos, donde cada nodo mide las variables que se muestran en la siguiente tabla (*Tabla 3*).

Tabla 3. Variables que miden cada nodo de la estación en Perú

	Sensor Microclima	Unidades
1	Temperatura Baja	°C
2	Temperatura	°C
3	Temperatura Alta	°C
4	Humedad Baja	%
5	Humedad	%
6	Humedad Alta	%
7	Punto de rocío	°C
8	Bulbo húmedo	°C
9	Índice de calor	°C
	Sensor de suelo	
10	Temperatura de Suelo	°C
11	Humedad de suelo	%
12	EC de suelo	dS/m

Los clientes de Davis tienen acceso a los datos mediante Weatherlink y también mediante una API. La ventaja de la API es su facilidad de trabajar en línea con los datos, siendo esto importante para el desarrollo de algoritmos de machine learning, inteligencia artificial, etc. (Figura 4).

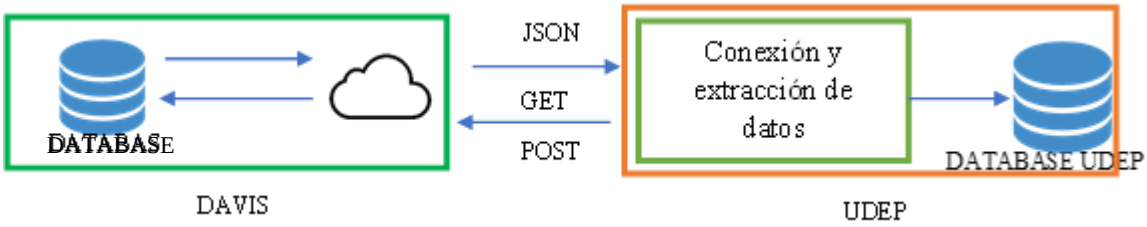


Figura 4: Diagrama comunicación Davis-UDEP

1.5. Desarrollo del algoritmo

Para el acceso a los datos históricos mediante la API de Davis es necesario el uso de las credenciales como “Api key”, “Api secret”, “Station ID”, siendo los dos primeros, valores generados en la página web de Weatherlink y el tercero es una identificación única de la estación. Todos estos datos son accesibles para los clientes de Davis por lo que cualquier cliente puede acceder a sus datos mediante el uso de la API.

El diagrama de flujo (Figura 5), resume de manera general la forma en la que se extraen los datos y su posterior almacenamiento, este algoritmo se puede replicar para todas las estaciones de Davis.

1.5.1 Algoritmo principal

- El algoritmo se ejecuta en este caso a los 15 minutos de cada hora (la estación sube los datos a la nube en intervalos de 1 hora). El valor de 15 minutos es referencial y se ha usado para las pruebas. Los datos en la web de Weatherlink se actualizan en la mayoría de los casos a los pocos segundos o minutos por lo que el valor se puede ajustar.
- Cuando se cumplen los 15 minutos de cualquier hora se envía una solicitud a la API de Davis para la obtención del último dato registrado. Para las solicitudes se necesita un rango de tiempo, un tiempo inicial y un tiempo final, para el tiempo final se utiliza el tiempo en el que se envía la solicitud y como tiempo inicial se le resta al tiempo anterior una hora quedando así un rango donde el único dato existente en la base de datos es el último registrado.
- La respuesta de la API es un formato JSON y puede tener datos, como también estar vacío debido a que la estación puede presentar algún problema y no se registran los valores de esa hora. Si fuese este el caso el algoritmo regresa a la espera de la siguiente hora.
- Si el JSON contiene los datos se procede a leer solo la fecha (Esta en formato UNIX), y a la vez se extrae la fecha del último registro en la base de datos.
- Si los tiempos anteriores son iguales, entonces quiere decir que esos datos ya existen por lo que no se graban y de esa forma se evita tener datos duplicados en la base de datos.

- Cuando las fechas son distintas se restan y se compara esa diferencia (se restan aprovechando el formato UNIX cuya unidad está en segundos).
- Una hora equivale a 3600 segundos así que si la diferencia es ese valor indica que los nuevos datos del JSON son de una hora después, lo que indicaría un correcto funcionamiento de la estación y se procedería a grabar dichos datos.
- Si la diferencia es un número mayor a 3600 entonces indicaría que en ese lapso no se han actualizado los datos en la BD, por fallas en la estación, por perdida de la red, etc. En este caso se ha implementado un algoritmo secundario que se encarga de actualizar la BD con esos valores que no se registraron en su momento.
- El algoritmo es aplicable para toda estación Davis, en este caso es similar para las estaciones de Colombia y Perú.

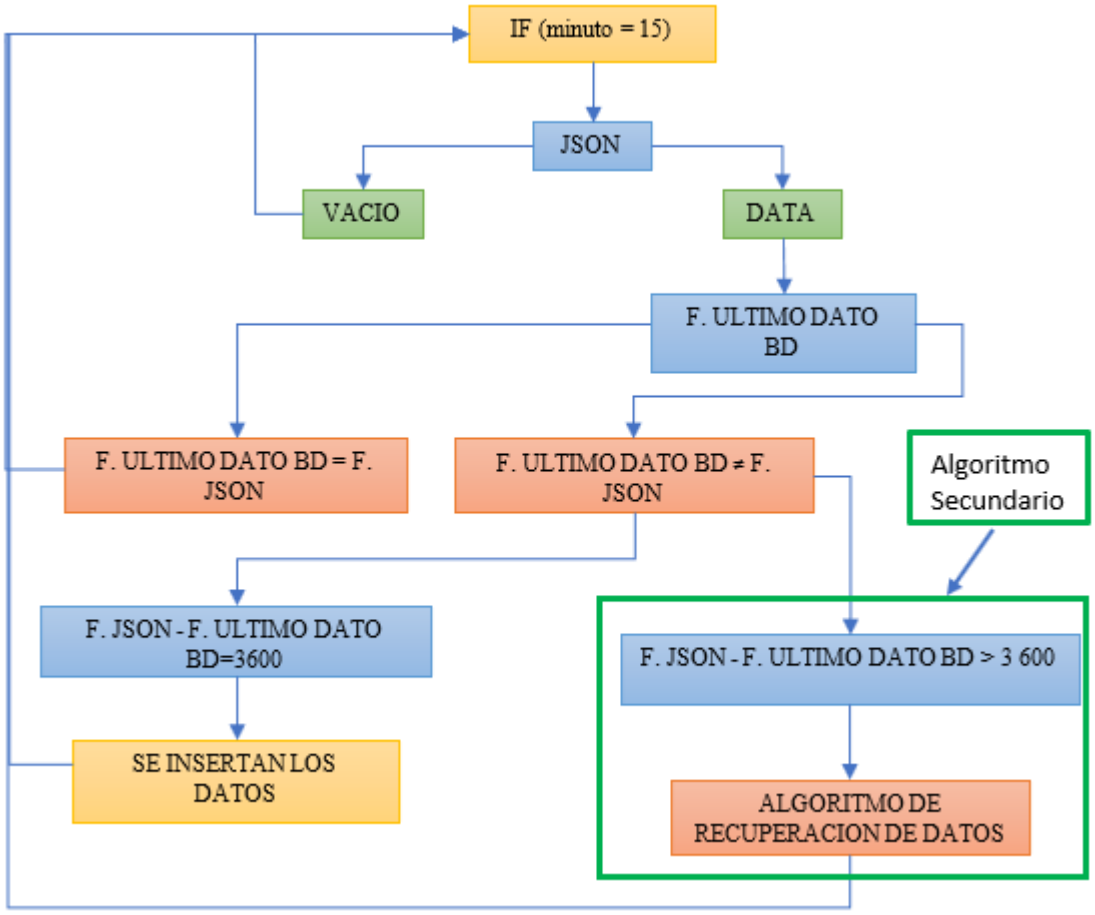


Figura 5: Diagrama de flujo para subir datos a la nube

1.5.2 Algoritmo secundario

Es un algoritmo cuya finalidad es exclusivamente recuperar los datos que por razones externas no se actualizan en la BD.

- Primero se calcula el número total de datos que faltan por actualizar (Datos perdidos).
- La API no funciona con rangos de solicitud de más de un día, por lo que es importante conocer cuantos días faltan de actualizar y solicitar la información a la API por día.
- Para conocer el número de días que faltan por actualizar se utiliza la siguiente formula, donde el 24 indica la cantidad de datos que se registran en un día.

$$\#dias = \frac{DATOS\ PERDIDOS}{24}$$

- De la formula anterior no siempre se obtendrá un numero entero, así que para las solicitudes a la API se trabaja solo con el cociente.
- Los datos que se obtienen por día se van almacenando en una única lista.
- Cuando se obtienen ya los datos perdidos de todos esos días, se agregan también los datos faltantes que se calculan de la siguiente manera.

$$\#Datos\ faltantes = DATOS\ PERDIDOS - 24 * \#dias$$

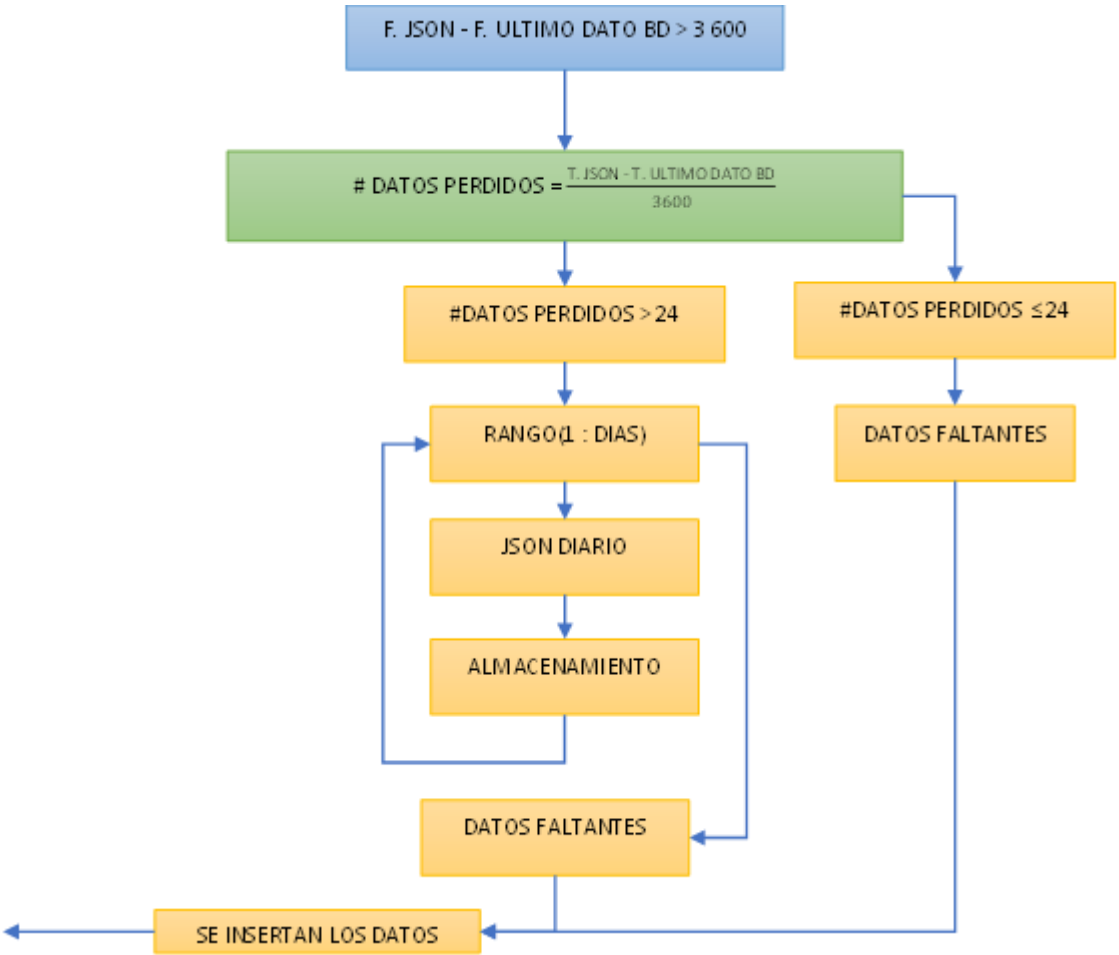


Figura 6: Algoritmo de recuperación de datos

El código ha sido desarrollado en Python y separado en funciones (Figura 7).

```
def grabar_BD(Datos_separados):···
def Mensaje_Json(temp_actual,x,y):···
def test(data_banano):···
def Separacion_datos(data_banano):···
def Separacion_datos2(M_json):···
def extraccion_ultimo_dato(data_banano):···
def Recup_datos_perdidos(ultimos_Datos,temp_actual):···
def comparacion_fechas(ultimos_Datos,temp_actual):···
```

Figura 7: Captura de pantalla funciones python

- Función grabar_BD: Es una función que contiene todas las credenciales de la BD, recibe los datos separados y los inserta a la BD, solo ingresa una fila de datos por cada hora.

- Función Mensaje_Json: Se encarga de enviar las solicitudes a la API de Davis y a la vez la almacena en una variable que es utilizada posteriormente.

- Función test: Lee el mensaje JSON almacenado por la función anterior y verifica si hay datos en el JSON. Esto se hace con la finalidad de que se pueda conocer la situación de la estación ya que al estar sin datos significaría que la estación ha perdido señal o ha sufrido alguna falla técnica por lo que se podrían tomar medidas al respecto.

- Función Separacion_datos: Separa el mensaje JSON recibido en variables para que se puedan manipular de manera independiente como el cambio a las unidades de interés. Esta función retorna una lista que contiene todas las 27 variables.

- Función Separación_datos2: Realiza la tarea de la anterior función, pero tiene la particularidad que solo se usa cuando en el código entra en el algoritmo secundario que es para la recuperación de los datos que se dejaron de actualizar.

- Función extracción_ultimo_dato: Extrae la última fecha registrada en la BD y también la fecha del JSON.

- Función Recuperación_datos_perdidos: Se ejecuta cuando se cumplan dos condiciones: la primera es cuando se dejan de actualizar los datos en los servidores de Davis y la segunda es que la cantidad de esos datos que se pierden son mayores o igual a 24 (1 día a más). Cuando se ejecuta la función las solicitudes de datos a los servidores de Davis se hacen por días y ya no por horas de esa forma se consigue que la

recuperación de datos sea mucho más rápida, como también se evita problemas de saturación al enviar muchas solicitudes.

- Función comparación_fechas: Compara las fechas sacadas por la función "extracción_ultimo_dato" y es aquí donde se evalúa si el mensaje JSON se graba o no. Si las fechas coinciden significa que esos datos ya han sido insertados.

  Si los datos no coinciden se presentan dos casos: El primero es que los datos del JSON son de la hora posterior a la última fecha insertada lo que significa un funcionamiento normal, pero también se puede dar el caso que por errores de conexión no se pudieran actualizar los datos y no se insertan en sus horas correspondientes.

2. Estaciones de Republica Dominicana

2.1. Objetivos del desarrollo del algoritmo

- Extraer los datos de las estaciones de Republica Dominicana.
- Solicitar datos a la API de Davis cada hora.
- Almacenamiento de los datos extraídos en una base de datos (BD) propia.
- Funcionamiento en tiempo real sin necesidad de intervención humana.

2.2. Características estaciones de República Dominicana

Para las pruebas de extracción de datos se utilizó la estación "La Pita", la elección fue de manera aleatoria.

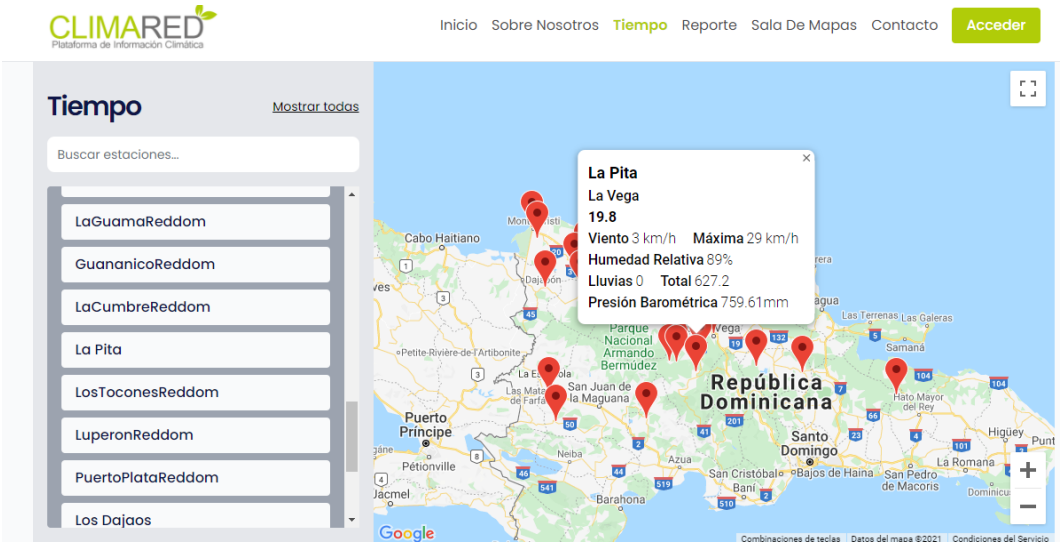


Figura 8: Estaciones de República Dominicana

De las variables que se muestran en la página web de dicha estación, las más importantes o las de interés son las que se mencionan a continuación (*Tabla 4*).

Tabla 4. Variables de interés

	Variable	
1	Temperatura	°C
2	Humedad	%
3	Índice de calor	°C
4	Viento frio	°C
5	Punto de rocío	°C
6	Presión barométrica	mb
7	Velocidad del viento	m/s
8	Dirección del viento	--
9	Intensidad de lluvia	mm
10	Evapotranspiración	mm
11	Radiación solar	W/m2

En la página web se muestran los valores de las variables de cada hora, por lo tanto, el código tiene que ejecutarse cada hora para extraer los últimos datos actualizados.

Figura 9: Entorno web de la estación "La Pita"- Captura 1.

Resumen de la Estación	Tiempo Actual	Máximas para hoy		Mínimas para hoy	
Temperatura Exterior	20 °C	26 °C	11:50am	16 °C	6:45am
Humedad Exterior	89 %	95 %	12:00am	69 %	11:52am
Índice de Calor	20.7 °C	27 °C	11:49am	---	---
Viento Frio	19.8 °C	---	---	16 °C	6:04am
Punto de Rocío	17.9 °C	21 °C	9:25am	14 °C	6:33am
Presión Barométrica	759.61 mm	812.6 mm	3:02am	422.96 mm	6:23am
Radiación Solar	30 W/m^2	1183 W/m^2	12:22pm	---	---
Radiación UV	0.0 Index	8.6 Index	12:21pm	---	---

Resumen del Viento	10 Minutos	Maximas Para Hoy	
Velocidad Media del Viento	2 km/h	1 km/h	1
Ráfagas de Viento	5 km/h	---	---

Figura 10: Entorno web de la estación "La Pita"- Captura 2.

Estudiando el HTML de esta página web se puede conocer la forma en la que los datos se actualizan, y es por medio de un formato JSON, siendo dicho formato característico de una estación DAVIS (En el JSON se ve la ID de la estación). Por lo tanto, se concluye que esta estación es Davis, y que los datos que se muestran son gracias al formato JSON que brinda Davis, pero la desventaja de este JSON es que no contiene datos históricos.

Para acceder a los datos de esta estación se tiene que conocer el enlace que el HTML requiere para actualizar los datos y esto se pudo conseguir gracias a la herramienta "Inspeccionar", en esta estación se utiliza el siguiente enlace: http://app.climared.com/api/v1/get/remote/station/data/LaPista.

Por medio del enlace mencionado se recibe un formato JSON con todos los datos actuales de la estación donde se extraen las variables de interés y se insertan en un servidor propio. El procedimiento se resume en el diagrama de flujo de la Figura 11.

2.3. Desarrollo del algoritmo

- Se ejecuta en el minuto 15 de cada hora.
- Se verifica si el mensaje JSON está vacío o tiene datos.
- Si tiene datos se extrae la última fecha de la BD y la fecha del JSON para la comparación.
- Si esas fechas coinciden, los datos ya han sido insertados o bien la API no está respondiendo con datos nuevos.
- Si son distintos se proceden a guardar y se repite el ciclo.

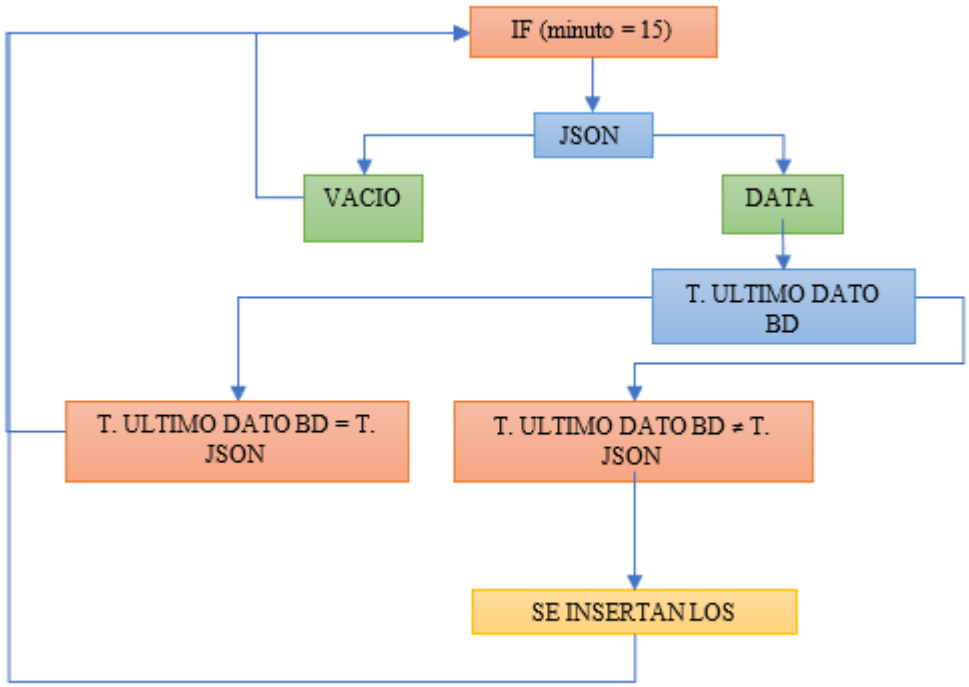


Figura 11: Diagrama de flujo de interface estaciones República Dominicana

II. Algoritmos de funcionamiento basados en indicadores atmosféricos

3. Librerías pandas

Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos.

Las principales características de esta librería son:

- Define nuevas estructuras de datos basadas en los arrays de la librería NumPy, pero con nuevas funcionalidades.
- Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL.
- Permite acceder a los datos mediante índices o nombres para filas y columnas.
- Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
- Permite trabajar con series temporales.
- Realiza todas estas operaciones de manera muy eficiente.

3.1 Tipos de datos de Pandas

Pandas dispone de tres estructuras de datos diferentes:

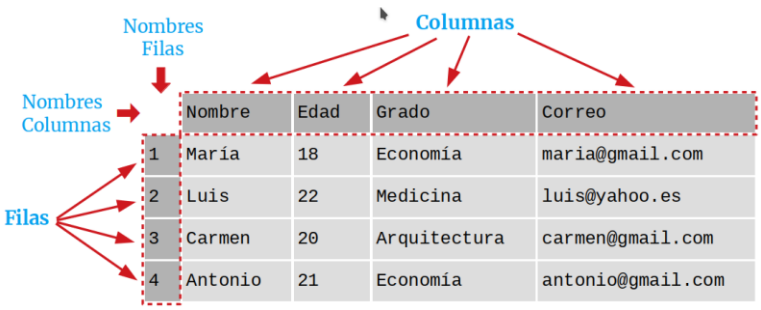
- Series: Estructura de una dimensión.
- DataFrame: Estructura de dos dimensiones (tablas).
- Panel: Estructura de tres dimensiones (cubos).

Estas estructuras se construyen a partir de arrays de la librería NumPy, añadiendo nuevas funcionalidades.

3.1.1 DataFrame

Para el desarrollo de las funciones para la aplicación, se hace uso de la clase DataFrame. Un objeto del tipo DataFrame define un conjunto de datos estructurado en forma de tabla donde cada columna es un objeto de tipo Series, es decir, todos los datos de una misma columna son del mismo tipo, y las filas son registros que pueden contender datos de distintos tipos.

Un DataFrame contiene dos índices, uno para las filas y otro para las columnas, y se puede acceder a sus elementos mediante los nombres de las filas y las columnas.



	Nombre	Edad	Grado	Correo
1	María	18	Economía	maria@gmail.com
2	Luis	22	Medicina	luis@yahoo.es
3	Carmen	20	Arquitectura	carmen@gmail.com
4	Antonio	21	Economía	antonio@gmail.com

Figura 12: Ejemplo de objeto DataFrame con información de los alumnos de un curso. Cada fila corresponde a un alumno y cada columna a una variable

3.1.2 Atributos de un DataFrame

Existen varias propiedades o métodos para ver las características de un DataFrame.

- df.info(): Devuelve información (número de filas, número de columnas, índices, tipo de las columnas y memoria usado) sobre el DataFrame df.
- df.shape: Devuelve una tupla con el número de filas y columnas del DataFrame df.
- df.size: Devuelve el número de elementos del DataFrame.
- df.columns: Devuelve una lista con los nombres de las columnas del DataFrame df.
- df.index: Devuelve una lista con los nombres de las filas del DataFrame df.
- df.dtypes: Devuelve una serie con los tipos de datos de las columnas del DataFrame df.

- df.head(n): Devuelve las n primeras filas del DataFrame df.
- df.tail(n): Devuelve las n últimas filas del DataFrame df.

3.1.3 Acceso a los elementos de un DataFrame: El acceso a los datos de un DataFrame se puede hacer a través de posiciones o través de los nombres de las filas y columnas.

Accesos mediante posiciones

- df.iloc[i, j]: Devuelve el elemento que se encuentra en la fila "i" y la columna "j" del DataFrame "df". Pueden indicarse secuencias de índices para obtener partes del DataFrame.
- df.iloc[filas, columnas] : Devuelve un DataFrame con los elementos de las filas de la lista filas y de las columnas de la lista columnas.
- df.iloc[i] : Devuelve una serie con los elementos de la fila "i" del DataFrame "df".

Acceso a los elementos mediante nombres

- df.loc["fila", "columna"]: Devuelve el elemento que se encuentra en la fila con nombre "fila" y la columna con nombre "columna" del DataFrame "df".
- df.loc["filas", "columnas"]: Devuelve un DataFrame con los elementos que se encuentra en las filas con los nombres de la lista "filas" y las columnas con los nombres de la lista "columnas" del DataFrame "df".
- df["columna"]: Devuelve una serie con los elementos de la columna de nombre "columna" del DataFrame "df".
- df.columna: Devuelve una serie con los elementos de la columna de nombre "columna" del DataFrame "df". Es similar al método anterior pero solo funciona cuando el nombre de la columna no tiene espacios en blanco.

3.1.4 Resumen descriptivo de un DataFrame: Al igual que para las series, los siguientes métodos permiten resumir la información de un DataFrame por columnas:

- df.count(): Devuelve una serie número de elementos que no son nulos ni NaN en cada columna del DataFrame df.
- df.sum(): Devuelve una serie con la suma de los datos de las columnas del DataFrame df cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena str.
- df.cumsum(): Devuelve un DataFrame con la suma acumulada de los datos de las columnas del DataFrame df cuando los datos son de un tipo numérico.
- df.min(): Devuelve una serie con los menores de los datos de las columnas del

DataFrame df.

- df.max(): Devuelve una serie con los mayores de los datos de las columnas del DataFrame df.
- df.mean(): Devuelve una serie con las medias de los datos de las columnas del DataFrame df cuando los datos son de un tipo numérico.
- df.std(): Devuelve una serie con las desviaciones típicas de los datos de las columnas del DataFrame df cuando los datos son de un tipo numérico.
- df.describe(include = tipo): Devuelve un DataFrame con un resumen estadístico de las columnas del DataFrame df del tipo tipo. Para los datos numéricos (number) se calcula la media, la desviación típica, el mínimo, el máximo y los cuartiles de las columnas numéricas. Para los datos no numéricos (object) se calcula el número de valores, el número de valores distintos, la moda y su frecuencia. Si no se indica el tipo solo se consideran las columnas numéricas.

3.1.5 Funciones combinadas

Se puede usar a la función loc para aplicar cambios en los elementos de un DataFrame directamente con la siguiente estructura:

- df.loc[df[“columna”] (==, <, >) “Valor”, “columna”] = “valor asignado”

Se pueden aplicar una operación sobre los elementos de una columna del DataFrame, filtrando los valores de una segunda columna.

- df[df[“columna2”](==,>,<) “Valor”]][columna].sum()

Las condiciones puede ser de igualdad (==) o de desigualdad (<,>,>=,<=), comparando valores numéricos o comparación de cadenas de datos, también se pueden reemplazar a la operación suma por cualquiera que aparece en el resumen descriptivo de DataFrame.

4. Algoritmos de funcionamiento

4.1. Tratamiento de Datos

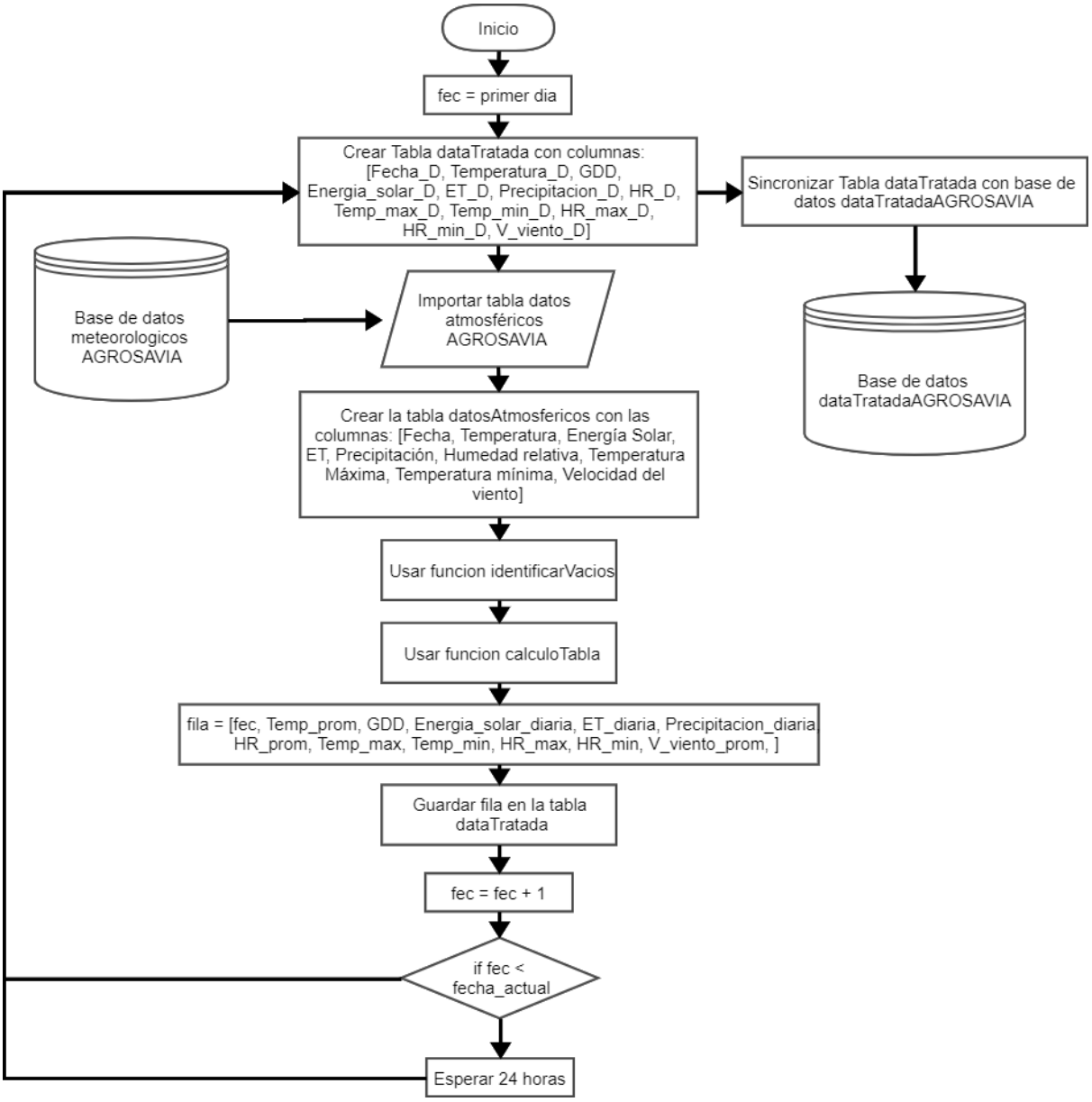


Figura 13: Diagrama de flujo tratamiento de datos AGROSAVIA

Como se detalla en *Figura 13* y *Figura 14*, el tratamiento de datos inicia leyendo la fecha actual, para poder descargar la información de la estación meteorológica de las bases de datos correspondientes. Actualmente se tienen dos bases de datos, cada una corresponde a una estación meteorológica en su respectiva ubicación: la estación meteorológica de AGROSAVIA que corresponde al equipo instalado en Colombia y la estación meteorológica de ASPROBO que corresponde al equipo instalado en el distrito de Buenos Aires, provincia

de Morropón en el departamento de Piura, Perú.

Una vez se descargaron los datos se reemplazan los valores que la estación meteorológica define como vacíos, en los casos de ambas estaciones colocan dos guiones cuando el dato no pudo ser subido correctamente a la nube. A estos valores se les debe reemplazar con un valor vacío (NaN) en la tabla generada con los datos descargados de la base de datos atmosféricos. De esta manera al hacer la suma de los valores y el conteo de valores no se tendrá un error.

Posteriormente se usa la rutina de cálculo por tablas, en la cual se usan las funciones de la librería Pandas. En *Figura 14* se detallan las operaciones que se realizan, primero se crea una nueva columna "Temperatura Ajustada" la cual se usa para el cálculo de los grados día diarios, como condiciones se tiene que la temperatura ajustada como máximo debe ser 33 y como mínimo deber ser 13.

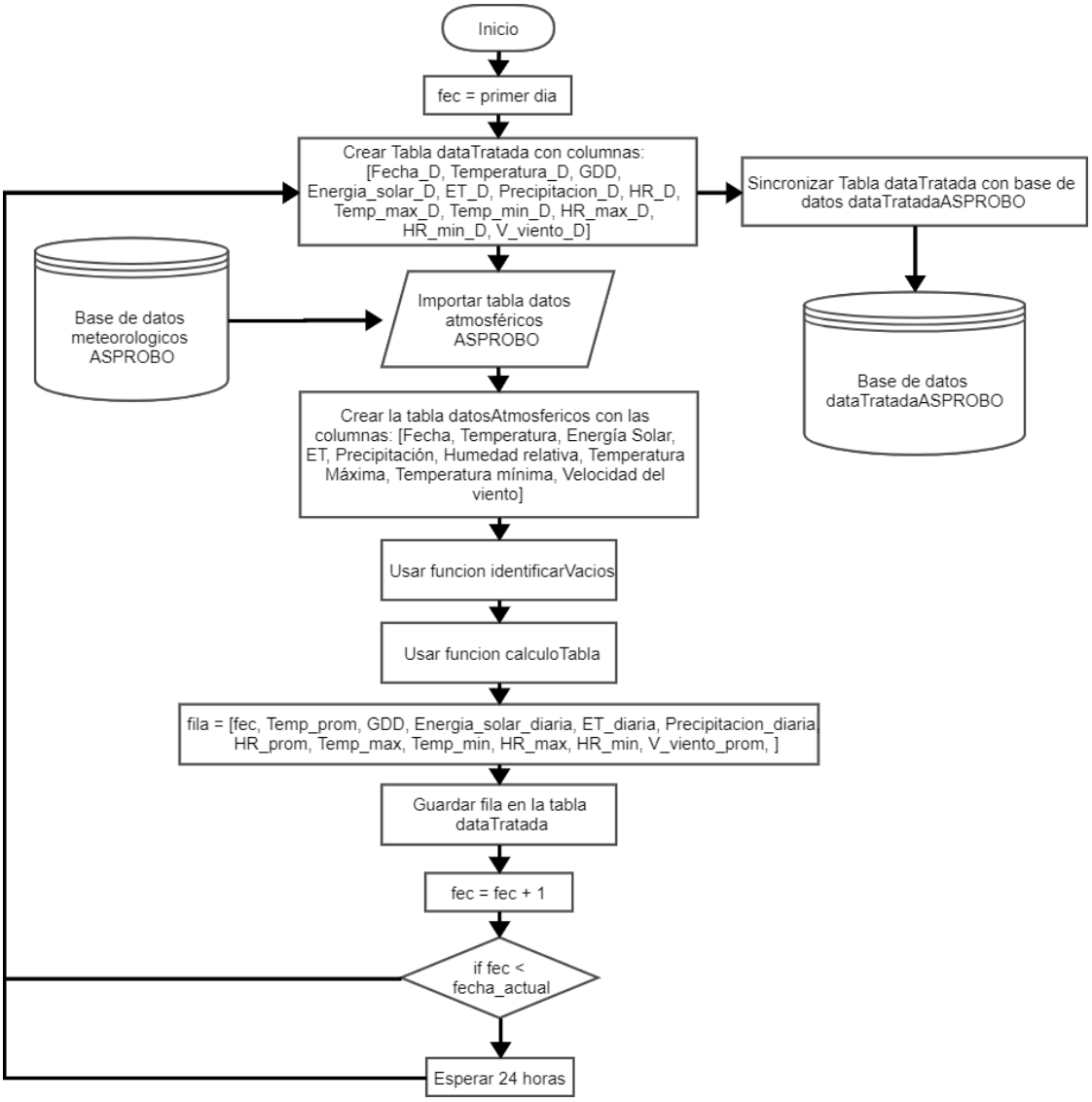


Figura 14: Diagrama de flujo tratamiento de datos ASPROBO

Luego se calcula la temperatura promedio para lo cual se suman todas las filas de la columna “Temperatura”, que cumplan con la condición de que el valor de la columna “Fecha” coincida con el valor de la fecha ingresada al inicio del algoritmo, y se divide entre el conteo de las filas de la columna “Temperatura” que cumplen con la condición de que el valor de la columna “Fecha” coincida con el valor de la fecha ingresada, se muestra en el diagrama de flujo de esta subrutina, ver *Figura 14*.

Para calcular los grados día diario (GDD) se calcula el promedio de la columna “Temperatura Ajustada”, los valores de la columna “Temperatura Ajustada” donde los valores de la columna “Fecha” sean igual a la fecha ingresada, se contabilizan y se suman. Con el promedio calculado se resta los 13 Grados Centígrados que corresponde a la temperatura base de desarrollo del banano.

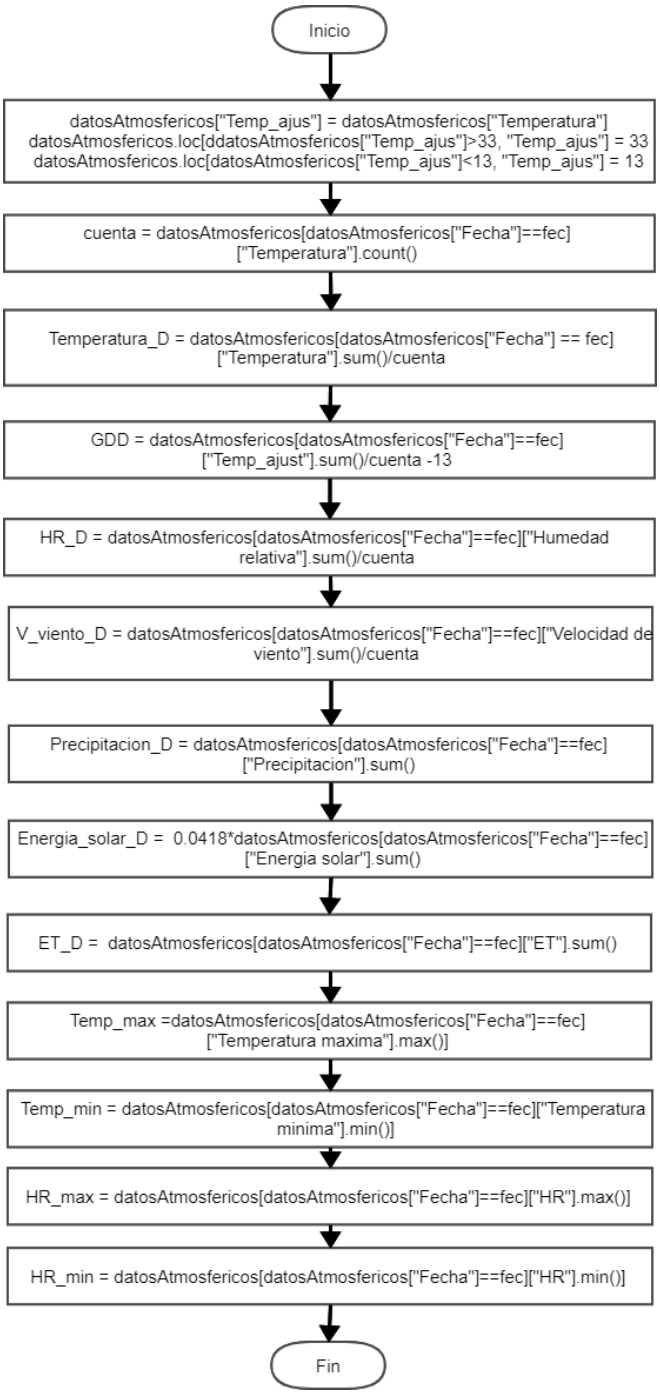


Figura 15: Función de cálculo por tablas.

El procedimiento que se usó para calcular el promedio de la temperatura se repite para las variables Humedad Relativa y Velocidad viento. En el caso de las variables de la Precipitación diaria, evapotranspiración diaria y energía solar diaria, únicamente se suman los de las filas de valores respectivos que cumplan con que el valor de la columna “Fecha” es igual al valor de la fecha ingresada.

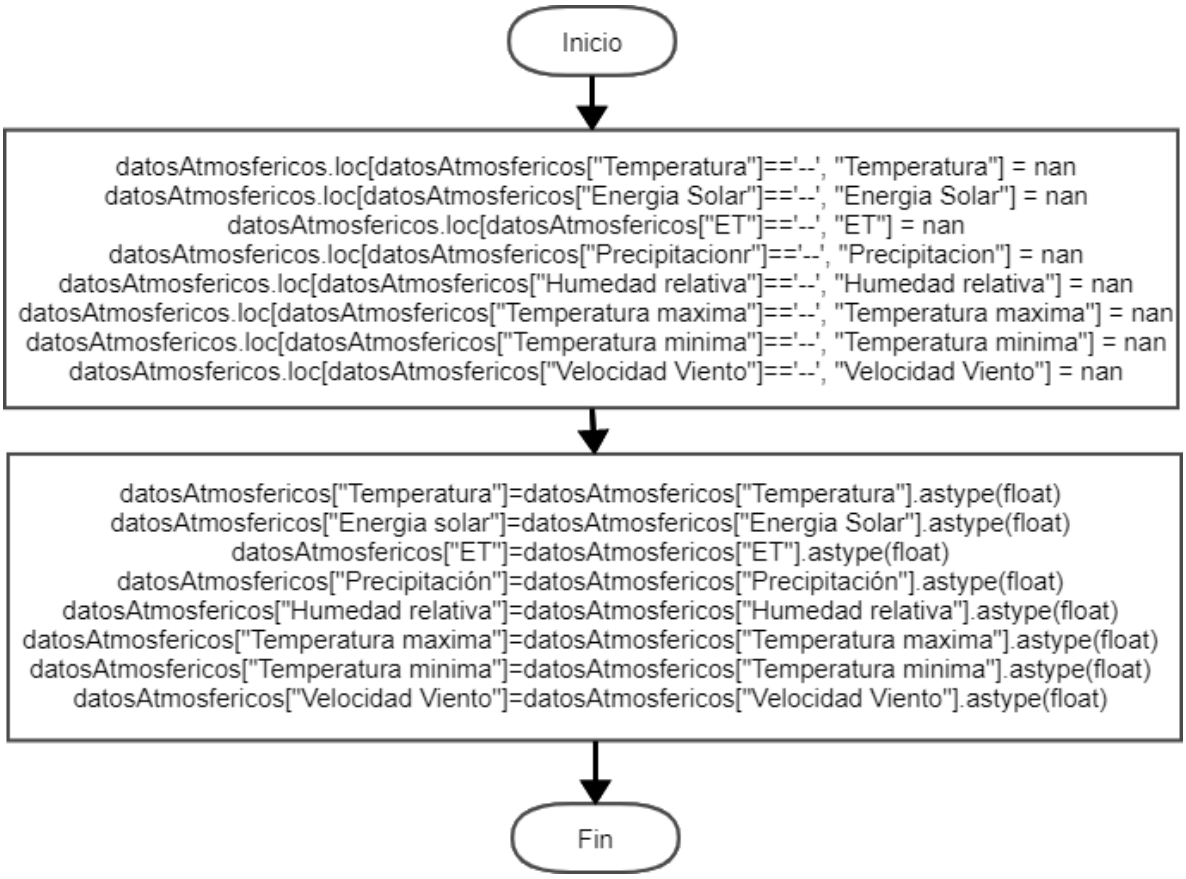


Figura 16: Identificar vacías.

La temperatura máxima del día es igual al valor máximo de las filas, en donde la columna "Fecha" sea igual a la fecha ingresada de la columna "Temperatura máxima". Se hace de forma igual con la humedad relativa máxima del día usando la columna humedad relativa. La temperatura mínima del día es igual al valor mínimo de las filas, en donde la columna "Fecha sea igual a la fecha ingresada, de la columna "Temperatura mínima". Se hace de forma igual con la humedad relativa mínima del día usando la columna humedad relativa.

Este proceso está pensado para que se haga de forma automática una vez al día, de forma que se tenga disponible todos los datos medidos durante el día.

4.2. Funcionalidades

4.2.1. Tasa potencial de crecimiento de hojas y gráficas.

Los diagramas de flujo que corresponden a esta función se muestran en *Figura 17* y *Figura 18*. Luego de iniciar el algoritmo se solicita una fecha que ingresa el usuario, se crea la tabla con los datos tratados de la base de datos correspondiente (ASPROBO para usuarios de Piura y AGROSAVIA para usuarios de Colombia). La tabla tiene las siguientes columnas:

Fecha, Temperatura promedio diaria, Grado día diario, Energía solar diaria, evapotranspiración diaria, precipitación diaria, humedad relativa promedio diaria, temperatura máxima del día, temperatura mínima del día, humedad relativa máxima del día, humedad relativa mínima del día, velocidad del viento promedio diaria.

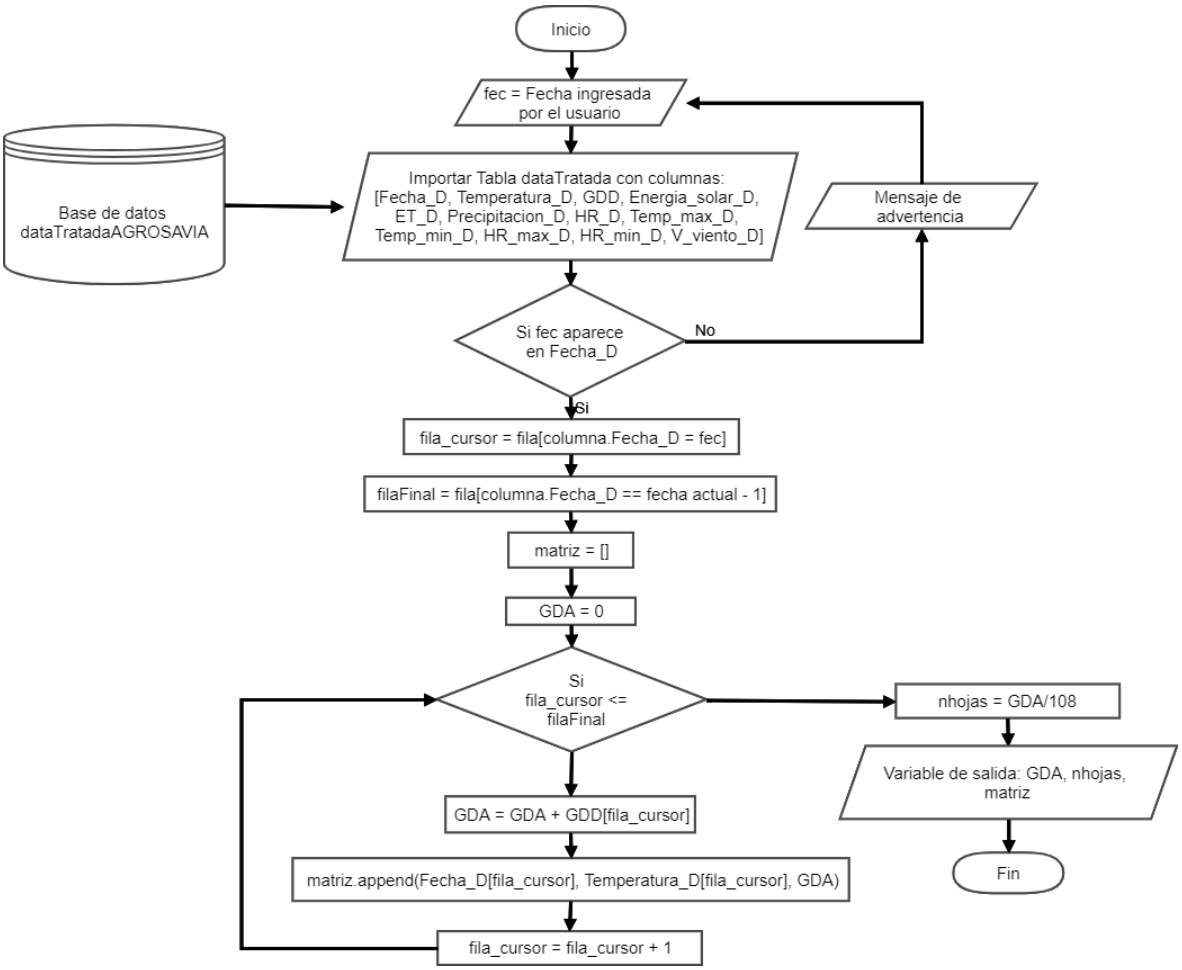


Figura 17: Algoritmo del cálculo de numero de hojas y gráfica ASPROBO

Se verifica que la fecha ingresada por el usuario se encuentra en la columna “Fecha” para esto se debe cumplir que la fecha en tiene que ser anterior al día donde se está realizando la consulta, si no se cumple esta condición se lanza un mensaje de advertencia al usuario, si la condición si se cumple se determina que la fila cursor es igual al número de fila donde el valor de la columna “Fecha” es igual a la fecha ingresada por el usuario.

La fila final se define como la fila donde el valor de la columna “Fecha” es igual a la fecha del día de ayer.

Se define una matriz donde se ordenan los datos para las gráficas, y una variable GDA que acumulara el valor de los grados día diarios hasta que se llegue a la última fila. Se hace un

bucle condicional, la fila cursor debe ser menor igual a la última fila para poder ingresar al bucle.

Dentro del bucle se debe acumular el valor de los grados día diario en la variable GDA, se agrega como una fila de la matriz la fecha y la temperatura que corresponden a la fila cursor y el valor GDA, se incrementa el valor de la fila cursor.

Cuando la condición del bucle se supera, se calcula el número de hojas, dividiendo la variable GDA entre 108 y se retorna al programa las variables de salida: GDA, numero de hojas y matriz.

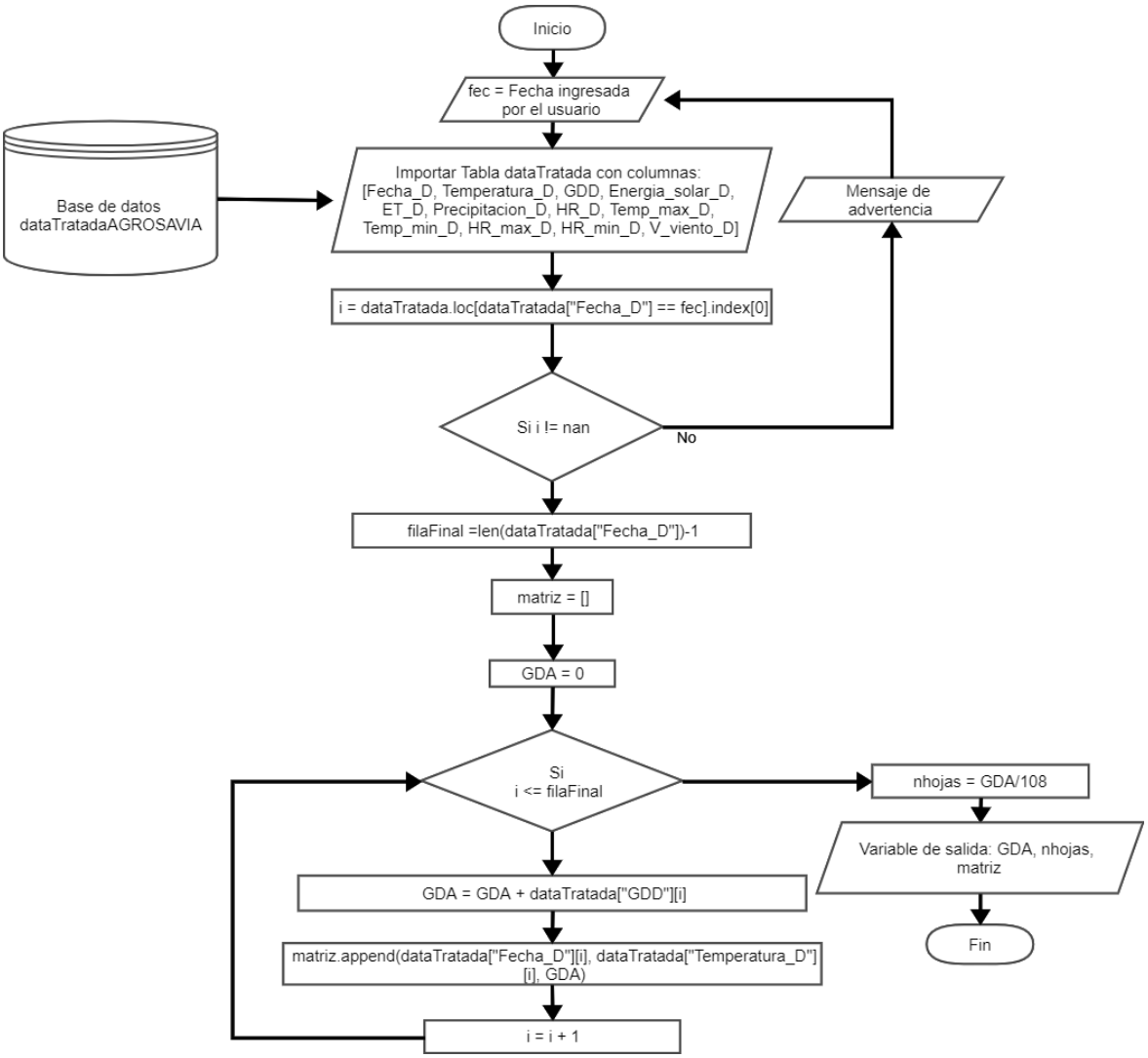


Figura 18: Algoritmo del cálculo de numero de hojas y gráfica AGROSAVIA

4.2.2. Grados día para cosecha – Estimación de fecha de floración y gráfica

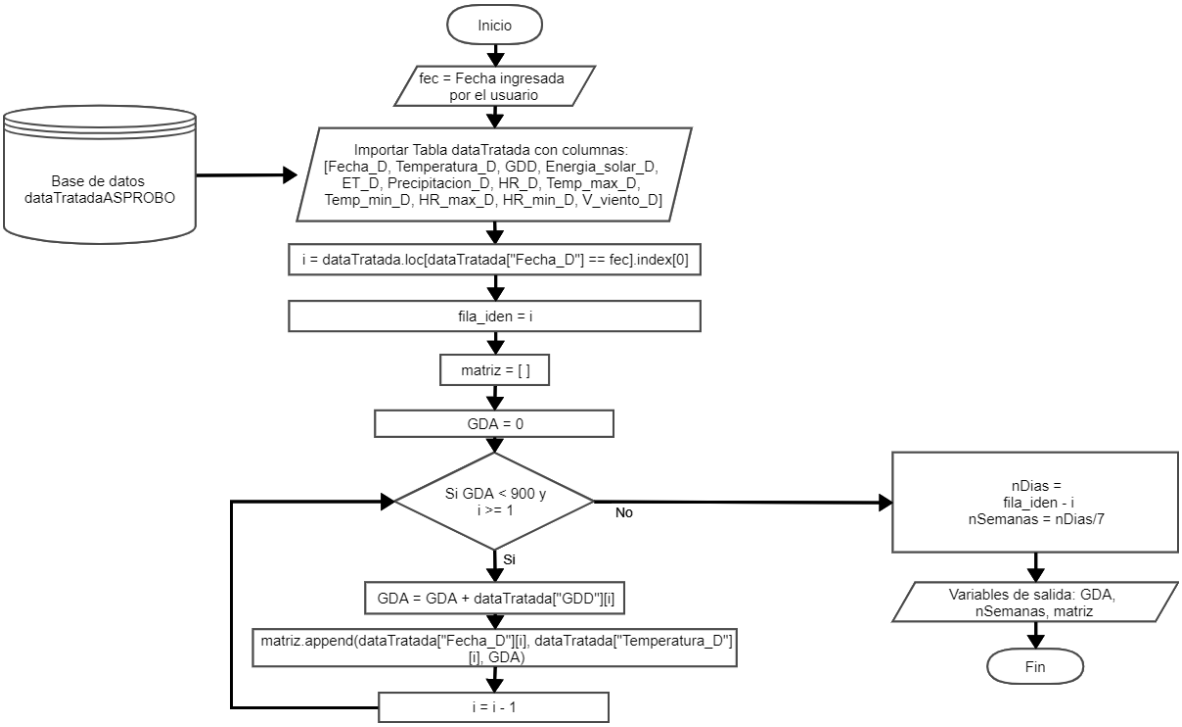


Figura 19: Algoritmo grados día para cosecha - estimación de fecha de floración y graficas ASPROBO

Los diagramas de flujo que corresponden a esta función se muestran en *Figura 19* y *Figura 20*. Luego de iniciar el algoritmo se solicita una fecha que ingresa el usuario, se crea la tabla con los datos tratados de la base de datos correspondiente (ASPROBO para usuarios de Piura y AGROSAVIA para usuarios de Colombia). La tabla tiene las siguientes columnas: Fecha, Temperatura promedio diaria, Grado día diario, Energía solar diaria, evapotranspiración diaria, precipitación diaria, humedad relativa promedio diaria, temperatura máxima del día, temperatura mínima del día, humedad relativa máxima del día, humedad relativa mínima del día, velocidad del viento promedio diaria.

Se verifica que la fecha ingresada por el usuario se encuentra en la columna “Fecha” para esto se debe cumplir que la fecha en tiene que ser anterior al día donde se está realizando la consulta, si no se cumple esta condición se lanza un mensaje de advertencia al usuario, si la condición si se cumple se determina que la fila cursor es igual al número de fila donde el valor de la columna “Fecha” es igual a la fecha ingresada por el usuario.

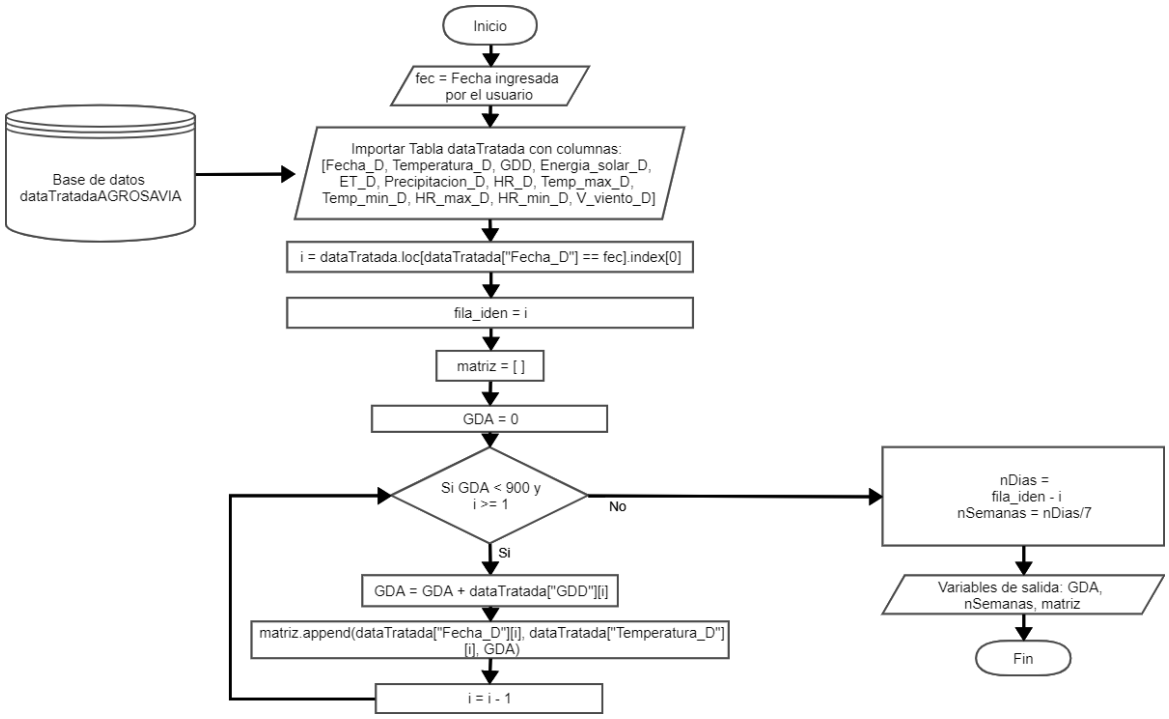


Figura 20: Algoritmo grados día para cosecha - estimación de fecha de floración y graficas AGROSAVIA

Luego se declaran las variables: matriz (para grabar los datos para ser graficados posteriormente) y GDA, esta última variable servirá como un acumulador de los grados días diarios.

Se tiene un bucle “while” con la siguiente condición, si GDA es menor a 900 y el valor de la fila cursor es mayor o igual a 1. Si la condición se cumple el valor de GDA incrementa el valor de la columna GDD correspondiente a la fila cursor, se guarda una fila en la variable matriz con la siguiente forma: Fecha y Temperatura correspondientes a la fila cursor, y el valor actual de GDA. Por último, dentro del condicional la fila cursor disminuye en 1. Si la condición no se cumple se calcula el número de días en los que se acumuló los 900 GDA, que se obtiene restando el valor de la fecha ingresada por el usuario menos el valor de la fila cursor. Para obtener el número de semana se divide al número entre días entre 7.

4.2.3. Grados día para cosecha – Estimación de fecha de cosecha y gráfica

Los diagramas de flujo que corresponden a esta función se muestran en *Figura 21* y *Figura 22*. Luego de iniciar el algoritmo se solicita una fecha que ingresa el usuario, se crea la tabla con los datos tratados de la base de datos correspondiente (ASPROBO para usuarios de Piura y AGROSAVIA para usuarios de Colombia). La tabla tiene las siguientes columnas: Fecha, Temperatura promedio diaria, Grado día diario, Energía solar diaria, evapotranspiración diaria, precipitación diaria, humedad relativa promedio diaria, temperatura máxima del día, temperatura mínima del día, humedad relativa máxima del día, humedad relativa mínima del día, velocidad del viento promedio diaria.

Se verifica que la fecha ingresada por el usuario se encuentra en la columna “Fecha” para esto se debe cumplir que la fecha en tiene que ser anterior al día donde se está realizando la consulta, si no se cumple esta condición se lanza un mensaje de advertencia al usuario, si la condición si se cumple se determina que la fila cursor es igual al número de fila donde el valor de la columna “Fecha” es igual a la fecha ingresada por el usuario.

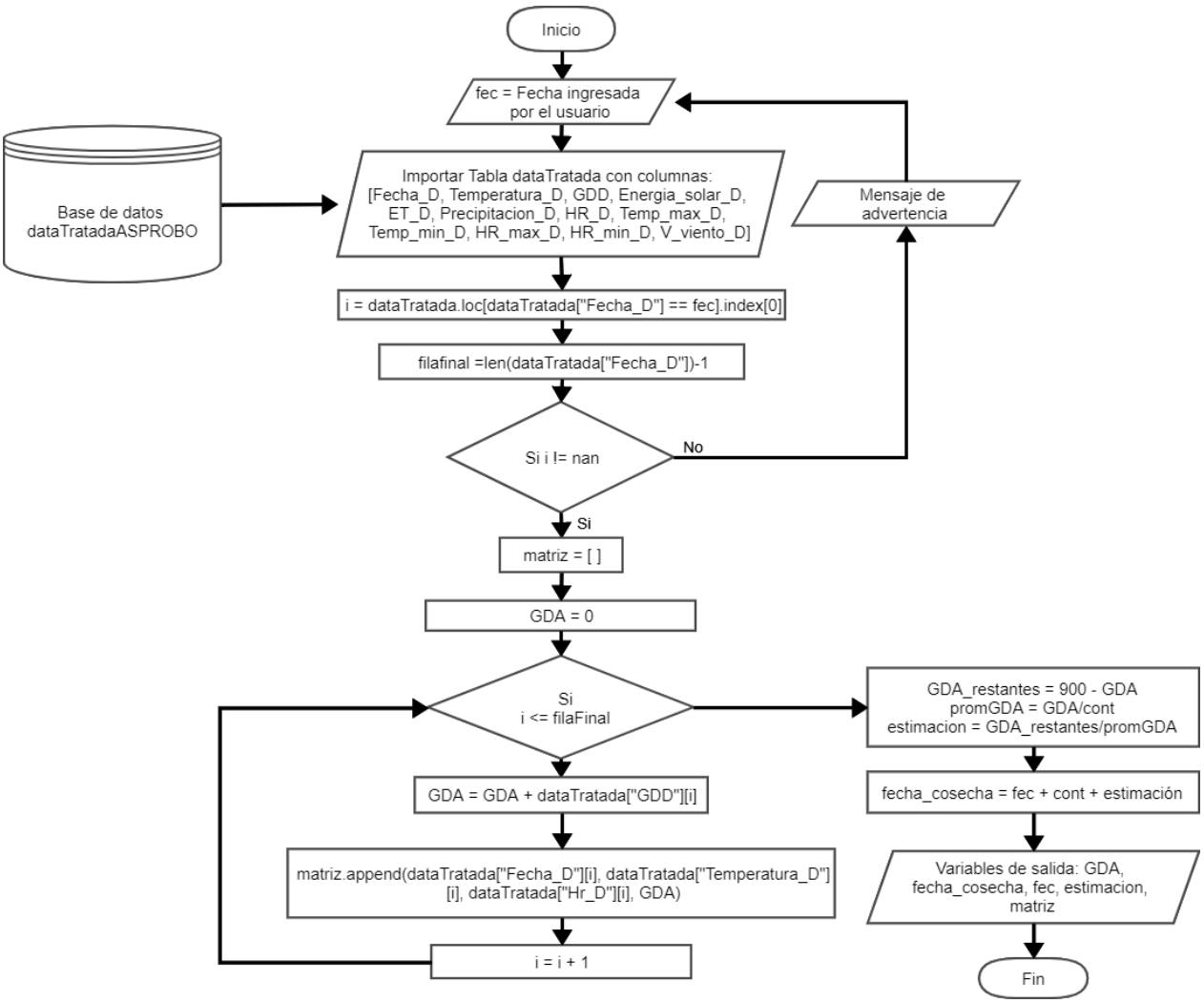


Figura 21: Algoritmo grados día para cosecha - estimación de fecha de cosecha y graficas ASPROBO

Se define una matriz donde se ordenan los datos para las gráficas, y una variable GDA que acumulara el valor de los grados día diarios hasta que se llegue a la última fila. Se hace un bucle condicional, la fila cursor debe ser menor igual a la última fila para poder ingresar al bucle.

Dentro de un bucle “while” con la condición de que la fila cursor sea menor a la fila final de la tabla, si la condición se cumple se acumulan en la variable GDA los valores de la columna

GDD que corresponde la fila cursor, además se agrega una fila a la variable matriz con la siguiente forma: Fecha, Temperatura y Humedad relativa correspondientes a la fila cursor, y el valor actual de GDA.

Cuando la condición se deje de cumplir, se calculan los GDA restantes, a los 900 grados días necesarios desde la floración a la cosecha se resta el valor de la variable GDA. Se obtiene el promedio de los GDA dividiendo a GDA entre el contador de filas. La estimación de días necesarios se obtiene dividiendo a los GDA restantes entre el promedio de GDA diarios.

La fecha de cosecha proyectada es igual a la suma de la fecha ingresada por el usuario más el conteo de filas más la estimación calculada.

Como variables de salida de esta función se tiene GDA, fecha de cosecha proyectada, estimación en días y matriz.

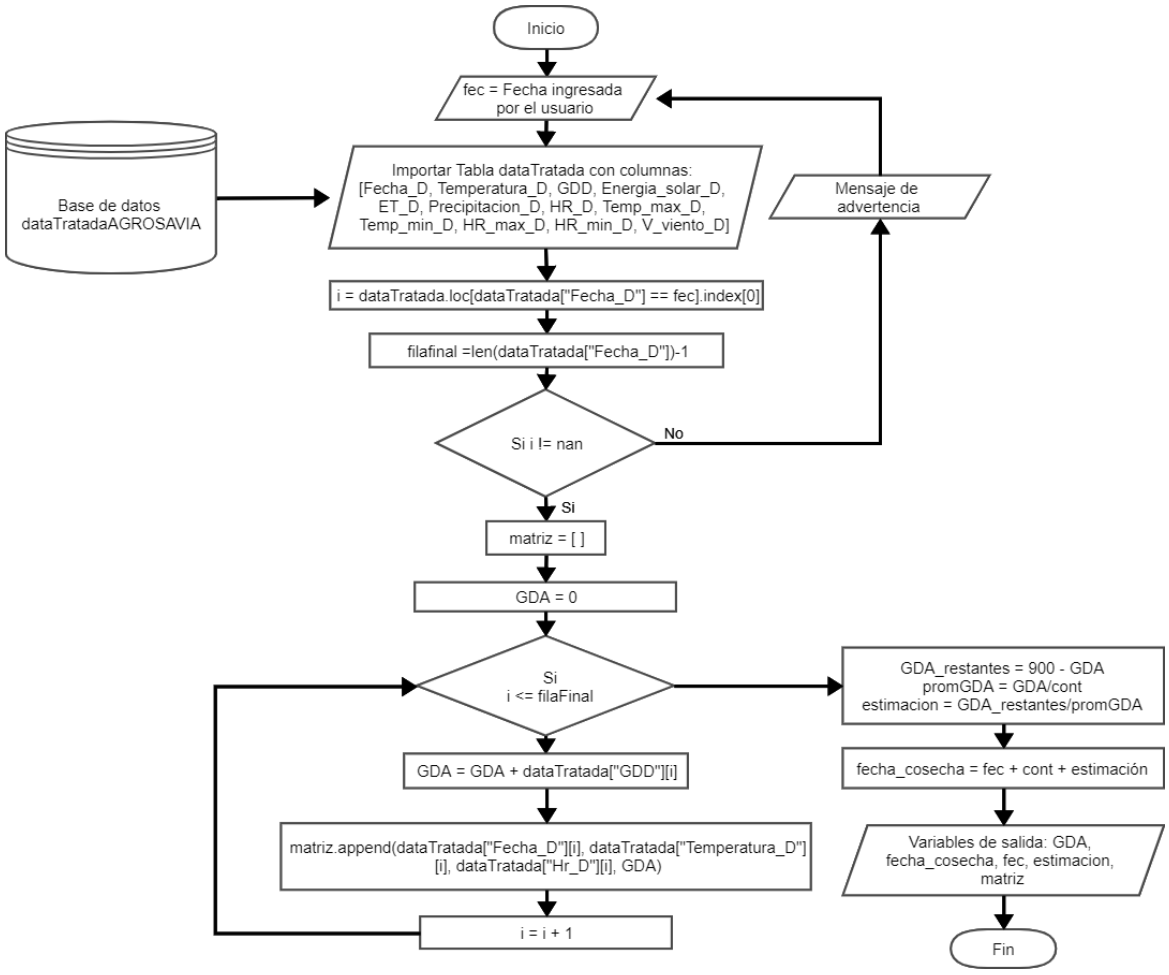


Figura 22: Algoritmo grados día para cosecha - estimación de fecha de cosecha y graficas AGROSAVIA

4.2.4. Cálculo de biomasa – Estimación de la masa producida por racimo y por parcela

Los diagramas de flujo que corresponden a esta función se muestran en *Figura 23* y*Figura*

24. Luego de iniciar el algoritmo se solicita una fecha que ingresa el usuario y la relación de planta por hectárea (rPA), se crea la tabla con los datos tratados de la base de datos correspondiente (ASPROBO para usuarios de Piura y AGROSAVIA para usuarios de Colombia). La tabla tiene las siguientes columnas: Fecha, Temperatura promedio diaria, Grado día diario, Energía solar diaria, evapotranspiración diaria, precipitación diaria, humedad relativa promedio diaria, temperatura máxima del día, temperatura mínima del día, humedad relativa máxima del día, humedad relativa mínima del día, velocidad del viento promedio diaria.

Se verifica que la fecha ingresada por el usuario se encuentra en la columna “Fecha” para esto se debe cumplir que la fecha en tiene que ser anterior al día donde se está realizando la consulta, si no se cumple esta condición se lanza un mensaje de advertencia al usuario, si la condición si se cumple se determina que la fila cursor es igual al número de fila donde el valor de la columna “Fecha” es igual a la fecha ingresada por el usuario.

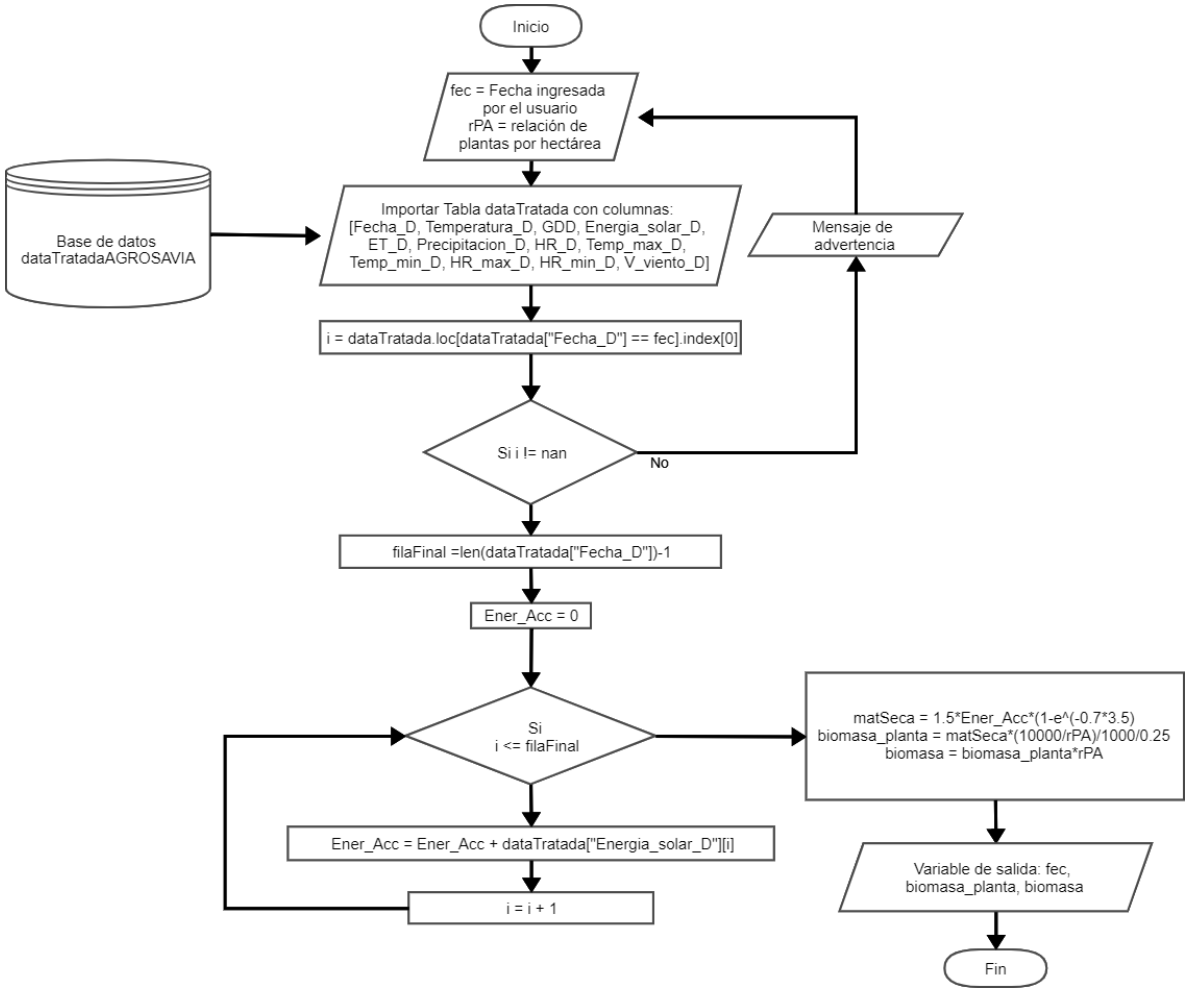


Figura 23: Algoritmo cálculo de biomasa - estimación del peso del racimo y parcela AGROSAVIA

Se define una variable Ener_Acc que acumulara el valor de la energía solar acumulada hasta que se llegue a la última fila. Se hace un bucle condicional, la fila cursor debe ser menor

igual a la última fila para poder ingresar al bucle.

Dentro de un bucle “while” con la condición de que la fila cursor sea menor a la fila final de la tabla, si la condición se cumple se acumulan en la variable Ener_Acc los valores de la columna Energía_solar_D que corresponde la fila cursor.

Cuando la condición se deje de cumplir, se calculan la materia seca, la biomasa por planta y la biomasa por parcela.

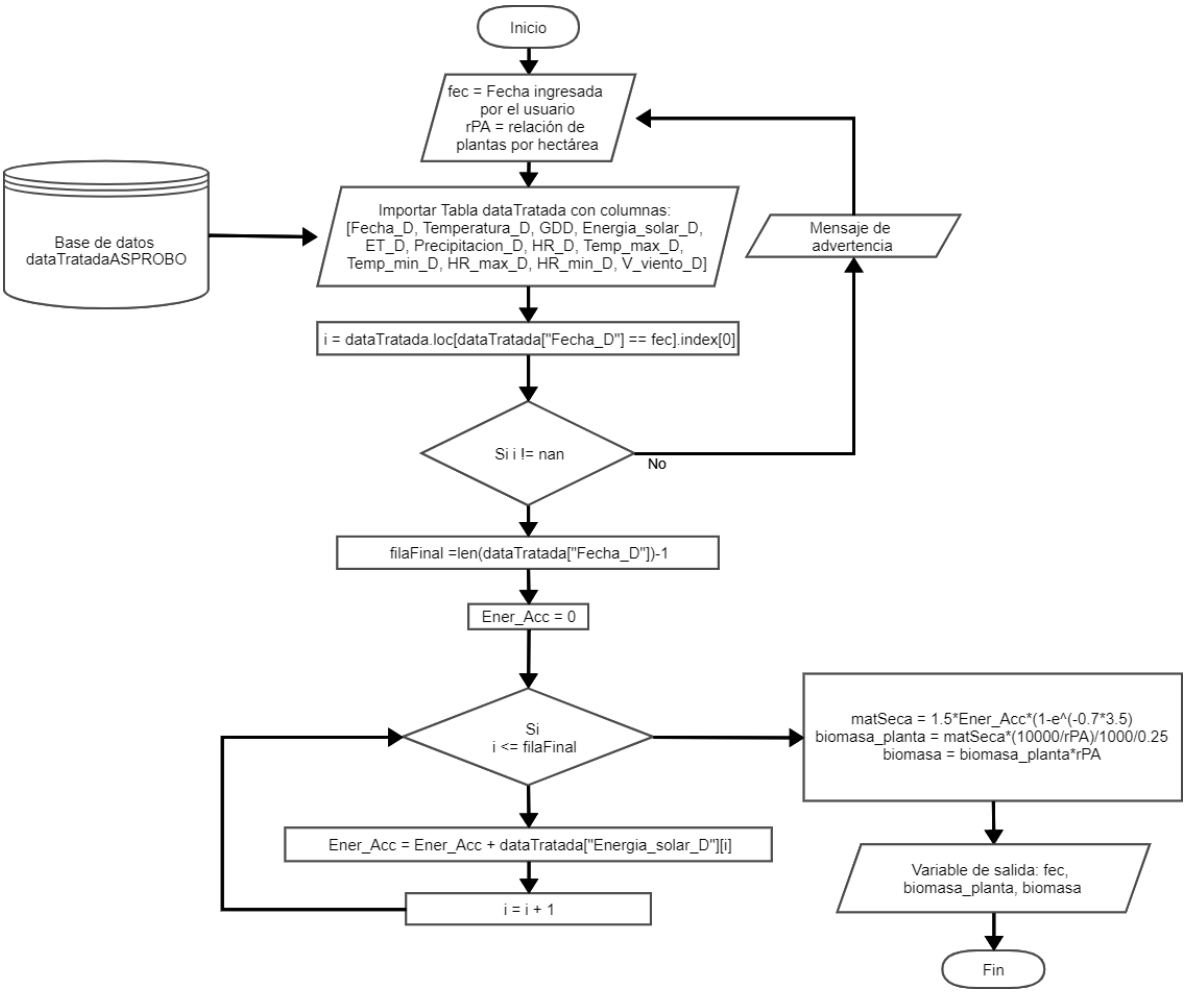


Figura 24: Algoritmo cálculo de biomasa - estimación del peso del racimo y parcela ASPROBO

4.2.5. Cálculo del potencial de nutriente a reponer – Estimación de la masa nutrientes que se deben reponer por planta y parcela

Los diagramas de flujo que corresponden a esta función se muestran en *Figura 25* y *Figura 26*. Luego de iniciar el algoritmo se solicita un intervalo de tiempo para ser evaluado y la relación de planta por hectárea (rPA), se crea la tabla con los datos tratados de la base de datos correspondiente (ASPROBO para usuarios de Piura y AGROSAVIA para usuarios de

Colombia). La tabla tiene las siguientes columnas: Fecha, Temperatura promedio diaria, Grado día diario, Energía solar diaria, evapotranspiración diaria, precipitación diaria, humedad relativa promedio diaria, temperatura máxima del día, temperatura mínima del día, humedad relativa máxima del día, humedad relativa mínima del día, velocidad del viento promedio diaria.

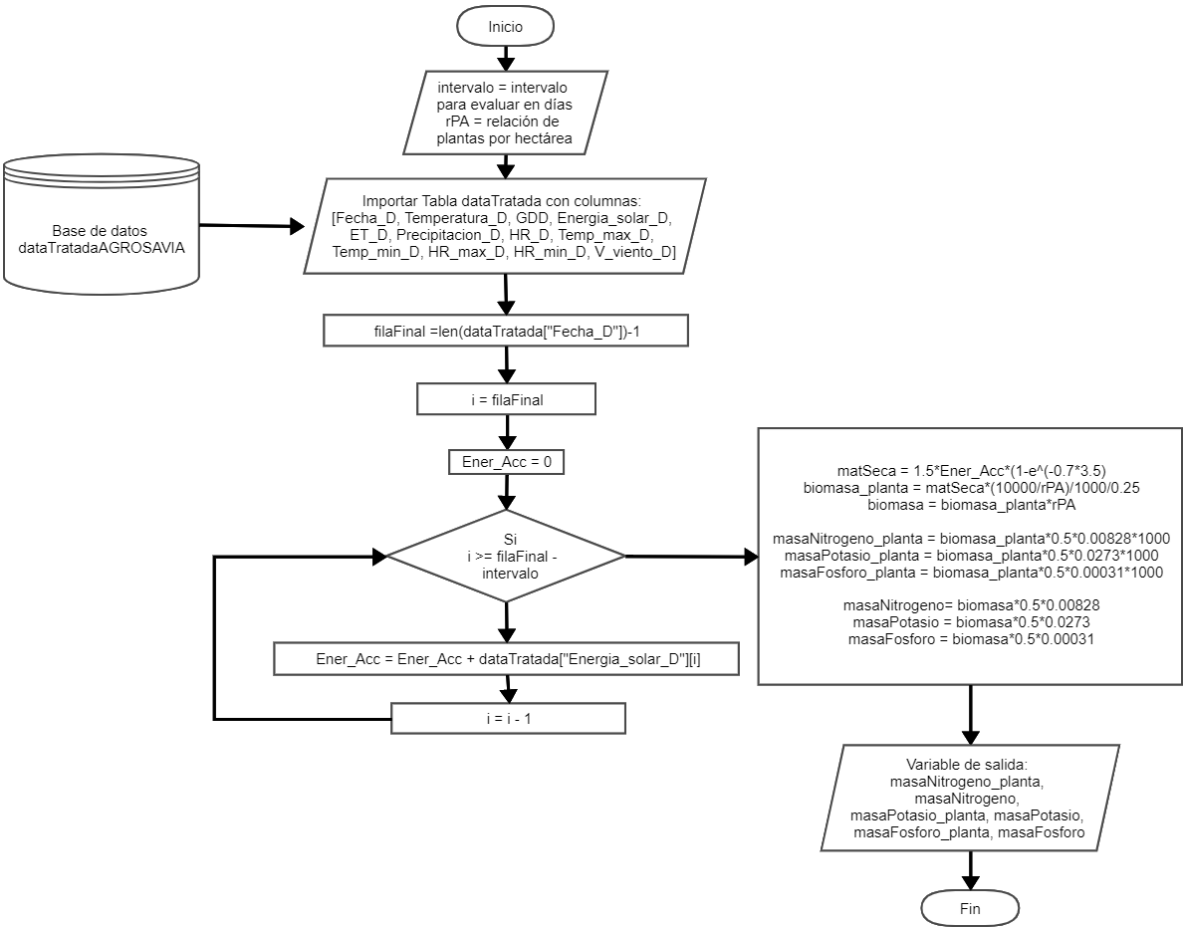


Figura 25: Algoritmo cálculo de potencial de nutrientes - estimación de la masa de nutrientes a reponer por planta y parcela AGROSAVIA

Se define una variable Ener_Acc que acumulara el valor de la energía solar acumulada hasta que se llegue a la última fila. Se hace un bucle condicional, la fila cursor debe ser menor igual a la última fila para poder ingresar al bucle.

Dentro de un bucle “while” con la condición de que la fila cursor sea mayor a la fila final menos el intervalo que ingreso el usuario, si la condición se cumple se acumulan en la variable Ener_Acc los valores de la columna Energía_solar_D que corresponde la fila cursor.

Cuando la condición se deje de cumplir, se calculan la materia seca, la biomasa por planta y la biomasa por parcela, para luego calcular las masas de nitrógeno, potasio y fosforo que se deben reponer por planta y por hectárea.

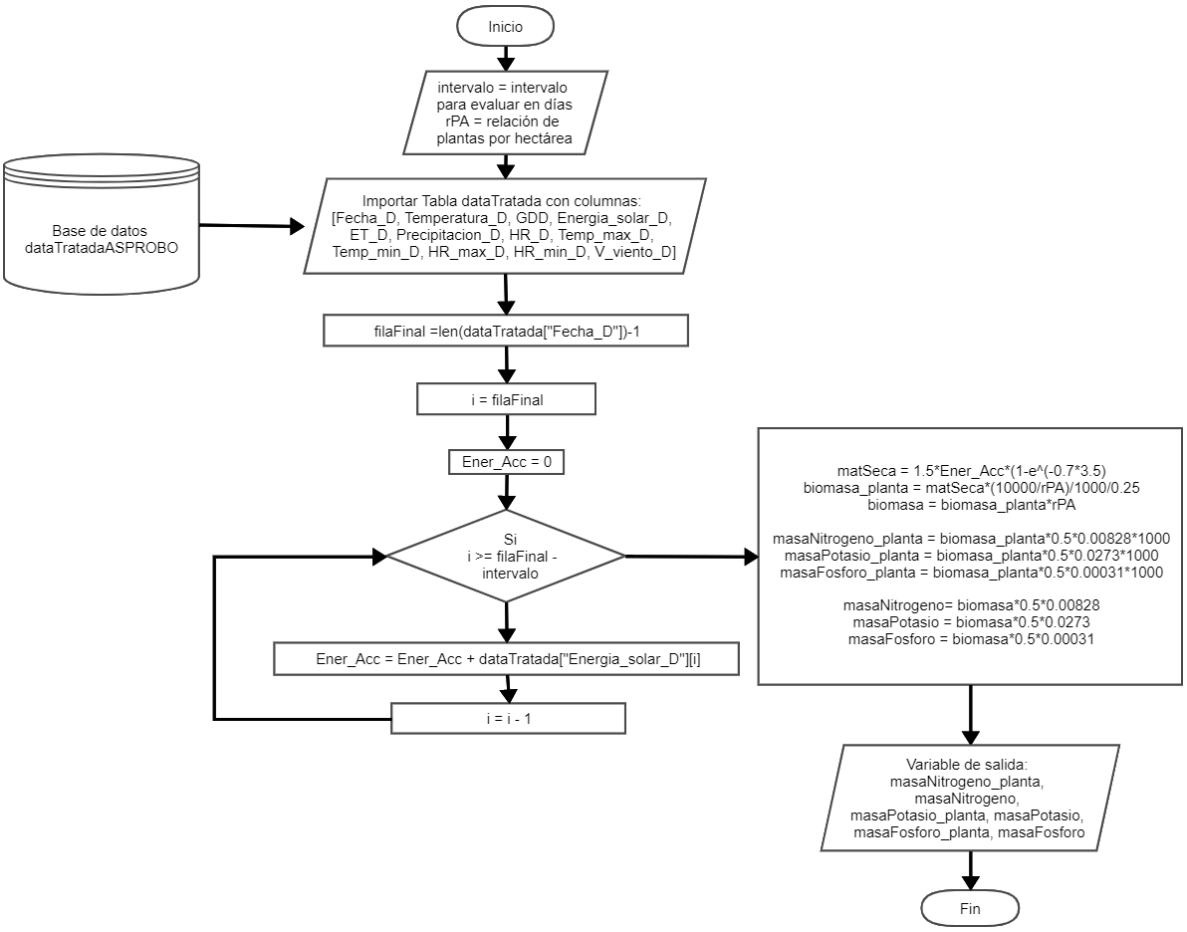


Figura 26: Algoritmo cálculo de potencial de nutrientes - estimación de la masa de nutrientes a reponer por planta y parcela ASPROBO

4.2.6. Cálculo de la necesidad hídrica – Estimación del volumen de agua que se deben reponer por hectárea

Los diagramas de flujo que corresponden a esta función se muestran en *Figura 27* y *Figura 28*. Luego de iniciar el algoritmo se solicita un intervalo de tiempo (días) para ser evaluado, tipo de suelo y el tipo de riego, se crea la tabla con los datos tratados de la base de datos correspondiente (ASPROBO para usuarios de Piura y AGROSAVIA para usuarios de Colombia). La tabla tiene las siguientes columnas: Fecha, Temperatura promedio diaria, Grado día diario, Energía solar diaria, evapotranspiración diaria, precipitación diaria, humedad relativa promedio diaria, temperatura máxima del día, temperatura mínima del día, humedad relativa máxima del día, humedad relativa mínima del día, velocidad del viento promedio diaria.

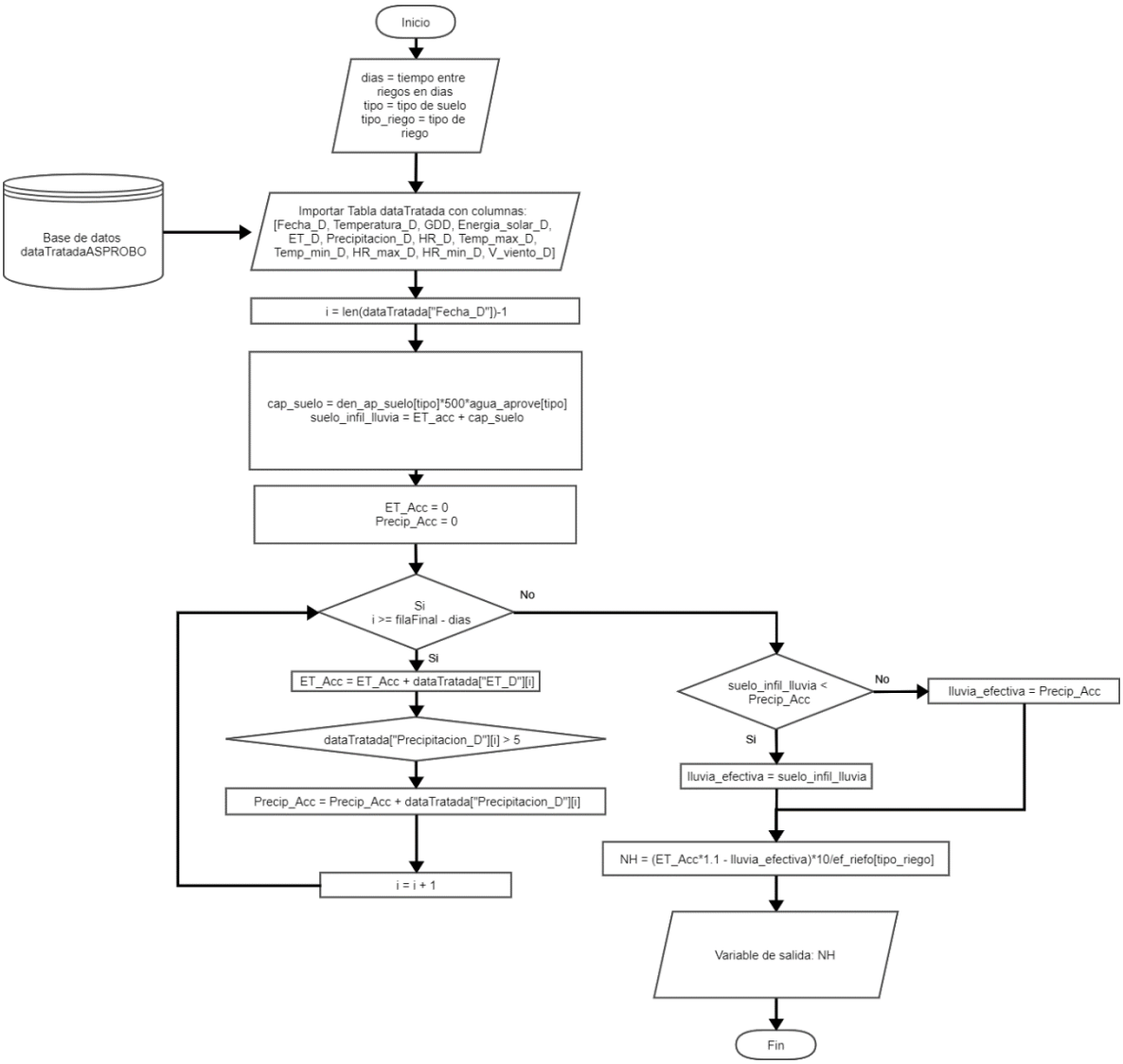


Figura 27: Algoritmo cálculo de la necesidad - estimación la cantidad de agua para evitar el estrés hídrico ASPROBO.

Se define a una variable i, la cual servirá de cursor para navegar por la tabla, luego se calcula la capacidad del suelo y la capacidad de infiltración del suelo.

Luego se declaran las variables ET_Acc y Precip_Acc en las cuales se acumularán los valores de la evapotranspiración y de las precipitaciones.

Dentro de un bucle “while” con la condición de que la fila cursor sea menor a la fila final de la tabla menos la variable días, si la condición se cumple se acumulan en la variable ET_Acc los valores de la columna ET_D que corresponde la fila cursor, lo mismo puede ocurrir con la variable Precip_Acc pero además se debe cumplir la condición que valor de la columna Precipitación_D debe ser mayor a 5 mm, sino es así solo se sumara 0.

Cuando la condición se deje de cumplir, se compara los valores de la capacidad de infiltración del suelo y la precipitación acumulada, la lluvia efectiva toma el menor valor de

los dos mencionados.

La necesidad hídrica se calcula a partir de la lluvia efectiva.

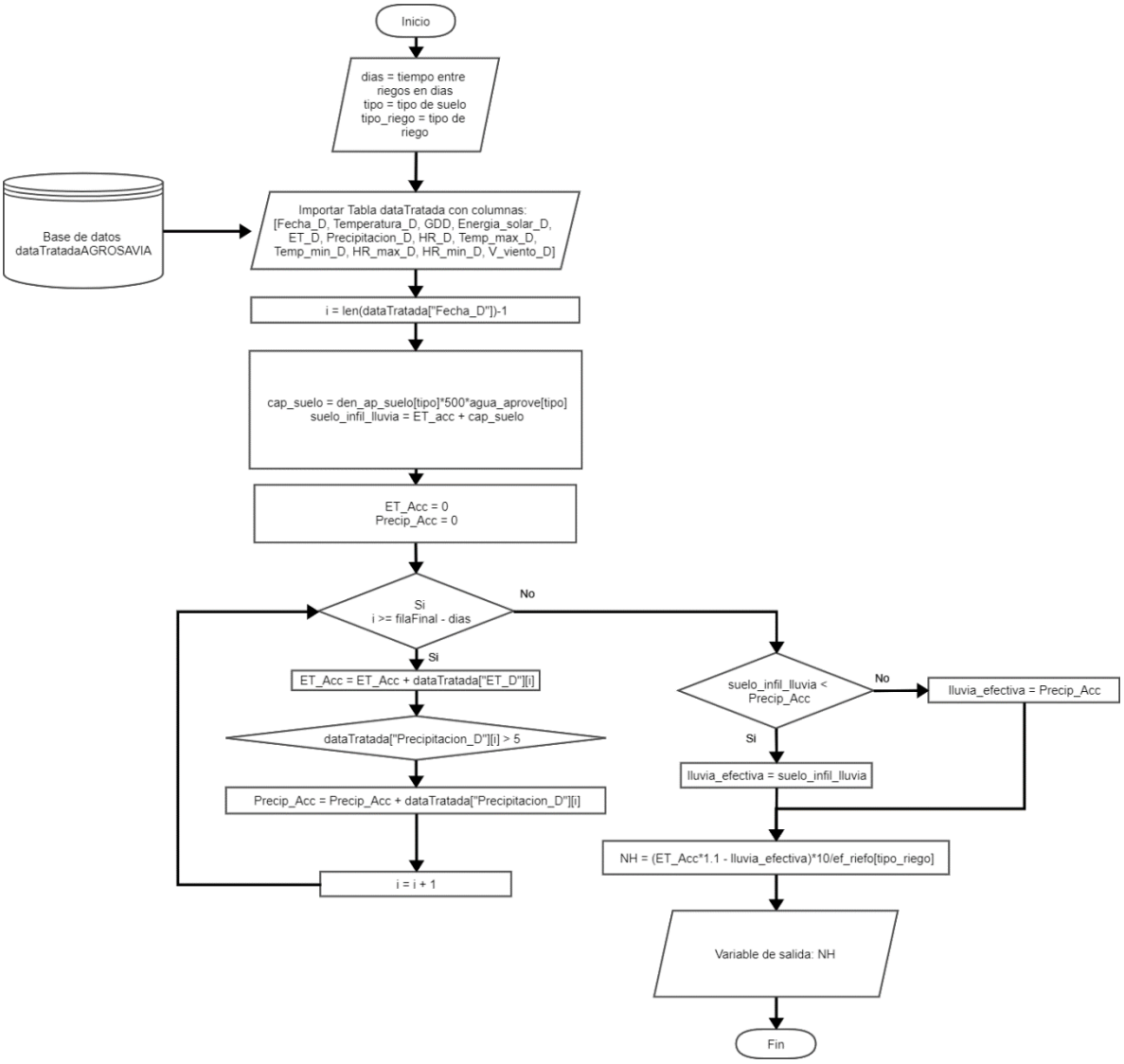


Figura 28: Algoritmo cálculo de la necesidad - estimación la cantidad de agua para evitar el estrés hídrico AGROSAVIA

III. Desarrollo de la aplicación web ahora app

5. ¿Qué es Flask?

Flask es un marco de aplicación web escrito en Python. Fue desarrollado por Armin Ronacher, quien dirigió un equipo de personas de Python llamado Poocco. Flask se basa en el kit de herramientas Werkzeg, WSGI y el motor de plantillas Jinja2, ambos proyectos de Pocco (Grinberg, 2014).

5.1. WSGI: La Interfaz de puerta de enlace del servidor web (Web Server Gateway Interface, WSGI) se ha utilizado como estándar para el desarrollo de aplicaciones web Python. WSGI es la especificación de una interfaz común entre servidores web y aplicaciones web (Dwyer et al., 2017).

5.2. Werkzeug: es un conjunto de herramientas WSGI que implementa solicitudes, objetos de respuesta y funciones de utilidad. Esto permite construir un marco web sobre él. El marco Flask utiliza Werkzeg como una de sus bases (Copperwaite y Leifer, 2015).

5.3. Jinja2: es un motor de plantillas popular para Python. Un sistema de plantillas web combina una plantilla con una fuente de datos específica para representar una página web dinámica (Grinberg, 2014).

Esto le permite pasar variables de Python a plantillas HTML como esta:

```
<div class="alert alert-success alert-dismissible fade show" role="alert">
    <strong>Se han acumulado {{valor1}} GD desde la fecha {{valor2}} y le ha tomado {{valo
r3}} semanas</strong>
  </div>
```

5.4. Ventajas De Uso: A diferencia del marco de Django, Flask es muy explícito, lo que aumenta la legibilidad, aunque sea un microframework no significa que toda la aplicación deba estar dentro de un solo archivo de Python. Si la aplicación crece se puede usar muchos archivos para programas más grandes, y manejar de mejor manera la complejidad.

Flask es uno de los frameworks web más populares, lo que significa que está actualizado y es moderno. Puede ampliar fácilmente su funcionalidad. Y permite escalar las aplicaciones a formas más complejas.

6. Esquema de desarrollo

La aplicación se ha trabajado con el Modelo Vista Controlador, para la vista se han usado plantillas HTML, mientras que el modelo y controlador se han trabajado con archivos

Python.

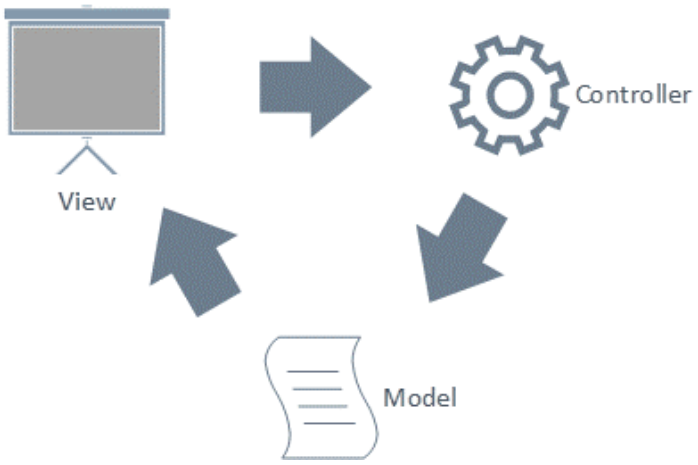


Figura 29: Modelo Vista Controlador

Fuente: Elaboración propia

6.1. Vistas

Se han creado diferentes vistas (archivos html) para la aplicación, dentro de las más importantes están:

- Index.html: Página principal de la aplicación web, donde están configurados los enlaces de las funcionalidades del sistema: Calcular hojas, Grados día Backward y Grados día Forward. En esta vista también se cargan los logos de las entidades participantes del proyecto.

- viewCalcularHojas.html: En esta vista se configura los mensajes del cálculo de número de hojas para los últimos 28 y 14 días respectivamente. Además, se ha programado un formulario para agregar el número de hojas reales.

```
<div class="alert alert-success alert-dismissible fade show" role="alert">
    <strong>En los últimos 14 días han crecido: {{valor1}} hojas potenciales</strong>
  </div>

  <div class="alert alert-success alert-dismissible fade show" role="alert">
    <strong>En los últimos 28 días han crecido: {{valor2}} hojas potenciales</strong>
  </div>
```

- viewBackward.html: En esta vista se configura los mensajes de la funcionalidad de los grados día en retroceso.

```
<div class="alert alert-success alert-dismissible fade show" role="alert">
```

```
    <strong>Se han acumulado {{valor1}} GD desde la fecha {{valor2}} y le ha tomado {{valo
r3}} semanas</strong>
  </div>
```

- viewForward.html: En esta vista se configura el mensaje de la funcionalidad de grados día en adelante. Además, también se usa la librería de JavaScript llamada Chart.js para lograr mostrar gráficos de la temperatura promedio, humedad relativa y grados día acumulado.

```
<div class="alert alert-success alert-dismissible fade show" role="alert">
    <strong>Se han acumulado {{valor1}} GD desde la fecha {{valor2}} y la fecha estimada p
ara completar los 900 GD es
     {{valor3}} </strong>
  </div>
```

```
<script type="module">
       function totalCasesChart(ctx) {
        const chart = new Chart(ctx, {
         type: 'line',
         data: {
          labels: {{ fechas| safe}},
       datasets: [
        {
         label: "Temperatura promedio",
         data: {{ tempPromedio| safe}},
        fill: false,
        borderColor: 'blue',
        lineTension: 0.1
            },

       ]
         },

       options: {
        scales: {
         xAxes: [{
          gridLines: {
           display: false,
          }
         }]
        },
        responsive: true
       },
       legend: {
```

```
    position: 'bottom',
      labels: {
      padding: 20,
        boxWidth: 15,
          fontFamily: 'system-ui',
            fontColor: 'black',
        }
    },
    layout: {
      padding: {
        right: 50
        }
    },
    tooltips: {
      backgroundColor: '#0584f6',
        titleFontSize: 20,
          xPaddding: 20,
            yPadding: 20,
              bodyFontSize: 15,
                bodySpacing: 10,
                  mode: 'x'
    },
    elements: {
      line: {
        borderWidth: 8,
          fill: false
      },
      point: {

        radius: 6,
          borderWidth: 4,
            backgroundColor: 'white',
              hoverRadius: 8,
                hoverBorderWidth: 4
      }
    }
      });
    }
    function renderCharts() {
      const ctx = document.querySelector('#chart1').getContext('2d');
      totalCasesChart(ctx);

    }
```

```
    renderCharts();

  </script>
```

6.2. Controlador

El controlador de la aplicación Web es un archivo Python llamado App.py, dentro de este archivo se han creado todas las rutas que debe tener la aplicación para navegar entre las diferentes vistas y funcionalidades. Dentro de los métodos más importantes tenemos.

- Función run: Función de inicio de la aplicación, debe configurarse el puerto y el modo de depuración para iniciar el servidor web.

```
if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

- Función Index: Ruta que activa la pantalla principal de la aplicación que es el index.html.

- Función ViewCalcularNroHojas: Función que enlaza la vista del archivo viewCalcularHojas.html con la función de acceso de base de datos que está configurado en el modelo de la aplicación.

```
@app.route('/ ViewCalcularNroHojas)
def ViewCalcularNroHojas ():
    valor1,valor2 = functions.NroHojas()

    valor1 = round(valor1, 1)
    valor2 = round(valor2,1)

    cur = mysql.connection.cursor()
    cur.execute('SELECT * FROM NROHOJAS')
    datos = cur.fetchall()

    return render_template(ViewCalcularNroHojas.html', datos = datos, valor1 = valor1, val
or2 = valor2)
```

- Función AddNroHojas: Función que recibe los parámetros del formulario de hojas reales para ingresarlos en la base de datos.

- Función ViewForward: Función que enlaza la vista del archivo viewForward.html con la función de cálculos ubicada en el modelo.

```
@app.route('/viewForward', methods =['POST'])
def viewForward():
  if request.method == 'POST':
   fecha = request.form['fecha']

   valor1,valor2, valor3 = functions.GDA_forward(fecha)
   valor1 = round(valor1)
   data = functions.Graficas(fecha)

  fechas = [row[0] for row in data]

   tempPromedio = [row[1] for row in data]
   humedad = [row[2] for row in data]
   gradosDia = [row[3] for row in data]


   return render_template('viewforward.html', valor1 = valor1, valor2 = valor2, valor3 = va
lor3, fechas = fechas, tempPromedio= tempPromedio, humedad = humedad, gradosDia=gr
adosDia, datosCompletos = data)
```

- Función ViewBackward: Función que enlaza la vista del archivo viewBackward.html con la función de cálculos ubicada en el modelo.

```
@app.route('/viewBackward')
def viewBackward():
   valor1,valor2, valor3 = functions.GDA_backward()
   valor1 = round(valor1)
   return render_template('viewbackward.html', valor1 = valor1, valor2 = valor2, valor3 = v
alor3)
```

6.3. Modelo

El modelo de la aplicación Web es un archivo Python llamado Functions.py, dentro de este archivo se han creado las funcionalidades de acceso a base de datos para realizar los cálculos de las funcionalidades.
Las funciones son las siguientes:

- Función NroHojas: Función que calcula el número de hojas accediendo a los datos de la base de datos ubicada en el servidor de la Universidad de Piura.

```
def NroHojas():
```

```
  sqlEngine = create_engine("conexión")
  dbConnection = sqlEngine.connect()

  try:
     df_data = pd.read_sql("select * from VARIABLES_DIA_ASPROBO", dbConnection)
     print("Se importo la base de datos correctamente.")
  except ValueError as vx:
     print(vx)
  except Exception as ex:
     print(ex)
  finally:
     dbConnection.close()
     print("MySQL conexion terminada extraccion de dato")
  GD_calculo = df_data['GDD']
  fecha = df_data['Fecha_D']

 GDA14 = np.array(GD_calculo[-14:]).sum()

  GDA28 = np.array(GD_calculo[-28:]).sum()
  nHojas14 = GDA14/108
  nHojas28 = GDA28/108
  return nHojas14, nHojas28
```

- Función GDA_Backward: Función que calcula los grados día en retroceso

```
def GDA_backward():
  sqlEngine = create_engine("conexion”)
  dbConnection = sqlEngine.connect()
  try:
     df_data = pd.read_sql("select * from VARIABLES_DIA_ASPROBO", dbConnection)
     print("Se importo la base de datos correctamente.")
  except ValueError as vx:
     print(vx)
  except Exception as ex:
     print(ex)
  finally:
     dbConnection.close()
     print("MySQL conexion terminada extraccion de dato")

  GD_calculo = df_data['GDD']
  fecha = df_data['Fecha_D']

  print('Calculo de GDA al día de hoy.')
```

```
  GDA = 0
  i = len(fecha)
  while (GDA<900 and i>=1):
     GDA += GD_calculo[i-1]
     i += -1
  nSemanas = int((len(fecha)-i-1)/7)
  print("Se han acumulado","{:.2f}".format(GDA), "desde la fecha", fecha[i], "al dia de hoy
.")
  print("Le ha tomado", nSemanas, "semanas.")
  return GDA, fecha[i], nSemanas
```

- Función GDA_Forward: Función que calcula los grados día hacia adelante recibiendo la fecha de floración.

```
def GDA_forward(fechaWeb):
  sqlEngine = create_engine("conexion")
  dbConnection = sqlEngine.connect()

  try:
     df_data = pd.read_sql("select * from VARIABLES_DIA_ASPROBO", dbConnection)
     print("Se importo la base de datos correctamente.")
  except ValueError as vx:
     print(vx)
  except Exception as ex:
     print(ex)
  finally:
     dbConnection.close()
     print("MySQL conexion terminada extraccion de dato")
  GD_calculo = df_data['GDD']
  fecha = df_data['Fecha_D']

  from datetime import datetime, timedelta
  print('Estimación fecha de cosecha.')
  print("fecha en fuctions", fechaWeb)
  fec = fechaWeb
  i = df_data.loc[df_data.Fecha_D == fec].index[0]
  print(i)
  GDA = 0
  cont = 0

  while (i<=len(fecha)-1):
     cont += 1
```

```
    GDA += GD_calculo[i]
    i += 1
    if GDA > 900:
        break
fec = datetime. strptime(fec, '%d/%m/%Y')  #A DESCOMENTAR

if GDA < 900:
    GDA_restantes = 900 - GDA
    promGDA = GDA/cont
    estimacion = GDA_restantes/promGDA
    fec_final = fec + timedelta(estimacion)
    fec_str = fec.strftime("%d/%m/%Y")
    fec_final_str = fec_final.strftime("%d/%m/%Y")
    print("Se han acumulado","{:.2f}".format(GDA), "desde la fecha", fec_str)
    print("Fecha estimada para completar los 900 GDA:", fec_final_str)
else:
    print("Los 900 GDA se completaron en la fecha", fecha[i])
    fec_final = datetime. strptime(fecha[i], '%d/%m/%Y')

return GDA, fec.date(), fec_final.date()
```

7. Base de datos de la aplicación Web

7.1. Definición de base de datos

Una base de datos es un conjunto de datos organizados y conexos entre sí, los cuales son recolectados y utilizados por los sistemas de información.

Las bases de datos brindan los componentes necesarios a las aplicaciones para apoyar en la toma de decisiones. Un software explota la información contenida en las bases de datos y eso permite lograr ventajas considerables. Es por eso que es importante conocer el funcionamiento y estructura de una base de datos para saber manejarla.

7.2. Ventajas del uso de base de datos

- Permite globalizar la información ya que diferentes usuarios pueden considerarla como un recurso colectivo.
- Permite compartir información ya que múltiples usuarios y sistemas pueden utilizar la misma base de datos para cruzar información que necesiten.
- Permite mantener la integridad en la información ya que se debe almacenar la información correcta sin duplicidad.

- Para un buen modelo de base de datos es importante la independencia de los mismos. Esto implica que los registros sean abstractos al programa o software que interactúa con él; es decir, que se puedan hacer cambios en la información sin hacer cambios en el software desarrollado.

7.3. Diagrama entidad – relación para el sistema de la plataforma virtual

La base de datos para la aplicación web está conformada por las siguientes tablas (Figura 30):

- "pais"
- "ciudad"
- "estaciones"
- "variables_climaticas"
- "variables_dia"
- "nro_hojas"

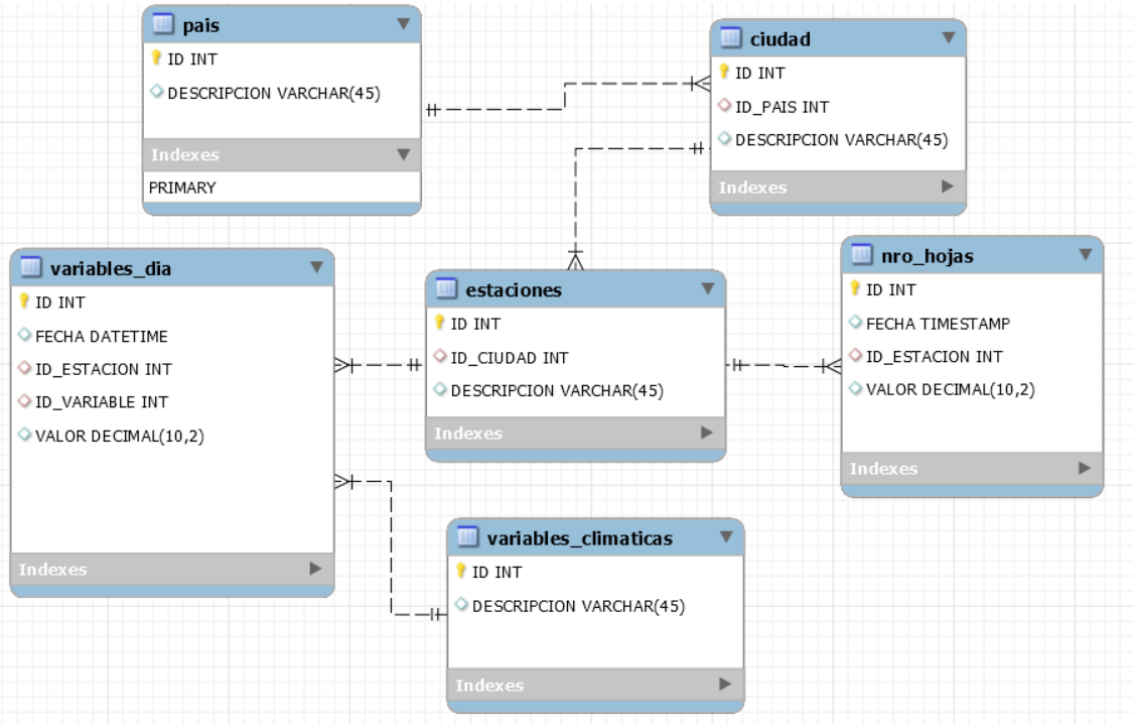


Figura 30: Diagrama entidad - relación
Fuente: Elaboración propia.

7.4. Descripción de las tablas de la base de datos diseñada

- "Tabla país": En esta tabla se registran los países involucrados en la aplicación. Ejemplo: Perú, Colombia, República Dominicana.

- “Tabla ciudad”: En esta tabla se guardan los datos de las ciudades donde están instaladas las estaciones y se enlazan con su respectivo país.
- “Tabla estaciones”: Esta tabla registra las estaciones con las que trabaja el software y se les asigna su respectiva ciudad.
- “Tabla variables climáticas”: En esta tabla se registran todas las variables climáticas con las que trabaja la aplicación. Ejemplo: Temperatura, humedad, radiación, etc.
- “Tabla variables_dia”: Tabla que registra el valor de cada variable climática medida, la fecha en la que se tomó y de que estación se obtuvo ese valor.
- “Tabla nro_hojas”: En esta tabla se registran el valor de hojas reales que ingresa el usuario según la estación que lo está monitoreando.

8. SERVIDOR WEB

Para la primera versión se ha usado con servidor Web la plataforma Heroku que es una plataforma en la nube como servicio (PaaS) basada en contenedores que permite construir, desplegar, monitorear y escalar aplicaciones de forma rápida.

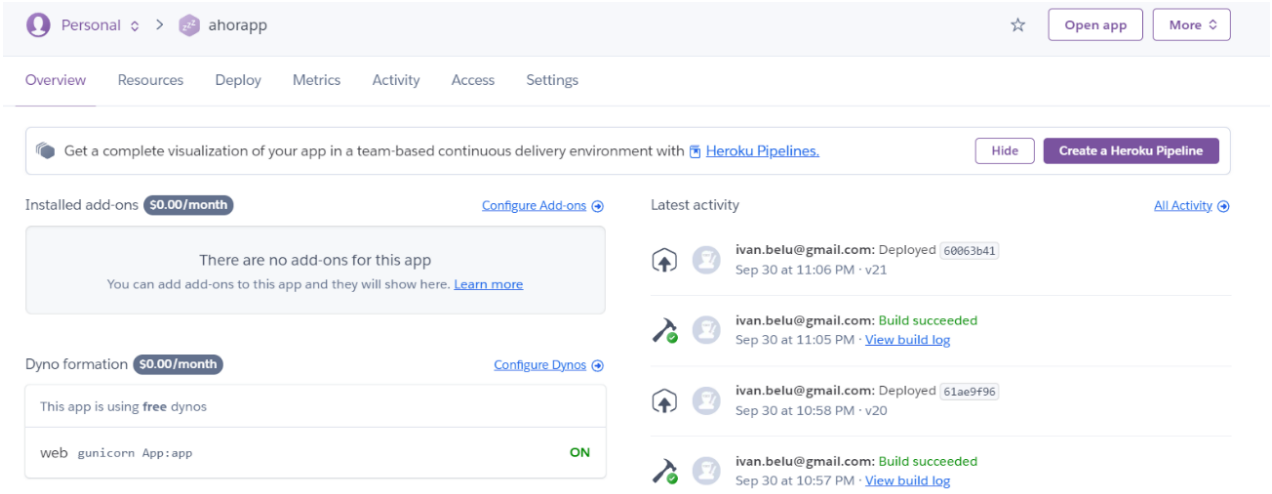


Figura 31: Panel de control Heroku

Fuente: Elaboración propia

Conclusiones:

- Se mostró la experiencia que tiene la Universidad de Piura (UDEP), en la participación de proyectos agropecuarios con pequeños productores, recolección y procesamiento de datos climatológicos, y desarrollo de software e inteligencia artificial.

- El aplicativo °AHoRa integra cinco (5) ecuaciones en temas clave en los que los productores requieren apoyo para la toma de decisiones sobre el manejo del cultivo. En el presente documento se ilustraron los algoritmos empleados para desarrollar estas ecuaciones en Python y se informó acerca del proceso empleado desde la extracción de los datos de los servidores de Davis, su almacenamiento en una base de datos propia, y el desarrollo de la interfaz web.

Referencias Bibliográficas

Copperwaite, Matt, and Charles Leifer. 2015. Learning Flask Framework : Build Dynamic, Data-Driven Websites and Modern Web Applications with Flask / Matt Copperwaite, Charles Leifer. 1st edition. Birmingham: Packt Publishing. Print.

FAO. 2020. En Guinea Ecuatorial se emprenden acciones para la reducción del impacto ambiental. En línea: https://www.fao.org/guinea-ecuatorial/noticias/detail-events/en/c/1259988/

Gareth Dwyer, Shalabh Aggarwal, and Jack Stouffer. 2017. Flask: Building Python Web Services. Packt Publishing.

Guarín, G. (2011). Impacto de la variabilidad climática en la producción de banano en el Urabá Antioqueño. Universidad Nacional de Colombia. Extraído de, http://cort.as/-MLsu

Higuera, I. (2015). Bananos y plátanos, frente al cambio climático. Extraído de, http://cort.as/- MKp_

Miguel Grinberg. 2014. Flask Web Development: Developing Web Applications with Python (1st. ed.). O'Reilly Media, Inc.

Távara, H. M. (2020). Efectos del cambio climático en la productividad del banano orgánico en el Valle del Chira - Sullana - Piura. [Dissertation, Universidad Nacional de Piura, Piura, Perú].

Yela. Y, Boza. J, Baquedano. L, Fierro. J, Rivas. K y Quiñonez. M. 2016. Efectos del cambio climático en la producción agrícola del Banano en el Cantón Valencia. Revista Caribeña de Ciencias Sociales. En línea: https://www.eumed.net/rev/caribe/2016/09/banano.html

Instituciones participantes

Secretaría Técnica Administrativa

FONTAGRO

Con el apoyo de:

BID
Banco Interamericano de Desarrollo

IICA

www.fontagro.org
Correo electrónico: fontagro@fontagro.org