Universidad de las Ciencias Informáticas
Vertex, Automática
Facultad de tecnologías Iterativas

Sistema para la visualización de datos de estaciones meteorológicas de bajo costo mediante Internet de las cosas.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Daimarilis Guillot Reyes

Tutora: Tutor: Ing. Julio Alberto Leyva Durán.

La Habana, 2024

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein

Dedicatoria

A mi hijo Dilan Dailier.
A mis padres y familiares.
A mis grandes amigos por el apoyo.

Agradecimientos

A mis padres en especial a mi madre Olivia Reyes Olivares por estar en cada momento de este largo camino por apoyarme y ser abuela y madre a la vez para que yo continúe con mis sueños de ingeniera para ella todo el mérito y el amor del mundo. A mi pequeño bebe de 4 años porque saber que existe y necesita de mí, me ha impulsado a seguir adelante para cumplir con mi papel de madre porque he estado en pequeños momentos de su vida.

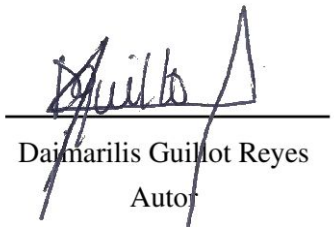
A todas esas personas que me han acompañado en este largo camino que para mí ha sido un tín largo agradecer por tantos momentos que han formado parte de mi vida desde 2016 llegue a esta universidad por algunos motivos sigo aquí pero no me rendí jamás, gracias por su apoyo y comprensión a los que están y por los que alguna razón ya no están en esta universidad. Mi gran amiga y compañera de toda la vida que aún está a mi lado Mardelis gracias por ser parte de mi vida durante toda la etapa de escuela y universidad más de 12 años de amistad.

A mis amistades que he conocido en el camino que de hecho son varias gracias por todo darles, laura, vivi, yami que por varios motivos ya no está con nosotros al igual que felipe, gretel, lisbeth, zoreidis, michel, alejandro meriño que fue un gran apoyo en gran parte de mi proceso me apoyó demasiado y me brindo su ayuda en varios momentos gracias por tanta comprensión y otro muchos más que han cambiado el rumbo de su vida por varias razones, arian, jaydy, danieska, adachelis, javier, gretel amiguis y otros muchos más que han formado parte de esta bella familia UCI. Gracias a mi novio reydel por tanta comprensión y apoyarme siempre... verdaderamente gracias a todos a mis compañeros de aula, de departamento y profesores durante todos estos años. Agradecer también a mi tutor por tanta dedicación en este proceso.

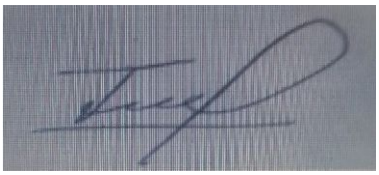
Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los 3 días del mes de diciembres del año 2024.



Daimarilis Guillot Reyes
Autor



Tutor: Ing. Julio Alberto Leyva Durán.
Tutora

Resumen

RESUMEN

En Cuba, el Instituto de Meteorología (Insmet) es el responsable de proporcionar información meteorológica oportuna sobre el estado y el comportamiento futuro de la atmósfera. Esta información tiene como objetivo salvaguardar la vida humana y reducir las pérdidas materiales debido a los desastres naturales de origen meteorológico, contribuyendo directamente al bienestar de la comunidad y al desarrollo sostenible. La investigación, cuyos resultados se describen en este informe, tuvo como propósito desarrollar un sistema para la visualización de datos de meteorológicos en la estación meteorológica de bajo costo. El desarrollo del mismo fue guiado por la metodología de desarrollo de software XP, cumpliendo con cada una de sus fases y generando todos los artefactos. Para la implementación de este se utilizó como marco de trabajo web Django con su lenguaje de programación Python utilizando como protocolo de comunicación MQTT. Para validar que la solución cumpliera con los requisitos definidos por el cliente, se aplicaron pruebas unitarias y pruebas de aceptación.

Palabras clave: estación, sistema, meteorología,visualización, MQTT.

Índice general

Índice de figuras

Índice de tablas

Introducción

Con el desarrollo de las TIC surgieron las aplicaciones web que son software de tipo cliente-servidor que permite realizar funciones determinadas en Internet, estas aplicaciones evolucionan a diario haciendo disimiles aportes a la sociedad que generan gran impacto en las comunicaciones, los negocios, la innovación y la educación.

En la actualidad la información meteorológica se encuentra mucho más accesible gracias al desarrollo de la tecnología y las comunicaciones que permiten la interconexión de la información de las estaciones meteorológicas de forma global donde la Organización Meteorológica Mundial (OMM) creada por Organización de las Naciones Unidas asegura la cooperación entre los servicios meteorológicos internacionales.

En Cuba el Instituto de Meteorología (Insmet) es el encargado de suministrar la información meteorológica oportuna sobre el estado y comportamiento futuro de la atmósfera. Esta información está dirigida a velar por la seguridad de la vida humana y a reducir las pérdidas de bienes materiales ante desastres naturales de origen meteorológico, contribuyendo directamente al bienestar de la comunidad y al desarrollo sostenible.

Existen en Cuba un total de 43 estaciones meteorológicas distribuidas por todo el país que contribuyen al pronóstico informativo sobre el estado del tiempo. Dicho pronóstico tiene un carácter público, transmitiéndose por la radio y televisión. Para realizar este pronóstico es preciso obtener continuamente la información sobre las condiciones existentes en la superficie y la atmósfera para posterior procesamiento en un modelo predicción.

Como resultado se simula la atmósfera, permitiendo la predicción del posible estado de la misma dentro de un breve lapso de tiempo, pero el comportamiento de los parámetros atmosféricos es complejo y se encuentran en contante fluctuación; por tanto, a un pronóstico proyectado en un futuro más lejano podrían aparecer fenómenos que no fueron contemplados y esos pequeños cambios que se producen en la atmósfera pueden tener grandes efectos y alterar toda la predicción. Por lo que los pronósticos pierden confiabilidad con la proyección del tiempo. Los mismo son más precisos en áreas más pequeñas debido que las estaciones meteorológicas pueden registrar mayor nivel de detalle, y en ocasiones en lugares donde existan microclimas es necesario registrar esos valores para dar un pronóstico más confiable.

La ubicación topografía donde se encuentra la Universidad de las Ciencias Informáticas (UCI) es favorecida por una variada humedad, debido a la cobertura de árboles, el relieve de la zona y la radiación solar. Los microclimas urbanos provocan variaciones en la calidad del aire, el agua y que aumenten las olas de calor. Esto tiene un impacto muy dañino para la salud de las personas: dificultades respiratorias,

deshidratación, cansancio y agotamiento. Teniendo en cuenta estos factores que provocan los microclimas, se necesita conocer con mayor exactitud los parámetros meteorológicos en lugares donde las estaciones de meteorología se encuentren distantes y sea necesario conocer con exactitud los valores de los parámetros meteorológicos; como necesidad ante el cambio climático que nos permita evaluar el comportamiento de los microclimas, tanto en zonas urbanas o rurales así tomar decisiones precisas y efectiva que permita tomar medidas previsoras eficiente ante la ocurrencia de un fenómeno natural desfavorable.

Esta situación se podría resolver obteniendo en el mercado internacional estaciones meteorológicas para cada lugar de interés, pero por cuestiones económicas de afectaciones por el bloqueo del gobierno de Estado Unidos resulta inaccesible para el estado cubano. En la UCI, el Centro de Entornos Interactivos, tiene dentro de sus líneas de desarrollo la Automática Aplicada, en esta línea de investigación se desarrolla un prototipo de Tarjeta De adquisición de Datos (DAQ), que se caracteriza como un agente tipo software y hardware que puede ejecutar rutinas de control de baja complejidad y la medición de señales analógicas y digitales mediante sensores específicos para la medición de señales ambientales.

Actualmente el mecanismo de intercambio de información entre las DAQ y el usuario se realiza en texto plano, mediante una consola de comandos, sin una interfaz de usuario adecuada para mostrar las variables correspondientes de los sensores del medio ambiente, además no posee la capacidad semántica para intercambiar los datos a través de un conjunto común de formatos de intercambio, y no se garantiza la seguridad y la flexibilidad de manejar protocolos de comunicación en el dispositivo DAQ.

Teniendo en cuenta la situación problemática antes descrita, se identifica como **problema de investigación:** ¿Cómo gestionar la visualización e intercambio de información de variables meteorológicas?, como **objeto de estudio** se define los procesos de visualización e intercambio de información mediante protocolo MQTT, y el **campo de acción:** Sistema para la visualización de datos de estaciones meteorológicas de bajo costo mediante Internet de las cosas. Con el fin de solucionar el problema planteado se define como **objetivo general:** Desarrollar un Sistema para la visualización e intercambio de información de variables meteorológicas de bajo costo mediante Internet de las cosas (IoT).

Posibles resultados: Sistema de gestión Web para la visualización e intercambio de información con tarjetas de adquisición de datos meteorológicos mediante plataformas de hardware y software libre e IoT.

Marco tareas a cumplir: Para dar cumplimiento al objetivo propuesto se proponen las siguientes tareas investigativas:

- Estudio del estado del arte del tema en cuestión.
- Generación de los artefactos relacionados con el análisis y diseño de la solución propuesta.
- Diseño e implementación de la solución propuesta.
- Validación de la propuesta de solución.

Para obtener los conocimientos necesarios que hagan posible el cumplimiento del objetivo trazado, se lleva a cabo una investigación en las que se utilizan algunos de los métodos científicos existentes, tanto teóricos como empíricos.

Métodos teóricos: Modelación: es empleada en la representación mediante diagramas de las características, procesos y componentes de la solución propuesta, así como la relación existente entre ellos.

Analítico-sintético: se emplearon en el proceso de análisis documental y revisión bibliográfica, con el objetivo de extraer las ideas esenciales que permitirán fundamentar desde el punto de vista teórico la investigación, así como la propuesta que se realiza.

Hipotético-deductivo: para guiar la investigación desde el planteamiento del problema hasta la verificación de la solución a partir de las validaciones, orientando la secuencia lógica de las tareas que se realizaron.

Métodos empíricos:

Entrevista: se emplea en encuentros con el cliente para conocer la necesidad del desarrollo de la propuesta de solución, definir sus funcionalidades e identificar a la vez particularidades de cada usuario y las restricciones que se imponen.

La investigación consta con una estructura organizativa en el documento que incluye una introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos. A continuación, se detalla el contenido propuesto para cada uno de los capítulos:

Capítulo 1: Fundamentos y Referentes Teórico-Metodológicos del proceso de visualización e intercambio de información con la DAQT. Este capítulo se centra en los aspectos teóricos y conceptos fundamentales relacionados con el objeto de estudio y el campo de acción de la investigación. Se analizan las soluciones existentes tanto en el ámbito internacional como en el nacional y se describe el entorno de desarrollo a partir de la fundamentación del uso de la metodología, tecnologías y herramientas propuestas para el desarrollo de la propuesta de solución.

Capítulo 2: Análisis y Diseño del sistema e Implementación. En este capítulo se llevará a cabo la caracterización de la propuesta de solución. Se especifican los requisitos funcionales y no funcionales a partir de la metodología seleccionada, y se realiza el diseño ingenieril donde se describen los patrones de diseño definidos como buenas prácticas durante el ciclo de desarrollo del software, la realización del modelado de diagramas, los elementos fundamentales del diseño y de la arquitectura que se deben tener en cuenta para llegar propuesta solución.

Capítulo 3: Validación del sistema. Se analizan los procesos referentes al desarrollo de la solución y las pruebas realizadas para validar el correcto funcionamiento del sistema. Finalmente, se presentan las Conclusiones, Recomendaciones, Referencias Bibliográficas y Anexos derivados de la investigación.

CAPÍTULO 1

Fundamentación Teórica

En el presente capítulo se precisan un conjunto de elementos para conformar el marco teórico relacionado con el desarrollo de estaciones meteorológica de bajo costo. Dentro de los cuales se analizan las formas de adquisición de variables y tomando como referencias diferentes fuentes de información. Además, se identifican, analizan y se seleccionan las tecnologías y herramientas existentes para la conformación del prototipo de Estación Meteorológica de bajo costo.

Fundamentación teórica

Para lograr una comprensión total del presente trabajo, a continuación se concretan aspectos relevantes para la investigación.

1.1. Meteorología

La meteorología es la ciencia encargada del estudio de la atmósfera, de sus propiedades y de los fenómenos que en ella tienen lugar, los llamados meteoros. El estudio de la atmósfera se basa en el conocimiento de una serie de magnitudes, o variables meteorológicas, como la temperatura, la presión atmosférica o la humedad, las cuales varían tanto en el espacio como en el tiempo.**(rodriguez2004)**

El estudio de la meteorología no solo nos permite comprender los fenómenos atmosféricos que rigen nuestro clima, sino que también nos brinda las herramientas necesarias para anticipar y mitigar los impactos de eventos extremos, promoviendo así una convivencia más armónica entre la humanidad y su entorno natural.Se ocupa de describir y predecir el tiempo y el clima, que son dos conceptos relacionados pero distintos:

El clima es el conjunto de condiciones meteorológicas que caracterizan a un lugar determinado del planeta, es decir, es la tendencia meteorológica normal de un sitio específico. En cambio, el tiempo se refiere al estado particular de la atmósfera (en realidad la tropósfera, su capa más baja) en un lugar y un momento determinado. En resumen, el tiempo en un lugar se refiere a cómo está la atmósfera en ese momento; mientras que el clima de un lugar se refiere a cómo suele estar la atmósfera en las distintas épocas del año.**(clima2023)**

Entre los elementos del clima se encuentran la Temperatura, la presión atmosférica, el viento, la humedad y la precipitación.

1.1.1. Microclimas

Los microclimas son variaciones locales del clima que se producen en áreas relativamente pequeñas, a menudo debido a factores como la topografía, la vegetación, la urbanización y el uso del suelo. Estos microclimas pueden influir en las condiciones ambientales, afectando la temperatura, la humedad, el viento y la precipitación en un área específica.**(clima2023)**

Factores que Influyen en los Microclimas

- Topografía: Las montañas, valles y cuerpos de agua pueden crear variaciones significativas en el clima. Por ejemplo, las laderas orientadas al sol pueden ser más cálidas y secas que las laderas sombreadas.
- Vegetación: Los bosques, parques y jardines pueden moderar las temperaturas y aumentar la humedad localmente.
- Urbanización: Las áreas urbanas tienden a ser más cálidas que sus alrededores rurales debido al efecto de isla de calor urbano, donde las superficies de concreto y asfalto absorben y retienen el calor.
- Cuerpos de Agua: Los lagos y ríos pueden influir en las temperaturas locales, proporcionando un efecto moderador.

Ejemplos de Microclimas

Zonas Urbanas: Las ciudades suelen tener microclimas más cálidos debido al aumento de superficies impermeables y la actividad humana.

Valles: Los valles pueden acumular aire frío durante la noche, creando microclimas más fríos que las áreas circundantes.

Áreas Costeras: La proximidad al mar puede moderar las temperaturas, creando microclimas más suaves.

1.1.2. Importancia de los Microclimas

Agricultura: Los microclimas pueden ser cruciales para la agricultura, ya que ciertas plantas pueden prosperar en condiciones específicas que difieren del clima general de la región.

Biodiversidad: Los microclimas pueden crear hábitats únicos que favorecen la diversidad biológica. Planificación Urbana: Comprender los microclimas es esencial para el diseño urbano sostenible y la gestión ambiental.

1.2. Estación meteorológica

Una estación meteorológica es una instalación equipada con instrumentos y tecnologías diseñadas para medir y registrar diversas variables atmosféricas, como temperatura, humedad, presión atmosférica, veloci-

dad y dirección del viento, precipitación, entre otros. Estas estaciones son fundamentales para la observación del clima y el tiempo, así como para la investigación meteorológica y climática. **(melendez2022)**

Figura 1.1. Estación de meteorológica (CAMPBELL SCIENTIFIC SPAIN, 2023).

1.2.1. Importancia de las estaciones meteorológica

Desde el inicio el hombre ha debido crear estrategias para adaptarse a las condiciones climáticas. Sin embargo, el desarrollo económico y la urgente necesidad de optimizar los recursos productivos, entre los que se incluye el clima, la obligación de estudiar y entender mejor su comportamiento por medio de la observación sistemática y permanente de variables que lo integran, y gracias a la implementación y uso de estaciones meteorológicas manuales y automáticas **(maldonado2018)**

La información meteorológica resulta de gran utilidad en distintas disciplinas como la agronomía y la hidrología, entre otras. La observación de variables y fenómenos meteorológicos se lleva a cabo en Estaciones Meteorológicas Convencionales (EMC) asistidas por un observador capacitado.

En los últimos años, el uso de Estaciones Meteorológicas Automáticas (EMA) ha experimentado un incremento significativo. La OMM las define como "las estaciones en las cuales las observaciones son realizadas y transmitidas automáticamente".

Las estaciones meteorológicas no solamente nos permiten hacer un seguimiento en tiempo real, sino que son la base de datos que nos facilita cualquier tipo de información meteorológica de tiempo atrás, con lo que podemos saber cómo se ha comportado la atmósfera a partir de los registros. Pero para tener una representación lo más acercada posible a la realidad de los eventos meteorológicos que ha habido no sólo basta tener

estaciones fiables, sino que debe haber una cantidad y distribución adecuada de las mismas.**(alggar2018)**

1.2.2. Tipos de estaciones meteorológicas

Las estaciones meteorológicas automáticas y de bajo costo son, muy populares entre los aficionados y usuarios domésticos debido a su accesibilidad.

1.2.3. Estación meteorológica automatizada

Una estación Meteorológica automatizada es un sistema autónomo y automático formado por un conjunto de sensores de medición de variables meteorológicas y equipos eléctricos y electrónicos montados sobre una estructura de soporte y conectados a un sistema de adquisición de datos con capacidades de almacenamiento, cálculos y transmisión de información a oficinas centrales. La utilización de las estaciones meteorológicas automatizadas es el registro datos de forma continua y permitir realizar mediciones en intervalos de tiempo menores que tomando los datos manualmente, además de ser totalmente autónomo al no necesitar intervención humana.**(alggar2018)**

Según la Organización Mundial de Meteorología (OMM), una estación meteorológica automática (EMA), corresponde al equipamiento con el cual las observaciones se generan y transmiten automáticamente. Una EMA está compuesta de dispositivos de medición, sensores, que captan las variables meteorológi-cas, un sistema de almacenamiento y procesamiento de los datos, datalogger, un sistema de comuni-caciones para enviar los datos a un servidor en Internet (vía celular o satelital) y una unidad para alma-cenar y entregar energía para el funcionamiento de los equipos electrónicos (panel solar, regulador de voltaje y batería recargable).**(alggar2018)**

1.2.4. Estación meteorológica de bajo costo

Las estaciones meteorológicas y sensores profesionales suelen tener una estructura compleja para lograr una alta precisión por lo que el precio de estos productos suele ser sumamente elevado. Como consecuencia el monitorio de estos sistemas meteorológico una vez implementado conlleva a un costo adicional; siendo no factible del punto de vista económico. Siendo necesario la búsqueda de opciones que permita tener las mismas prestaciones a un costo más accesibles; dentro las opciones actuales para el desarrollo de una estación de meteorología se encuentran el uso de plataformas hardware y software libre entre las que se encuentran.**(bareno2011)**

El empleo de computadores de placa única (SBC, Single Board Computers, por sus siglas en inglés), como el conocido Raspberry pi. El uso de elementos de la plataforma Arduino. Posibilitando el desarrollo la una estación meteorológica con menos costos monetarios y con un diseño modular que nos permitan adaptación y mejora del producto

Las estaciones meteorológicas de bajo costo han ganado popularidad en los últimos años gracias al avance de la tecnología y la reducción de costos en componentes electrónicos. Estas estaciones son dispo-

sitivos que permiten medir diversas variables climáticas, como temperatura, humedad, presión atmosférica, velocidad del viento y precipitación, entre otros.

Las estaciones meteorológicas de bajo costo son herramientas valiosas que ofrecen numerosas ventajas en términos de accesibilidad y democratización de información climática. Aunque presentan algunas desventajas, su importancia en sectores como la agricultura, la investigación, la gestión de desastres y la educación es innegable. Al permitir un monitoreo más amplio y accesible del clima, estas estaciones contribuyen al desarrollo sostenible y a una mejor comprensión del entorno natural. **(alggar2018)**

Ventajas de las estancaciones de bajo costo

Costo reducido: Su precio asequible permite que más personas, instituciones educativas y comunidades puedan adquirirlas.

Facilidad de instalación: Muchas estaciones de bajo costo son fáciles de instalar y no requieren conocimientos técnicos avanzados Permiten a comunidades locales acceder a datos meteorológicos relevantes, lo que puede ser crucial para la toma de decisiones informadas.

Los usuarios pueden adaptar las estaciones a sus necesidades específicas, eligiendo los sensores que desean incluir y modificando el software para personalizar la recolección y visualización de datos.

Muchas estaciones de bajo costo pueden conectarse a Internet, lo que permite la recopilación y transmisión de datos en tiempo real, facilitando el monitoreo continuo.

Desventajas de las estaciones de bajo costo.

Algunos modelos pueden no estar diseñados para resistir condiciones climáticas extremas o para un uso prolongado, lo que puede resultar en un mayor mantenimiento o reemplazo frecuente.

Pueden no ser tan precisas o fiables como las estaciones meteorológicas profesionales. La calidad de los sensores puede variar y afectar la exactitud de las mediciones.

No todas las estaciones de bajo costo ofrecen la misma variedad de sensores o la capacidad de medir variables más complejas (como calidad del aire o radiación solar).

Importancia de las estaciones de bajo costo en varios Sectores

Agricultura:

Los agricultores pueden utilizar estaciones meteorológicas de bajo costo para monitorear condiciones climáticas locales, optimizando el riego, la siembra y la cosecha. Esto puede mejorar los rendimientos y reducir el uso innecesario de recursos.

Investigación científica:

En el ámbito académico, estas estaciones permiten a investigadores recopilar datos para estudios sobre cambio climático, patrones meteorológicos y fenómenos ambientales sin incurrir en altos costos.

Gestión de desastres:

Las comunidades pueden instalar estaciones para monitorear condiciones climáticas extremas (como tormentas o sequías), lo que les ayuda a prepararse mejor y responder a emergencias.

Ciudades inteligentes:

En el contexto de las ciudades inteligentes, estas estaciones pueden integrarse en sistemas más amplios

para monitorear y gestionar el clima urbano, contribuyendo a la sostenibilidad y planificación urbana.

Salud pública:

Con el monitoreo del clima y condiciones ambientales, se pueden anticipar brotes de enfermedades transmitidas por vectores (como el dengue) o evaluar el impacto del clima en la salud pública.

En resumen, Las estaciones meteorológicas de bajo costo se desarrollaron para hacer la tecnología meteorológica más accesible a un público más amplio. Estas estaciones suelen ser más simples que las EMA y ofrecen una funcionalidad básica, pero a un precio más asequible.

Esta accesibilidad permite a personas, organizaciones y comunidades que no tienen acceso a las estaciones tradicionales obtener información meteorológica básica. Esto es útil para aplicaciones como la agricultura, la gestión de recursos hídricos y la educación ambiental.

Ademas, Las estaciones meteorológicas automatizadas: Se utilizan para monitorear el clima en aeropuertos, estaciones de esquí, granjas, ciudades, etc. Mientras que la estaciones meteorológicas de bajo costo: Pueden ser utilizadas por aficionados a la meteorología, escuelas, agricultores, o incluso para la investigación ciudadana sobre el clima.

1.3. Sensores Meteorológicos de bajo costo

Las estaciones meteorológicas de bajo costo utilizan una variedad de sensores para medir diferentes variables climáticas.Los sensores son componentes esenciales en las estaciones meteorológicas de bajo costo, ya que permiten medir diversas variables climáticas que son fundamentales para el monitoreo del tiempo y el clima. **(envira2021)**

Sensor de Temperatura y Humedad relativa (HTU21D: temperature, humidity).

Función: Mide la temperatura del aire en grados Celsius (°C) o Fahrenheit (°F) y mide la humedad relativa del aire, que es el porcentaje de vapor de agua presente en comparación con la cantidad máxima que el aire puede contener a esa temperatura.

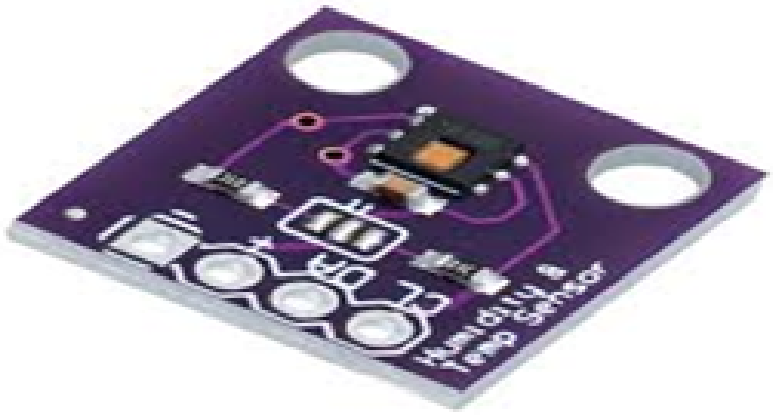


Figura 1.2. Sensor de Temperatura y Humedad relativa.

Sensor de Presión Atmosférica y Temperatura (MS5637: temperature, pressure).

Función: Mide la presión atmosférica en milibares (hPa) o pulgadas de mercurio (inHg). Es útil para prever cambios en el clima y Función: Mide la temperatura del aire en grados Celsius (°C) o Fahrenheit (°F).

Figura 1.3. Sensor de Temperatura y Presión Atmosférica.

Sensor de Radiación Solar (ML8511:UV)

Función: Mide la cantidad de radiación solar que llega a la superficie terrestre, expresada en vatios por metro cuadrado.

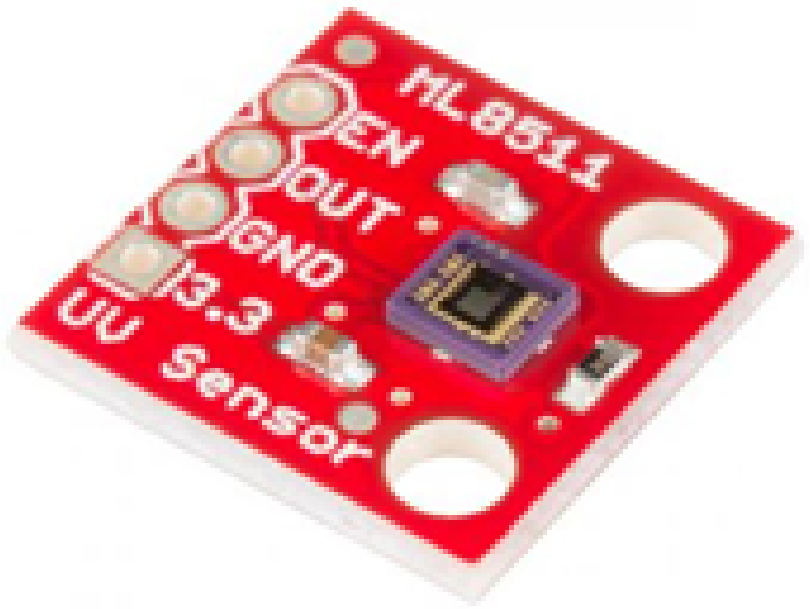


Figura 1.4. Sensor de radiación solar.

Sensor de Humedad relativa y Temperatura (MS8607: temperature, pressure, humidity).

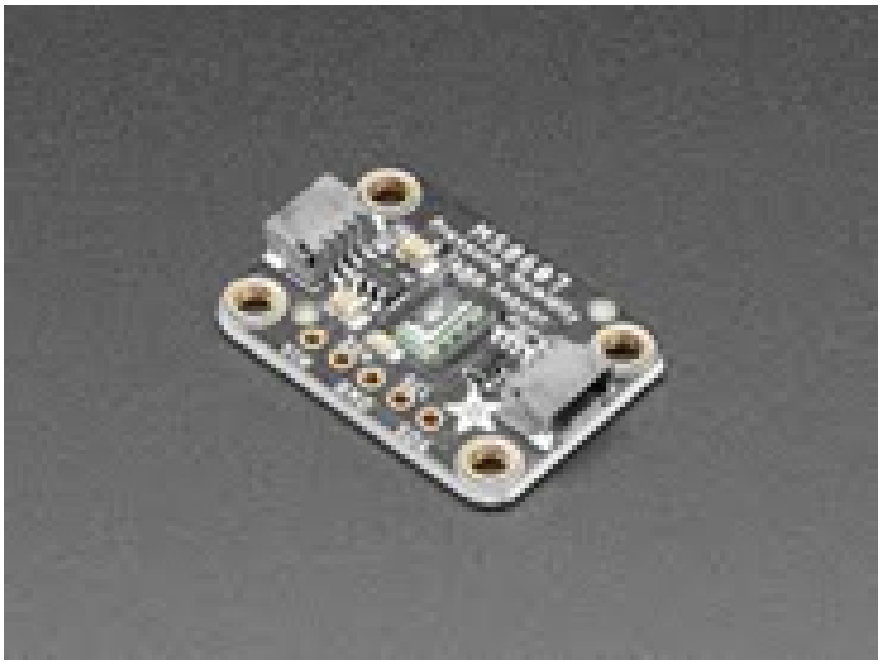


Figura 1.5. Sensor de Temperatura y Humedad relativa.

Sensor de temperatura (TSYS01: temperature).

Figura 1.6. Sensor de Temperatura.

Sensor de Temperatura y Temperatura de un objeto. (TSD305: temperature, object temperature).

Figura 1.7. Sensor de Temperatura y Temperatura.

WH-SP-RG: pluviómetro.

El pluviómetro es el instrumento más sencillo y más comúnmente empleado para medir la cantidad de lluvia.

Figura 1.8. Pluviómetro.

JL-FS2: Anemometer. (Anemómetro, velocidad del viento).

Un anemómetro mide la velocidad del viento. En interiores, un anemómetro mide la velocidad y el caudal del aire.

Figura 1.9. Anemómetro.

1.4. Plataforma arduino

1.4.1. Arduino

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso. (Fernandez, 2022).**(rosales2023)**

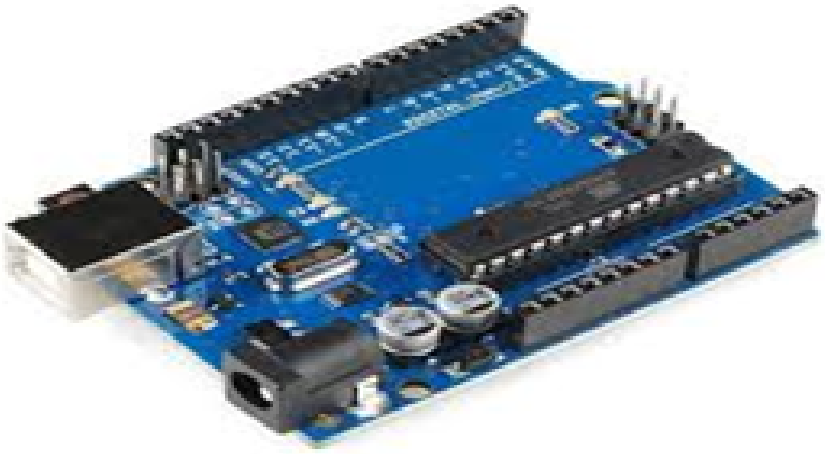


Figura 1.10. Placa Arduino.

1.4.2. DAQT(Tarjetas de adquisición de datos)

DAQT es un dispositivo que se utiliza para recopilar, procesar y almacenar datos relacionados con diferentes parámetros meteorológicos. Estas tarjetas son componentes esenciales en estaciones meteorológicas automáticas y sistemas de monitoreo ambiental. La tarjeta DAQ incluye conectores y circuitos para conectar los sensores, permitiendo la conversión de señales analógicas (como voltajes) en datos digitales que pueden ser procesados por una computadora o un microcontrolador. Una vez que los datos son adquiridos, la tarjeta puede realizar ciertas operaciones de procesamiento, como filtrado, promediado o calibración de las lecturas. Los datos recopilados pueden ser almacenados en memoria interna, o transmitidos a un sistema externo para su almacenamiento y análisis posterior. Esto puede incluir el uso de bases de datos o plataformas en la nube.**(jmi2022)**

1.5. Medición y transmisión de datos en las DAQT

En el proceso de recolección de datos los sensores recolectan datos en intervalos regulares y los envían aun servidor central utilizando un protocolo de cuminicación. Este proceso puede incluir: Adquisición de datos, transmisión de datos y almacenamiento.

1.5.1. Flujo de datos

1. Los sensores miden las variables meteorológicas como temperatura, humedad, presión atmosférica, velocidad del viento, etc.

2. Los datos se transmiten al microcontrolador (Arduino).
3. El microcontrolador envía los datos al servidor central utilizando un protocolo de comunicación.
4. Los datos se almacenan en una base de datos en la nube de forma temporal.
5. La información es enviada a la aplicación web para almacenar los datos como defina el usuario.
6. Los datos son mostrados al usuario a través de gráficas y mapas con ubicaciones de los dispositivos.

1.6. Visualización de datos meteorológicos

La visualización de datos es la representación de datos mediante el uso de gráficos comunes, como gráficas, diagramas, infografías e incluso animaciones. Estas representaciones visuales de información comunican relaciones de datos complejas y conocimientos basados en datos de una manera que resulta fácil de comprender. Para Amazon Web Services la visualización de datos es el proceso de utilizar elementos visuales como gráficos o mapas para representar datos. De esta manera, se trasladan datos complejos, de alto volumen o numéricos a una representación visual más fácil de procesar. Las herramientas de visualización de datos mejoran y automatizan el proceso de comunicación visual para lograr precisión y detalle. La visualización de datos es crucial para interpretar y analizar la información meteorológica. Permite a los usuarios identificar patrones, tendencias y anomalías en los datos recogidos.

La visualización de variables meteorológicas mediante estaciones meteorológicas ofrecen una forma efectiva de monitorear y analizar el clima en tiempo real. A través del uso adecuado de tecnologías y herramientas, es posible construir sistemas robustos que no solo recopilen datos, sino que también los presenten de manera comprensible y útil para diferentes usuarios, desde investigadores hasta ciudadanos interesados en el clima local. A medida que avanza la tecnología IoT, las posibilidades para mejorar la precisión y accesibilidad de los datos meteorológicos continúan expandiéndose, brindando oportunidades emocionantes para el futuro del monitoreo climático.

1.7. Internet de las cosas (IoT)

El Internet de las cosas (IoT) es el proceso que permite conectar los elementos físicos cotidianos al Internet: desde los objetos domésticos comunes, como las bombillas de luz, hasta los recursos para la atención de la salud, como los dispositivos médicos; las prendas y los accesorios personales inteligen-tes; e incluso los sistemas de las ciudades inteligentes.

Los dispositivos del IoT que se encuentran dentro de esos objetos físicos suelen pertenecer a una de estas dos categorías: son interruptores (es decir, envían las instrucciones a un objeto) o son en sensores (recopilan los datos y los envían a otro lugar).

1.7.1. Aplicación de IoT

Una aplicación de IoT es un conjunto de servicios y de machine learning o inteligencia artificial (IA) para analizar servicios y software que integra los datos recibidos de varios dispositivos de IoT. Estas decisiones se comunican al dispositivo de IoT y este responde de forma inteligente a las entradas.

1.7.2. Utilización de sensores de IoT para recopilar datos en tiempo real

La utilización de sensores IoT para recopilar datos en tiempo real ha revolucionado la forma en que obtenemos información sobre el clima. Gracias a esta tecnología, los meteorólogos y científicos pueden acceder a datos más precisos y actualizados para realizar predicciones meteorológicas más confia-bles. **(bareno2011)**

Estos sensores, conectados a través de Internet, recolectan información sobre diferentes variables climáticas, como la temperatura, la humedad, la presión atmosférica y la velocidad del viento. Estos datos se recopilan de manera continua y se envían a una plataforma centralizada, donde son procesados y analizados.**(aws2021)**

La conexión de estos sensores a través de la IoT permite obtener información en tiempo real, lo que significa que los datos se actualizan de forma constante y se pueden acceder en cualquier momento y desde cualquier lugar. Esto es especialmente importante en la predicción meteorológica, ya que las condiciones climáticas pueden cambiar rápidamente y es fundamental contar con datos actualizados para realizar pronósticos precisos.**(aws2021)**

Además, la utilización de sensores IoT no solo mejora la precisión de las predicciones, sino que también permite obtener información en áreas remotas o de difícil acceso. Los sensores pueden ser instalados en lugares como montañas, océanos o zonas desérticas, donde la recolección de datos puede ser complicada o costosa.

1.7.3. Beneficios del análisis de datos meteorológicos históricos

El análisis de datos meteorológicos históricos ofrece varios beneficios para mejorar las predicciones meteorológicas:

- Mayor precisión en los pronósticos: Al analizar los datos del pasado, es posible identificar patrones climáticos y tendencias que pueden ayudar a predecir de manera más precisa el clima futuro.
- Detección de eventos extremos: El análisis de datos históricos puede ayudar a identificar eventos climáticos extremos, como tormentas o sequías, y predecir su aparición en el futuro.
- Mejor planificación: Con predicciones más precisas, las personas y las organizaciones pueden planificar mejor actividades al aire libre, como eventos deportivos o agricultura.
- Optimización de recursos: Al conocer con mayor precisión el clima futuro, es posible optimizar el uso de recursos como el agua o la energía.

El análisis de datos meteorológicos históricos es una herramienta poderosa para mejorar las prediccio-nes

meteorológicas y aprovechar al máximo el Internet de las Cosas. Gracias a esta conexión, ahora podemos obtener pronósticos más precisos y tomar decisiones informadas en base al clima.

1.7.4. Beneficios de utilizar la conexión IoT en la predicción meteorológica

- Mayor precisión: al contar con datos en tiempo real de múltiples puntos de medición, las predicciones son más precisas y confiables.
- Actualizaciones en tiempo real: la conexión IoT permite recibir actualizaciones constantes sobre las condiciones atmosféricas, lo que permite tomar decisiones rápidas en caso de cambios repentinos.
- Reducción de costos: al utilizar sensores conectados a través de IoT, se reduce la necesidad de estaciones meteorológicas costosas y su mantenimiento.
- Mejora en la planificación: al contar con predicciones precisas, se puede planificar de manera más eficiente actividades que dependen del clima, como siembras, vuelos o eventos al aire libre.

La combinación de la conexión IoT y la predicción meteorológica permite obtener información precisa y actualizada sobre las condiciones atmosféricas, lo que resulta en una mayor eficiencia y seguridad en diversas industrias y actividades que dependen del clima.**(aws2021)**

1.7.5. Importancia de compartir información meteorológica con otros dispositivos y aplicaciones utilizando IOT

En el mundo actual, donde la tecnología está cada vez más presente en nuestras vidas, es importante aprovecharla para mejorar diversos aspectos, como por ejemplo, las predicciones meteorológicas. Gracias a la conexión a Internet de las cosas (IoT), ahora es posible compartir información meteoroló-gica en tiempo real con otros dispositivos y aplicaciones, lo que nos permite tener acceso a prediccio-nes mucho más precisas y actualizadas.

La conexión entre dispositivos y aplicaciones meteorológicas a través de IoT permite que los datos recopilados por estaciones meteorológicas o sensores especializados sean transmitidos y compartidos de manera rápida y eficiente. Esto significa que no solo los meteorólogos profesionales tienen acceso a esta información, sino que cualquier persona puede acceder a ella desde su teléfono inteligente, computadora u otros dispositivos conectados a Internet.**(aws2021)**).

La disponibilidad de datos meteorológicos precisos y actualizados tiene numerosas ventajas. Por un lado, nos permite tomar decisiones informadas en diferentes aspectos de nuestra vida diaria, como planificar actividades al aire libre, vestirnos adecuadamente según el clima o incluso tomar medidas para proteger nuestra salud en caso de condiciones climáticas adversas.

Además, la conexión de dispositivos y aplicaciones meteorológicas a través de IoT también puede tener un impacto positivo en diferentes sectores, como la agricultura, la aviación, la navegación marítima o la gestión de desastres naturales. Gracias a la información meteorológica precisa y en tiempo real, es posible optimizar procesos y tomar decisiones más acertadas en estas áreas.**(aws2021)**

Es importante destacar que la precisión de las predicciones meteorológicas depende en gran medida de la calidad de los datos recopilados. Por eso, es fundamental contar con estaciones meteorológicas o sensores confiables y bien calibrados, así como también tener en cuenta otros factores que puedan influir en la precisión de las predicciones, como la ubicación geográfica o la altitud.

1.7.6. Ventajas de la conexión entre dispositivos y aplicaciones meteorológicas

- Predicciones más precisas: Gracias a la conexión entre dispositivos y aplicaciones meteorológi-cas, podemos acceder a predicciones más precisas y actualizadas, lo que nos permite planificar nuestras actividades de manera más efectiva.
- Accesibilidad: La posibilidad de acceder a información meteorológica desde cualquier dispositivo conectado a Internet nos brinda una mayor accesibilidad y nos permite tomar decisiones informadas en tiempo real.
- Optimización de procesos: En diferentes sectores, como la agricultura o la navegación marítima, la conexión entre dispositivos y aplicaciones meteorológicas a través de IoT permite optimizar procesos y tomar decisiones más acertadas, lo que puede tener un impacto positivo en la productividad y eficiencia.
- Seguridad y protección: La conexión entre dispositivos y aplicaciones meteorológicas también puede ser de gran utilidad en la gestión de desastres naturales, permitiendo tomar medidas de protección y seguridad basadas en información meteorológica precisa y en tiempo real.

La conexión entre dispositivos y aplicaciones meteorológicas a través de IoT nos brinda la posibilidad de acceder a predicciones meteorológicas más precisas y actualizadas, así como también nos permite tomar decisiones informadas en diferentes aspectos de nuestra vida diaria. Además, esta conexión puede tener un impacto positivo en diversos sectores, optimizando procesos y mejorando la seguridad y protección en casos de desastres naturales.(Rodríguez,2022).

1.7.7. Importancia de proporcionar pronósticos personalizados basados en la ubicación y preferencias del usuario

En la era de la Internet de las cosas (IoT), la conexión de dispositivos y sensores en tiempo real ha revolucionado la forma en que accedemos a la información. Y uno de los campos donde esta conexión se ha vuelto especialmente relevante es en las predicciones meteorológicas.

Gracias a la combinación de sensores climáticos y la recopilación de datos en tiempo real, es posible proporcionar pronósticos mucho más precisos y personalizados. Ya no dependemos únicamente de las predicciones generales basadas en grandes regiones, ahora podemos obtener información específica basada en la ubicación y preferencias de cada usuario.**(aws2021)**

1.8. Protocolo MQTT (Message Queuing Telemetry Transport)

MQTT es un protocolo de mensajería publish-subscribe basado en el modelo cliente-servidor, desarrollado por IBM en la década de 1990. Es conocido por su bajo consumo de ancho de banda, baja sobre-carga y su capacidad para funcionar eficientemente en redes con ancho de banda limitado.**(oasis2023)**

1.8.1. Funcionamiento de MQTT

En MQTT, los dispositivos pueden publicar mensajes en un "tema.^{es}pecífico y suscribirse a temas para recibir mensajes. Los mensajes se envían a través de un "broker"MQTT, que actúa como interme-diario entre los dispositivos. Los dispositivos pueden ser tanto clientes que publican mensajes como clientes que reciben mensajes, y pueden cambiar entre estos roles dinámicamente. **(oasis2023)**

1.8.2. Características principales de MQTT

- Ligero: MQTT está diseñado para ser ligero y eficiente, lo que lo hace adecuado para dispositivos con recursos limitados.
- Eficiente en el uso de la red: Debido a su baja sobrecarga y su capacidad para minimizar el uso de ancho de banda, MQTT es ideal para redes con restricciones de ancho de banda.
- Tolerancia a la pérdida de conexión: MQTT es capaz de manejar conexiones intermitentes, lo que lo hace adecuado para dispositivos que pueden estar desconectados periódicamente.
- Escalabilidad: MQTT es altamente escalable, lo que significa que puede manejar un gran número de dispositivos conectados simultáneamente.
- Seguridad: MQTT puede implementar medidas de seguridad como autenticación, autorización y cifrado para proteger la comunicación entre dispositivos y brokers.

1.8.3. Aplicaciones de MQTT

Internet de las cosas (IoT): MQTT es ampliamente utilizado en aplicaciones de IoT debido a su eficiencia y escalabilidad. Permite la comunicación entre dispositivos IoT y la transferencia de datos de forma eficiente. Sistemas de monitorización y control: MQTT se utiliza en sistemas de monitorización y control en tiempo real, como sistemas de telemetría, sistemas de gestión de edificios inteligentes y sistemas de gestión de energía.

Aplicaciones móviles y web: MQTT también se utiliza en aplicaciones móviles y web para la transmisión de datos en tiempo real, como aplicaciones de mensajería instantánea y aplicaciones de redes sociales. **(oasis2023)**

1.8.4. Implementaciones de MQTT

Existen varias implementaciones de MQTT disponibles, tanto de código abierto como comerciales. Algunas de las implementaciones más populares incluyen Eclipse Mosquitto, HiveMQ, IBM MessageSight y EMQ X. **(oasis2023)**

Implementacion de MQTT en las DAQT

La implementación de MQTT (Message Queuing Telemetry Transport) en sistemas de adquisición de datos (DAQT) es una excelente manera de facilitar la comunicación entre dispositivos IoT y servidores. MQTT es un protocolo ligero y que utiliza un modelo de publicación-suscripción, lo que permite que los dispositivos publiquen datos en un tema específico y que solo aquellos dispositivos que estén suscritos a ese tema reciban esos datos.

Pasos para implementar MQTT en DAQT

- Configurar el Broker MQTT: Selecciona y configura un broker MQTT (como Mosquitto) que actuará como intermediario entre los dispositivos y el servidor.
- Desarrollar el Cliente MQTT: Implementa el código necesario en los dispositivos de adquisición de datos para que puedan publicar datos en el broker MQTT.
- Configurar el Servidor: Configura el servidor para que suscriba a los temas relevantes y procese los datos recibidos.
- Seguridad: Asegúrate de implementar medidas de seguridad, como el uso de certificados y cifrado de datos.

1.9. Metodología de desarrollo

La metodología de desarrollo es un marco de trabajo que parte de una posición teórica y conlleva a una selección de técnicas concretas o métodos acerca del procedimiento para el cumplimiento de los objetivos. Es el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y que hacer para realizarlos y finalizar una tarea.

Estas se dividen en dos enfoques o ramas, las cuales son conocidas como metodologías ágiles y metodologías tradicionales. Las tradicionales centran su atención en llevar una documentación exhaustiva de todo el proyecto, la planificación y control del mismo, en especificaciones precisas de requisitos y modelado y en cumplir con un plan de trabajo, definido todo esto en la fase inicial del desarrollo del proyecto. Por otro lado, las ágiles son convenientes para guiar proyectos de escaso volumen comercial que demanden una rápida implementación **(pressman2015)**Tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo, haciendo énfasis en la calidad y menor tiempo de construcción del software.

Tabla 1.1. Comparación de Metodologías(Fuente: Elaboración propia).

Metodologías ágiles	Metodologías tradicionales
Se basan en heurísticas provenientes de prácticas de producción de código.	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo.
Preparados para cambios durante el proyecto	Cierta resistencia a los cambios.
Impuestas internamente por el equipo	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso muy controlado, numerosas normas.
Contrato flexible e incluso inexistente	Contrato prefijado.
El cliente es parte del desarrollo.	Cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10)	Grupos grandes.
Pocos artefactos	Más artefactos.

Teniendo en cuenta la comparación antes expuesta se decide seleccionar un enfoque ágil, debido a que, el equipo está compuesto por un desarrollador y el cliente forma parte del equipo de desarrollo. Además, se cuenta con la experiencia pertinente para el desarrollo de aplicaciones sobre la infraestructura tecnológica necesaria y se conoce claramente el dominio del problema a resolver.

Se realiza una comparación de diferentes metodologías ágiles para seleccionar la más adecuada a las necesidades planteadas en el trabajo.

1.9.1. Metodologías ágiles

Es aquella que posibilita adaptar la forma de trabajo a las condiciones que presenta el proyecto, para conseguir flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno **(pressman2015)**

Algunas de estas metodologías son:

- Programación extrema (XP), es de las más exitosas y se considera también emergente
- Mobile-D (ágil y extrema para móviles)
- Scrum
- Crystal
- Evolutionary Project Management (Evo)
- Lean Development
- AUP

Acontinuacion se presenta una tabla de comparación de tecnologías ágiles y tecnologías tradicionales.

Metodología	Características
Programación extrema (extreme programming, XP)	Metodología basada en prueba y error. Cliente bien definido. Los requisitos pueden y van a cambiar. Grupo pequeño y muy integrado (máximo 12 personas). Equipo con formación elevada y capacidad de aprender. Fundamentada en valores. Expresada en 12 prácticas. Implementa compatibilidad y usabilidad con otras metodologías. Está orientada hacia quien produce y usa el software. Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema. Programación organizada. Reduce la taza de errores
Crystal Methodologies	Entregas frecuentes, en base a un ciclo de vida iterativo e incremental. Cuantas más personas estén implicadas, más grande debe ser la metodología. Entorno técnico con pruebas automatizadas, gestión de la configuración e integración continua, Si el proyecto tiene mucha densidad, un error no detectado puede ser crítico. El aumento de tamaño o densidad añade un costo considerable al proyecto. La forma más eficaz de comunicación es la interactiva (cara a cara). Tamaño de un equipo (número de componentes). Equipo: • Claro es para equipos de hasta 8 personas o menos. • Amarillo para equipos entre 10 a 20 personas. • Naranja para equipos entre 20 a 50 personas. • Roja para equipos entre 50 a 100 personas. • Azul para equipos entre 100 a 200 personas. Comunicación entre los componentes. Distintas políticas a seguir. Mejora reflexiva.
Scrum	Aplicación de las metodologías ágiles en los procesos de calidad, entregando resultados óptimos. Su adopción en prácticas orientadas a planes como CMMI (Integración de modelos de madurez de capacidades o Capability Maturity Model Integration es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software) o PSP/TSP (Personal Software Process o Proceso Personal de Software, es un conjunto de prácticas disciplinadas para la gestión del tiempo y mejora de la productividad personal de los programadores o ingenieros de software, en tareas de desarrollo y mantenimiento de sistemas, mediante el seguimiento del desempeño predicho frente al desempeño real) / (Team Software Process proporciona un marco de trabajo de procesos definidos que está diseñado para ayudarle a equipos de gerentes e ingenieros a organizar y producir proyectos de software de gran escala).La productividad y competitividad de las empresas de software al implementar este tipo de metodología. El aporte a la industria a través de las buenas prácticas. El aporte a la gestión de proyectos a través del proceso. Aporte efectivo dentro de la gerencia de proyectos de software. La competitividad en el desarrollo de productos y proyectos industriales a través de la implementación de esta metodología ágil.

Figura 1.11. Comparación de metodologías ágiles (Fuente: Elaboración propia).

1.9.2. Metodología de desarrollo de software Programación Extrema

La metodología escogida es programación extrema (XP) es una metodología ágil de gestión de proyectos que se centra en la velocidad y la simplicidad con ciclos de desarrollo cortos. Los objetivos de XP son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, debemos responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación. El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software **(solis2003)**

Fundamentos de la metodología seleccionada

En la presente investigación se decide emplear Programación Extrema (XP) como metodología de desarrollo de software, pues se adecua a los requerimientos exigidos, dado que se trata de un proyecto de corta duración, su equipo de desarrollo es pequeño y el cliente forma parte del mismo, lo que contri-buye a fomentar las relaciones interpersonales y el buen clima de trabajo.

Esta es una metodología que consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. Este tipo de programación es la adecuada para los proyectos con requisitos imprecisos, muy cambiantes y con un riesgo técnico excesivo.

Por otro lado, tiene un desarrollo iterativo e incremental. Se ha tenido en cuenta que XP define la constante comunicación y la retroalimentación, teniendo como uno de sus fines fundamentales la construcción de un producto que vaya en línea con los requerimientos del cliente y las necesidades del proyecto. La aplicación será desarrollada por un programador, no estrictamente coincidiendo con la característica de programación en parejas que propone la misma, pero basado en el trabajo entre el cliente y el pro-gramador.

1.10. Herramientas y Tecnologías en el desarrollo

Lenguajes de programación utilizado en el desarrollo de la aplicación web

Los lenguajes de programación son un conjunto de reglas y sintaxis utilizados para escribir programas informáticos, estos lenguajes le permiten al programador interactuar con la computadora y darle instrucciones para que realicen algunas tareas específicas. Es un lenguaje de programación moderno creado por Guido van Rossum a inicios de los años noventa, esta bajo la licencia de software libre y se puede descargar del sitio web **(aguirre2022)**

Tabla 1.2. Comparación de lenguajes de programación (Fuente: Elaboración propia).

Lenguajes de programación	Características
Python	- Simplicidad: Python se destaca por su sintaxis clara y legible, lo que facilita la escritura y comprensión de código. - Versatilidad: Es un lenguaje utilizado en una varriedad de aplicaciones, desde desarrollo web y análisis de datos hasta inteligencia artificial y aprendizaje automático. - Comunidad activa: Python cuenta con una amplia comunidad de desarrolladores que comparten paquetes de código abierto, lo que facilita la cola-boración y el aprendizaje.
Java	- Portabilidad: Java es conocido por ser un lenguaje portátil, lo que significa que el código Java puede ejecutarse en diferentes plataformas sin necesidad de modificaciones. - Orientado a objetos: Java es un lenguaje orientado a objetos, lo que lo hace adecuado para aplicaciones grandes y complejas que requieren estructuras de datos avanzadas. - Desempeño: Java es conocido por su rendimiento y velocidad, lo que lo hace ideal para aplicacio-nes que requieren ejecución rápida y eficiente.

PHP	- Código Abierto: PHP es un lenguaje de código abierto, lo que significa que es gratuito y tiene una comunidad activa que contribuye a su desarrollo y mejora. - Amplia Compatibilidad con Bases de Datos: PHP ofrece soporte nativo para varias bases de datos, siendo MySQL la más popular. También es com-patible con PostgreSQL, SQLite, Oracle, entre otras. - Orientación a Objetos: Aunque PHP comenzó como un lenguaje procedural, ha evolucionado para incluir características de programación orientada a objetos, lo que permite un diseño más modular y reutilizable.

1.10.1. Python V3.8.10

Python es un lenguaje de programación de propósito general tipado dinámicamente, que tiene menos códigos, fácil de usar y más sencillo de aprender.

Se decide escoger Python como lenguaje de programación para el desarrollo del sistema debido a que:

- Simplicidad y Legibilidad: Python tiene una sintaxis clara y concisa que permite a los desarrolladores escribir código que es fácil de leer y entender. Esto es especialmente beneficioso para los principiantes y para proyectos donde la colaboración es clave.
- Versatilidad: Python se utiliza en una amplia variedad de campos, incluyendo desarrollo web, ciencia de datos, inteligencia artificial, automatización de tareas y más. Esta versatilidad lo convierte en una opción ideal para desarrolladores que desean trabajar en diferentes áreas.
- Gran Ecosistema de Bibliotecas: La disponibilidad de numerosas bibliotecas y frameworks facilita el desarrollo rápido de aplicaciones complejas. Por ejemplo, bibliotecas como NumPy y Pandas son esenciales para la ciencia de da-tos, mientras que Django y Flask son populares para el desarrollo web.
- Comunidad Activa: Python cuenta con una comunidad grande y activa que proporciona soporte, documentación y re-cursos educativos. Esto facilita la resolución de problemas y el aprendizaje continuo.
- Desarrollo Rápido: La capacidad de Python para permitir un desarrollo rápido lo hace ideal para startups y proyectos donde el tiempo es un factor crítico.

1.10.2. Lenguaje de modelado UML 2.1

UML (Unified Modeling Language) fue adoptado como estándar del Object Management Group (Grupo Gestor de Objetos) en 1997 debido a que representa una colección de las mejores prácticas de ingeniería que han sido probadas con éxito en el modelado de sistemas. Es un lenguaje para la especificación, visualización, construcción y documentación de sistemas, no solo de software. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, pues ha sido diseñado para modelar cualquier tipo de soluciones informáticas, arquitectura o cualquier otra rama **(larman2016)**

1.10.3. Visual Paradigm for UML 17.0

Visual Paradigm for UML es una herramienta CASE3 que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Por las razones antes expuestas se emplea esta herramienta de modelado para la propuesta de solución en su versión 8 .0. **(larman2016)**)

1.10.4. Visual Studio Code 1.82

Visual Studio Code es un IDE de desarrollo muy maduro que ha existido durante mucho tiempo. Es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity). (Herrera, 2019). En primer lugar, es un editor que se quita de en medio el ciclo de edición-construcción-depuración deliciosamente fluido significa menos tiempo jugando con su entorno y más tiempo ejecutando sus ideas. **(larman2016)**

Funciones de Visual Studio:

- Desarrolle: navegue, escriba y corrija su código rápidamente.
- Depurar: depurar, perfilar y diagnosticar con facilidad.
- Prueba: escriba código de alta calidad con herramientas de prueba completas.
- Colaborar: utilice herramientas de control de versiones como Perforce, Git. Sea ágil y colabore de manera eficiente.
- Ampliar: elija entre miles de extensiones para personalizar su IDE.
- Soporte: soporte empresarial y una gran comunidad.
- Windows: desarrolle aplicaciones y juegos para cualquier dispositivo Windows.
- Aplicaciones móviles: cree aplicaciones nativas o híbridas para Android, iOS y Windows.
- Aplicaciones de Azure: cree, administre e implemente aplicaciones de escala de nube en Azure.

- Aplicaciones web: desarrolle aplicaciones web modernas y aplicaciones web progresivas (PWA).
- Office: utilice potentes herramientas para el desarrollo de Office.
- Juegos: diseñe, codifique y depure juegos con gráficos de vanguardia.
- Extensiones: escriba sus propias extensiones de Visual Studio.
- Base de datos: desarrolle e implemente bases de datos SQL Server y Azure SQL.

1.10.5. Marco de trabajo(Django V 4.2.5)

Django es el mejor frameworks para la implementación de desarrollo de sistemas web . Este sistema cumple con toda las indicaciones del modelo de evaluación .Es de código abierto escrito en Python que permite construir aplicaciones web mas rápido y con menos códigos . Contiene un conjunto de componentes que permite desarrollar sitios web de manera mas fácil y rápida.

1.10.6. HTML5

Lenguaje de publicación especificado como un estándar por el W3C (World Wide Web Consortium) que permite la creación de páginas web. Inicialmente fue presentado por Tim Berners-Lee que propuso un sistema basado en hipertexto para el intercambio de información en la Web. La aparición del lenguaje influyó notablemente en el crecimiento de Internet, dónde la información era distribuida mediante colecciones fragmentadas de textos, imágenes y sonidos. HTML es independiente de la plataforma utilizada y se basa fundamentalmente en el uso de etiquetas estructurales y semánticas, adecuadas para la creación de documentos relativamente simples que permiten simplificar su estructura cite.

1.10.7. CSS 3

CSS es un lenguaje que nos permite otorgar atributos a los elementos de los documentos realizados en HTML , CSS permite realizar una separación del diseño (formato y estilo) de los contenidos de la pagina web .CSS ofrece características que también se pueden realizar en HTML .

1.10.8. Bootstrap CSS

Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adapta-bles a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías. **(aguirre2022)**

1.11. Tipos de sistemas gestores de bases de datos

Como elemento importante de esta investigación, se realizará un estudio y caracterización de algunos sistemas gestores de bases de datos para posteriormente seleccionar el más indicado a utilizar en la propuesta

de solución, dentro de los que se encuentran los siguientes.

MySQL: es un sistema de gestión de base de datos, multihilo y multiusuario seguramente el más usado en aplicaciones creadas como software libre. Por un lado, se ofrece bajo la GNU GPL2 , pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso. Entre las ventajas que ofrecen se encuentra:

- Velocidad al realizar las operaciones.
- Bajo costo en requerimientos para la elaboración de bases de datos.
- Facilidad de configuración e instalación.

PostgreSQL: es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y apoyada por organizaciones comerciales. La co-munidad PostgreSQL se denominada el PGDG (PostgreSQL Global Development Group). Sus principales características son:

- Alta concurrencia: mediante un sistema denominado MVCC (Acceso concurrente multi versión, por sus siglas en inglés).
- Amplia variedad de tipos nativos: provee nativamente varios soportes.
- Ahorros considerables de costos de operación.
- Estabilidad y confiabilidad.

1.11.1. Selección del sistema gestor de base de datos

Después de haber analizado las características de los diferentes sistemas gestores de bases de datos debido a que es necesario almacenar la información de todas las personas autorizadas en las diferentes áreas o locales en el sistema que se va a desarrollar, se decidió usar en PostgreSQL en su versión 9.6.8 pues tiene un alto rendimiento probado, fiabilidad, alta velocidad para consultas y facilidad de uso con el servidor apache.

PgAdmin 4

PmingAd es una herramienta indispensable para gestionar y administrar PostgreSQL, la base de datos de código abierto más avanzada del mundo. Por lo tanto, pgAdmin es la herramienta para gestionar nuestras bases de datos espaciales PostGIS.El software tiene la apariencia de una aplicación de escritorio sea cual sea el entorno de tiempo de ejecución (escritorio o web), y mejora enormemente respecto a pgAdmin III con elementos de interfaz de usuario actualizados, opciones de despliegue multiusuario / web, paneles y un diseño más moderno .

1.12. Broker de mensajería

Un Broker de mensajería es un componente esencial en los sistemas de comunicación distribuida y en la Internet de las Cosas (IoT), que facilita la transmisión de mensajes entre distintos sistemas, dispositivos y aplicaciones. Su función principal es gestionar la comunicación de datos de manera eficiente, segura y escalable. Actúa como intermediario entre los productores de los mensajes (quienes los crean) y los consumidores (quienes los leen y procesan), traduciéndolos al lenguaje formal establecido por cada uno de los mismos. De esta forma, los sistemas comunicantes no necesitan conocerse el uno al otro, ni siquiera estar escritos en el mismo lenguaje o emplear las mismas técnicas, lo que proporciona un gran desacoplamiento . **(microsoft2023)**

Mosquitto 2.0.18: es la última versión estable del broker de mensajes de código abierto Mosquitto que implementa el protocolo MQTT (Message Queuing Telemetry Transport). El protocolo MQTT proporciona un método ligero para enviar mensajes utilizando un modelo de publicación/suscripción. Esto lo hace adecuado para mensajería de Internet de las cosas, como sensores de baja potencia o dispositivos móviles como teléfonos, computadoras integradas o microcontroladores. **(oasis2023)**

1.13. Valoración de los sistemas homólogos

Aunque existen estaciones meteorológicas con el rótulo de automático y autónomo, no quiere decir que realicen todo un proceso de pronóstico del tiempo. Estos sistemas solo se ocupan de facilitar el proceso de medición de las variables climáticas, para que los profesionales se encarguen de la interpretación de la información.

1. Weather Underground (WU): Es una red meteorológica comunitaria que utiliza estaciones meteorológicas personales (PWS) para proporcionar datos meteorológicos hiperlocales.Permite a los usuarios ver datos meteorológicos en tiempo real desde múltiples estaciones meteorológicas.Posee una interfaz web y móvil intuitiva para mostrar gráficos y reportes detallados.Permite la integración de datos de estaciones meteorológicas personales, similar a la integración de DAQT.
2. WeatherFlow: Ofrece soluciones meteorológicas profesionales y personales, incluyendo estaciones meteorológicas inteligentes con conectividad IoT.Utiliza plataformas IoT para la recolección y transmisión de datos meteorológicos. Ofrece datos meteorológicos en tiempo real y la capacidad de generar reportes detallados.Compatible con diversos sensores y hardware para la adquisición de datos.
3. Ambient tworkWeather: Una red de estaciones meteorológicas personales que proporciona datos meteorológicos en tiempo real y permite la visualización y el análisis de dichos datos.Dispone de una aplicación web para visualizar datos en tiempo real, gráficos y reportes.

Tabla 1.3. Estudio de soluciones homólogas.

Característica	Weather Underground	WeatherFlow	Ambient Weather Network	Propuesta de Solución
Visualización en Tiempo Real	Si	Si	Si	Si
Interfaz de Usuario Amigable	Si	Si	Si	Si
Integración con IoT	Limitada	Limitada	Si	Si
Gestión de Dispositivos	Si	Si	Si	Si
Hardware Abierto	No	Si	Si	Si
Seguridad en la Comunicación	Sí (HTTPS)	Sí (HTTPS, MQTT con TLS)	Sí (HTTPS)	Sí (HTTPS, MQTT)
Disponibilidad en nuestro país	No	Si	Si	Si

Resumen del análisis de la soluciones existentes

A partir del análisis de los sistemas informáticos homólogos antes descrito se puede arribar a la conclusión que tienen ciertas similitudes con el sistema a desarrollar. Estas soluciones permiten la visualización de variables meteorológicas en tiempo real, gestionar dispositivos. Por lo que se desarrollará un nuevo sistema teniendo en cuenta estas propuestas identificadas. Cada uno de estos servicios tiene opciones gratuitas y de pago, así como similitudes en cuanto a la calidad de los datos meteorológicos y la disponibilidad de pronósticos detallados. Sin embargo, algunas desventajas potenciales incluyen la presencia de anuncios en las versiones gratuitas, posibles limitaciones en las funciones disponibles en las versiones gratuitas o con suscripción y ademas estos sistemas están cargados de información innecesarias para el usuario.

1.14. Conclusiones del capítulo

- En el desarrollo del capítulo se obtuvo una mejor visión acerca del problema planteado y se dieron cumplimiento a las primeras tareas de investigación llegando a las siguientes conclusiones:
- El análisis de los conceptos asociados a la solución permitió un acercamiento a los temas relacio-nados con el desarrollo de una estación de meteorología

- Se definió las metodologías, tecnologías y herramientas que se utilizaran en el desarrollo de la propuesta de solución, estableciendo la base teórica y metodológica de la aplicación a implementar.
- Se caracterizaron varios lenguajes de programación, que son fundamentales para la creación del sistema.
- El proyecto tiene las características de tener un tiempo de desarrollo corto y la necesidad de realizar entregas constantes en ciclos cortos, por lo que es necesario en uso de una metodología ágil XP.
- Se desarrolló el análisis de los sistemas homólogos a al sistema en cuestión.

CAPÍTULO 2

ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN

En este capítulo se abordan los elementos asociados a las fases de Planificación y Diseño en correspondencia con la metodología de desarrollo de software empleada. Se describe la propuesta de solución, a partir de las Historias de Usuario (HU) como artefacto para la descripción de las funcionalidades del sistema. Se muestran el plan de iteraciones, de entrega y los elementos relacionados con la arqui-tectura del sistema.

2.1. Propuesta de solución

La propuesta de solución para gestionar la visualización e intercambio de información entre los dispositivos o tarjetas de adquisición de datos (DAQT) meteorológicos en la Universidad de las Ciencias Informáticas (UCI):

Desarrollar una aplicación web que permita visualizar en tiempo real los datos meteorológicos recopilados por los DAQT.

La aplicación web debe tener las siguientes funcionalidades:

- Interfaz de usuario intuitiva y amigable para mostrar los datos meteorológicos.
- Capacidad para recibir y procesar los datos provenientes de las DAQT.
- Opción de configurar y gestionar los dispositivos (agregar, modificar, eliminar).
- Emplear una placa Arduino como DAQT para la recolección de datos meteorológicos.
- Desarrollar el firmware necesario para que el DAQT pueda comunicarse de manera segura y eficiente con la aplicación web.

Esta propuesta de solución permite gestionar de manera integral la visualización e intercambio de información entre los DAQT meteorológicos en la UCI, aprovechando las tecnologías de hardware abierto y el desarrollo de una aplicación web intuitiva y funcional. De esta forma, se podrá obtener datos meteorológicos precisos y en tiempo real enviados desde las DAQT hacia el broker utilizando el protocolo MQTT y luego esta información sera enviada a la plataforma web, lo que contribuirá a un mejor entendimiento y manejo de los microclimas presentes en la universidad.

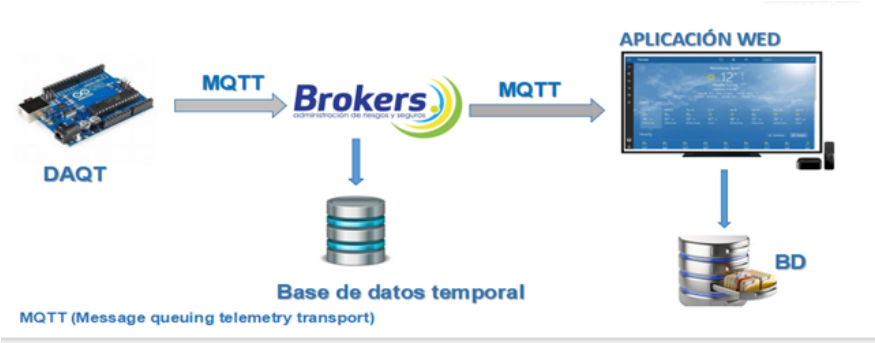


Figura 2.1. Propuesta de Solución(Fuente: Elaboración propia).

2.2. Fase de Exploración

Dando comienzo a las fases de la metodología de desarrollo XP está la fase de exploración. En esta se define el alcance general del proyecto. Donde el cliente define lo que necesita mediante la redacción de sencillas historias de usuarios. Los programadores estiman los tiempos de desarrollo en base a esta información. Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

2.2.1. Historias de Usuarios

El primer paso de cualquier proyecto que siga la metodología XP es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas dife-rencias: Constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados, etc. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. Cuando llega la hora de implementar una historia de usuario, el cliente y los desa-rrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia.

En el proceso de desarrollo del sistema se definió para mantener la organización, el nivel de prioridad de cada uno de los requisitos obtenidos en la entrevista con el cliente. Los niveles se dividen en:

- Alta: Funcionalidades que son sumamente primordiales en el sistema, que por su función no deberían faltar en el editor, es decir, el sistema depende de estas para su correcto funcionamiento.
- Media: Cuando el cliente cree que son necesarias, no son muy primordiales, pero otras dependen de estas para poder ejecutarse.
- Bajo: Funcionalidades que su existencia en el sistema no es significativa, es decir, este puede funcionar correctamente sin estas.

El riesgo en desarrollo se determinó mediante los siguientes indicadores:

- Alto: Cuando en la implementación de las HU pueden surgir errores que lleven a la inoperatividad del código.
- Medio: Cuando en la implementación de las HU pueden existir errores que retrasen la entrega del producto.
- Bajo: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que conlleven perjuicios para el desarrollo del proyecto.

Se definieron los siguientes requisitos funcionales:

1. Autenticar usuario.
2. Registrar usuario.
3. Modificar usuario.
4. Eliminar usuario.
5. Buscar usuario.
6. Registrar dispositivo.
7. Eliminar dispositivos.
8. Modificar dispositivos.
9. Buscar dispositivos.
10. Visualizar variables meteorológicas.
11. Editar perfil.
12. Registrar variables meteorológicas.
13. Modificar variables meteorológicas.
14. Eliminar variables meteorológicas.
15. Buscar variables meteorológicas.
16. Registrar municipios.
17. Modificar municipios.
18. Eliminar municipios.
19. Buscar municipios.
20. Registrar provincia.
21. Modificar provincia.
22. Eliminar provincia.
23. Buscar provincia.
24. Visualizar Historial de datos.
25. Mostrar ubicación en el mapa.
26. Notificar Alerta.

A continuación se detallan las mismas:

Historia de usuario

Descripción de la historia de usuario

Nombre: nombre que tiene la historia de usuario.

Número: es un identificador de la historia de usuario.

Programador: es el encargado de implementar el requisito.

Iteración: hace referencia a la iteración donde se va a realizar la historia de usuario.

Prioridad: indica la importancia relativa de la historia de usuario en comparación con otras historias del sistema.

Descripción: proporciona una explicación más detallada de lo que se espera lograr con la historia de usuario.

Riesgo en desarrollo: posibles desafíos o incertidumbres asociados con la implementación exitosa de esa historia en particular.

Tiempo real: es el tiempo límite.

Observación: es el monitoreo mediante el desarrollo.

Tabla 2.1. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Autenticar usuario.
Usuario: Daimarilis Guillot Reyes	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Daimarilis Guillot Reyes	
Descripción: El sistema debe permitirle al usuario autenticarse al sistema después de registrarse, ser revisada y aprobada su solicitud por el administrador del sistema después de introducir los datos requeridos usuario y contraseña para poder realizar las funciones del sistema.	
Observaciones:	

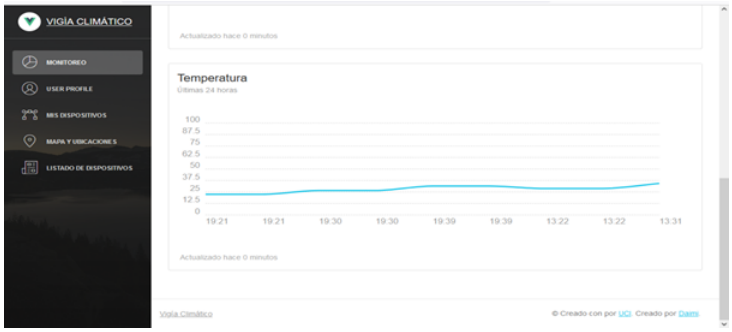


Tabla 2.2. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Visualizar variables meteorológicas.
Usuario: Daimarilis Guillot Reyes	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Daimarilis Guillot Reyes	
Descripción: El sistema debe permitirle al usuario una vez autenticado en el sistema con usuario y contraseña correspondientes visualizar las variables meteorológicas de estación meteorológica así como otras funciones del sistema.	
Observaciones:	

2.2.2. Requisitos no funcionales

Los requerimientos no funcionales son las características que hacen al producto atractivo, usable, rápido y confiable. ISO 25010 es parte de la serie ISO 25000, que proporciona orientación sobre los requisitos de calidad, evaluación y mejora del software. Un modelo de calidad de producto de software que consta de ocho características principales: adecuación funcional, eficiencia de rendimiento, compatibilidad, usabilidad, confiabilidad, seguridad, mantenibilidad y portabilidad (NC ISO/ IEC 25010, 2016). Cada característica se divide en subcaracterísticas que describen diferentes aspectos de la calidad. Por ejemplo, la usabilidad incluye subcaracterísticas como la capacidad de aprendizaje, la operatividad, la protección contra errores del usuario y la estética de la interfaz de usuario. ISO 25010 también define un conjunto de características de calidad en el uso, que son el grado en que un producto de software cumple con los objetivos de los usuarios en un contexto específico de uso **(andrei2023)**

Usabilidad:

(RNF) 1 El sistema será basado en la web con principios de aprendizaje baja que pueda ser usada por cualquier persona que posea un nivel básico de conocimientos de computación.

(RNF) 2 Deberá utilizar nombres sugerentes para lograr que el usuario encuentre lo que busca en el menor tiempo posible, las acciones a realizar serán fáciles de acceder.

(RNF) 3 El sistema debe poseer un diseño adaptable con el fin de garantizar la adecuada visualización

en múltiples computadores personales u de escritorio.

Seguridad:

(RNF) 4 El sistema debe usar roles para especificar los privilegios de cada usuario.

(RNF) 5 Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos.

Hardware:

(RNF) 6 El uso en las máquinas clientes los requerimientos serán menores. Es necesario que estas posean al menos 1GB de memoria RAM.

(RNF) 7 Los hardware clientes y servidor deben poseer una tarjeta de red que permita acceder y responder a las peticiones respectivamente.

Software

(RNF) 8 Para acceder a la aplicación web el dispositivo del cliente debe tener sistema operativo Linux o Windows versiones superiores a Windows 7.

(RNF) 9 Los dispositivos del cliente debe tener instalado algún navegador web, se recomienda Google Chrome o Mozilla Firefox.

2.2.3. Usuarios del sistema

Tabla 2.3. Usuarios del Sistema.

Rol	Descripción
Administradores	Es el encargado de gestionar todas las funciones de la estación meteorológica como la actualización de las variables meteorológicas.
Usuario	Su función específica es la visualización de datos del sistema específicamente las variables de medición de la estación meteorológica.

2.3. Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario, posteriormente los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. La estimación de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Las historias generalmente son de 1 a 2 puntos de estimación. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad"de desarrollo, establecida en puntos por iteración donde cada punto equivale a

una semana y una semana equivalen a 5 días hábiles y un día equivale a 8h de trabajo, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

2.3.1. Estimación de esfuerzos por cada historia de usuario.

La estimación de esfuerzos es el proceso de determinar cuánto tiempo llevará completar una historia de usuario. Ayuda al equipo de desarrollo a planificar su trabajo y a establecer plazos realistas. Para realizar la estimación de esfuerzos, primeramente, hay que leer la historia de usuario y comprenderla, luego se descompone el trabajo en tareas más pequeñas, se estima el esfuerzo de cada tarea y se suman para obtener una estimación del esfuerzo total de la historia de usuario. Comprender correctamente el número de iteraciones por cada historia de usuario permite obtener retroalimentación temprana del proyecto, adaptarse a cambios, entregar valor incremental y controlar tiempo y costos de manera efectiva.

Iteración	Historias de Usuarios		Puntos estimados
1	1	Autenticar usuario	1.0
	2	Gestionar usuarios	1.0
	3	Gestionar dispositivos	2.0
2	4	Gestionar variables meteorológicas	1.0
	5	Gestionar municipios	1.0
	6	Gestionar provincias	1.0
	7	Visualizar variables meteorológicas	1.0
3	8	Visualizar Historial de datos	1.0
	9	Editar perfil	0.40
	10	Mostrar ubicación en el mapa	0.40
	11	Notificar alertas	0.80
total		11	10 semanas con 3 días

Figura 2.2. Plan de iteraciones.(Fuente: Elaboración propia).

2.4. Plan de iteraciones y entrega:

Después de haber seleccionado un conjunto de Historias de Usuarios y una estimación previa de esfuerzo, finalmente comienza la planificación de la implementación del sistema, especificando la prioridad

según su valor para el cliente y su complejidad técnica, organizadas por iteraciones y sus posibles fechas de liberación.

Tabla 2.4. Plan de iteraciones y entrega (Fuente: Elaboración propia).

Iteración	Historia de usuario	Duración por semanas
1	Autenticar usuario Gestionar usuarios Gestionar dispositivos	4.0 equivale a 4 semanas
2	Gestionar variables meteorológicas Gestionar municipios Gestionar provincias Visualizar variables meteorológicas	4.0 equivale a 4 semanas.
3	Editar perfil Mostrar ubicación en el mapa Notificar alertas Visualizar historial de datos	2.6 equivale a 2 semana y 3 días.
Total	11	10 semanas y 3 días

Iteración 1:

Durante esta iteración se va a establecer la infraestructura del proyecto, así como una primera muestra del producto, donde el cliente pueda probar algunas de las funcionalidades como autenticarse en el sistema y realizar las diferentes gestiones del sistema. Se enfocará en la implementación de HU de alta prioridad relacionada con la etapa inicial del proceso, estableciendo una base fundamental de la arquitectura.

Iteración 2:

Durante esta iteración se seguirán implementando HU, que corresponden a los procesos que deben responder el sitio como: visualizar variables meteorológicas, gestionar provincias, gestionar municipios y gestionar variables meteorológicas.

Iteración 3:

Durante esta iteración, se enfocará en la implementación de HU de prioridad media proporcionando al cliente la comodidad en la gestión de otras tareas asociadas a las de mayor prioridad tales como: mostrar ubicación en el mapa, editar perfil y notificar alerta.

2.4.1. Desarrollo del plan de iteraciones

El ciclo de vida de XP se enfatiza en el carácter iterativo e incremental del desarrollo, una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas que en el caso de XP corresponden a un conjunto de historias de usuario, esas historias de usuario atraviesan por un proceso de planificación en el cual se define la duración de cada iteración a la que pertenecen. Para ellos se

realiza el plan de iteraciones por cada una de las iteraciones, la duración en semanas para definir entre todas las iteraciones la duración del proyecto y el plan de entregas.

Al realizarse las iteraciones debe coincidir el número total de puntos de las HU a cada una de las iteraciones que se realizaron, para que se cumpla satisfactoriamente. En la tabla anterior se muestran las HU correspondientes a cada iteración con el total de días estimados que debe demorarse el equipo de desarrollo en realizarlas, en la siguiente tabla, se muestra el plan de entrega con fechas de cada una de las iteraciones realizadas anteriormente.

Iteraciones	Fecha de inicio	Fecha de fin
1	1/04/2024	27/04/2024
2	29/04/2024	24/05/2024
3	27/05/2024	12/06/2024

Figura 2.3. Plan de entregas de las iteraciones.(Fuente: Elaboración propia)..

2.4.2. Patrón Arquitectónico

Un patrón arquitectónico brinda la descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseños específicos, y presenta un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que lo constituyen, sus responsabilidades desarrollos, así como también la forma como estos colaboran entre sí. El patrón arquitectónico es quien define la estructura básica de la aplicación.**(larman2016)**

Arquitectura de software

Un estilo arquitectónico es una lista de tipos de componentes que describen los patrones o las interacciones a través de estos. Un estilo influye en toda la arquitectura del software y puede combinarse en la propuesta de solución. Los estilos ayudan a un tratamiento estructural que concierne más bien a la teoría, la investigación académica y la arquitectura en el nivel de abstracción más elevado, expresando la arquitectura en un sentido más formal y teórico. **(sommerville2005)**

Para la solución propuesta se determinó el empleo de Django se suele llamar un frameworks Modelo Vista Plantilla (MVT por sus siglas en inglés) debido a que está fuertemente influenciado por Modelo Vista Controlador (MVC) y es incluso posible argumentar que la terminología MVC es el único patrón de cambios en Django. Las tres capas básicas son el modelo, la vista, y la plantilla. **(sommerville2005)**

Modelo (Model): En la aplicación de Django, se define un modelo que representa los datos relacionados con MQTT, como los mensajes publicados, los suscriptores, los temas, etc. Este modelo se encargará de interactuar con MQTT a través de una biblioteca MQTT, para enviar y recibir datos. El modelo define los datos almacenados, se encuentra en forma de clases de Python, cada tipo de dato que debe ser almacenado

se encuentra en una variable con ciertos parámetros y posee métodos también. Todo esto permite indicar y controlar el comportamiento de los datos.

Vista (View) : En la vista, se puede manejar las solicitudes de la aplicación web y coordinar la interacción entre el modelo y la plantilla. Se puede definir vistas que se encarguen de publicar mensajes en un tema específico, suscribirse a un tema o recibir mensajes MQTT y mostrarlos en la interfaz de usuario. La vista se presenta en forma de funciones en Python, su propósito es determinar qué datos serán visualizados, entre otras cosas más. El ORM (Object Relational Mapping) de Django permite escribir código Python en lugar de SQL para hacer las consultas que necesita la vista. También se encarga de tareas conocidas como el envío de correo electrónico, la autenticación con servicios externos y, la validación de datos a través de formularios .

Template (Template): En el template, se puede utilizar filtros de Django para mostrar los datos recibidos de MQTT en la interfaz de usuario. Es la encargada de personalizar la apariencia de la página web para mostrar la información de MQTT de manera adecuada. La plantilla es la encargada de recibir los datos de la vista y luego los organiza para la presentación al navegador web. La plantilla es básicamente una página HTML con algunas etiquetas extras propias de Django, en sí no solamente crea contenido en HTML (también XML, CSS, JavaScript, CSV, etc.)

Controlador (Control): En Django, el controlador está implícito en el framework web. Utiliza las vistas para manejar la lógica de negocio y las interacciones con MQTT. Utiliza las bibliotecas MQTT para realizar las operaciones MQTT necesarias, como publicar y suscribirse a temas.

Los modelos, vista y plantillas están estrechamente integrados en el frameworks permitiendo un desarrollo rápido y escalable de aplicaciones web.

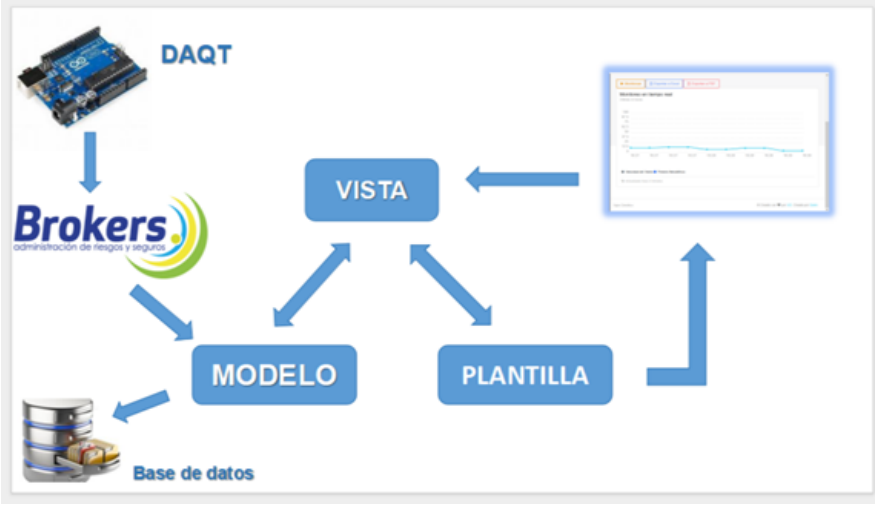


Figura 2.4. Arquitectura de Software(Fuente: Elaboración propia).

En este diagrama, las DAQT se comunican con el broker MQTT para enviar los datos recopilados por la estación meteorológica. La interacción entre estos componentes se realiza a través de llamadas y co-

municación con el protocolo MQTT en el flujo de control de la aplicación comienza en las DAQT quien envía la información hacia el broker a través del protocolo MQTT y luego es envía la información hacia App(modelo). La vista se comunica con el modelo para obtener los datos de MQTT y realizar operaciones con ellos. Luego, la vista utiliza la plantilla para mostrar los datos recibidos en la interfaz de usuario.

2.5. Patrones del diseño

Un patrón es una forma de dar solución a un problema, con un nombre y que es aplicable a otros contextos, cuenta con una sugerencia sobre la manera de usarlo en situaciones nuevas. Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.**(larman2016)**

Patrones GRASP

Los patrones de software para la asignación de responsabilidades (GRASP), describen los principios esenciales de la asignación de responsabilidades a objetos, por tanto, fueron considerados en la confección de las clases que componen el modelo de diseño. **(larman2016)**

Alta cohesión: cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Este patrón permitirá al equipo de desarrollo que la implementación de las clases encargadas de codificar la información sea fácil de comprender, reutilizar, mantener y poco susceptibles a cambios. En el sistema implementado se agruparon las clases según la funcionalidad de cada una

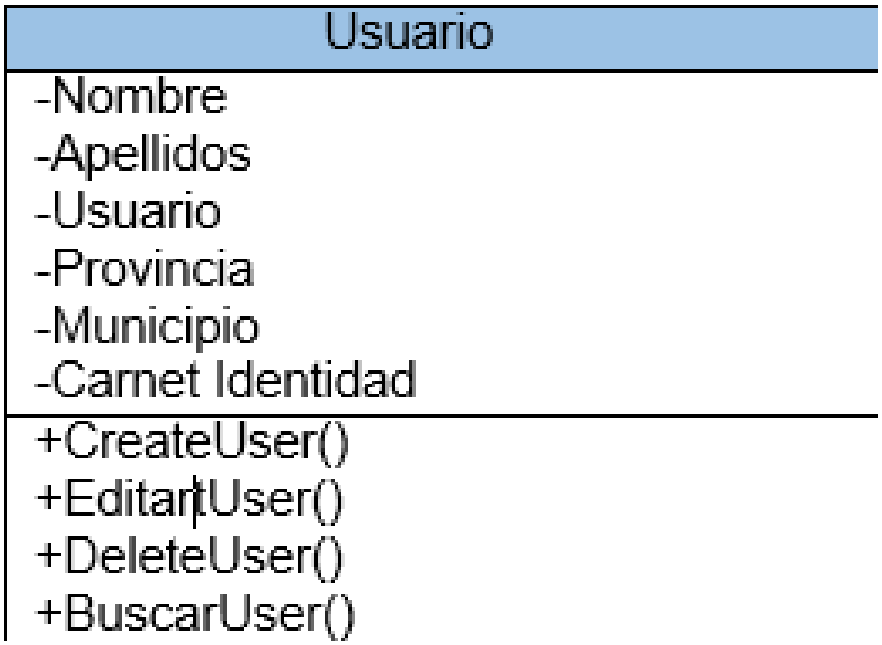


Figura 2.5. Patron alta cohesión(Fuente: Elaboración propia).

Bajo acoplamiento: Debe haber pocas dependencias entre las clases. El propósito de este patrón es

aumentar la reutilización y eliminar las dependencias entre las clases para propiciar un fácil mantenimiento y entendimiento, Django implementa este patrón por defecto evitando altas dependencias.

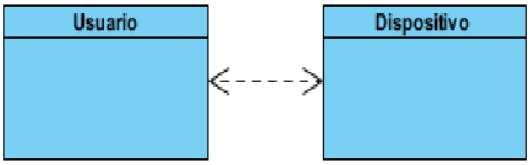


Figura 2.6. Patrón bajo acoplamiento(Fuente: Elaboración propia).

Controlador: asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

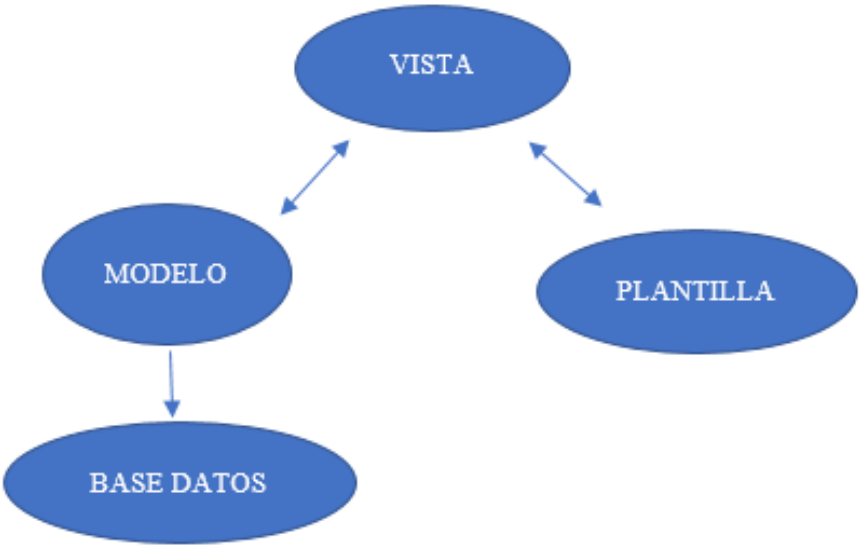


Figura 2.7. Patrón Controlador(Fuente: Elaboración propia).

Patrones GoF:

Describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado

a objetos entre los componentes, es decir, no importan los cambios que se realicen en alguno de estos componentes influirán de manera mínima en el resto. **(larman2016)**

Patron Observador

Este patrón es adecuado para cualquier escenario que requiera notificaciones push. El modelo define un proveedor (también conocido como un tema o una observable) y cero, uno o más observadores. Los observadores se registran con el proveedor y, siempre que se produce una condición predefinida, un evento o un cambio de estado, el proveedor lo notifica automáticamente a todos los observadores mediante la llamada a uno de los delegados. En esta llamada al método, el proveedor puede proporcionar también información sobre el estado actual a los observadores. **(microsoft2023)** Este patrón es útil para implementar el manejo de eventos nes.y notificacio.

Permite que un objeto (el sujeto) notifique a otros objetos (los observadores) sobre cambios en su estado.

@receiver(post save, sender=Dispositivo)

def subscribe to device(sender, instance, created, **kwargs):

if created:

instance.subscribe

Aquí, se utiliza el sistema de señales de Django para observar el evento de guardado de un objeto Dispositivo. Cuando se crea un nuevo dispositivo (created es True), se llama al método subscribe para suscribirse a un tema MQTT.

Patrón Factory

Define una interfaz para crear un objeto, pero permite a las subclases alterar el tipo de objetos que se crean. Aunque no hay una implementación directa de un patrón Factory en el código, el uso del método subscribe en el modelo Dispositivo puede considerarse una forma de encapsular la lógica de creación de un cliente MQTT. def subscribe(self):

client = mqtt.Client()

client.on message = on message

client.connect("localhost", 1883, 60)

topic = f"dispositivo/self.nombre identificador/data"

client.subscribe(topic)

client.loop start()

print(f"Suscrito a topic")

La creación del cliente MQTT y la configuración de suscripción se encapsulan dentro del método subscribe, lo que permite cambiar la lógica de conexión sin afectar a otras partes del código.

Patrón Composite

Permite tratar objetos individuales y compuestos de manera uniforme.

class Dispositivo(models.Model):

variables = models.ManyToManyField(Variable)

El modelo Dispositivo tiene una relación ManyToMany con Variable, lo que permite tratar múltiples

variables como una colección. Esto puede ser visto como una implementación del patrón Composite donde un dispositivo puede tener múltiples variables asociadas.

Patrón Command

Encapsula una solicitud como un objeto, lo que permite parametrizar a los clientes con colas, solicitudes y operaciones.

class UserDispositivosCount(APIView):

permission classes = [IsAuthenticated]

def get(self, request):

user = request.user

count = user.dispositivos.count()

return Response('count': count)

Este APIview puede ser considerado como una implementación simplificada del patrón Command, donde la acción principal es contar y devolver el número de dispositivos asociados al usuario.

Patron decorador

El patrón Decorator permite añadir dinámicamente nuevos comportamientos a objetos colocándolos dentro de objetos especiales que los envuelven (wrappers). Esto permite extender el comportamiento de una función o clase sin modificar su código original.

@api view(['GET'])

@permission classes([IsAuthenticated])

def consultar id(request, id):

Esta función utiliza el patrón Decorator para aplicar permisos de autenticación a una vista API específica.

2.6. Diseño de base datos del sistema

El diseño de la base de datos es una colección de pasos que ayudan a crear, implementar y mantener los sistemas de administración de datos de una empresa. El propósito principal del diseño de una base de datos es producir modelos físicos y lógicos de diseños para el sistema de base de datos propuesto.

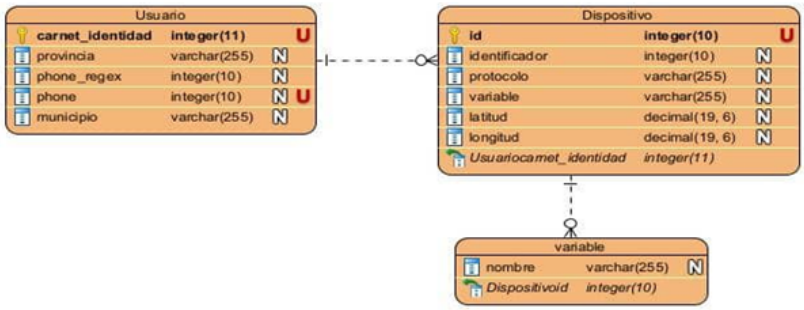


Figura 2.8. Diseño de la base de datos(Fuente: Elaboración propia).

2.7. Tarjetas CRC

La utilización de tarjetas CRC (Class-Responsibility-Collaboration) es una técnica de diseño orientado a objetos. El objetivo de la misma es hacer, mediante tarjetas, un inventario de las clases que se van a necesitar para implementar el sistema y la forma en que van a interactuar, de esta forma se pretende facilitar el análisis y discusión de las mismas por parte de varios actores del equipo de proyecto con el objeto de que el diseño sea lo más simple posible verificando las especificaciones del sistema.**(jummp2012)**

Las tarjetas CRC son una técnica utilizada en el diseño de software orientado a objetos. Su objetivo es ayudar a los desarrolladores a identificar, organizar y documentar las responsabilidades de cada clase dentro de un sistema.

2.7.1. Principales pasos para trabajar con tarjetas CRC:

Identificar las clases: Durante el análisis del problema, se identifican las principales clases y conceptos que formarán par-te del sistema.

Definir las responsabilidades: Para cada clase, se define cuáles son sus principales responsabilida-des, es decir, qué tareas y funcionalidades debe llevar a cabo.

Identificar las colaboraciones: Se determinan qué otras clases necesita una clase para poder cum-plir con sus responsabilidades. Esto establece las relaciones y colaboraciones entre las clases.

Documentar en tarjetas: Toda esta información se plasma en tarjetas físicas o digitales, donde se registra el nombre de la clase, sus responsabilidades y las clases con las que colabora.

Las tarjetas CRC ayudan a los equipos de desarrollo a:

Entender mejor la estructura y diseño del sistema.

Identificar responsabilidades y colaboraciones entre clases.

Facilitar la comunicación y el trabajo en equipo.

Tomar decisiones de diseño y refactorización.

Documentar de forma sencilla y visual la arquitectura del software.

Tabla 2.5. Tarjeta CRC # 1

Tarjeta CRC	
Clase: Gestionar Usuarios	
Responsabilidad	**Colaboración**
• CreateUser() • DeleteUser() • BuscarUser() • EditarUser()	AutenticacionUsuarios Usuarios

Tabla 2.6. Tarjeta CRC # 2

Tarjeta CRC	
Clase: Gestionar Dispositivos	
Responsabilidad	**Colaboración**
• CreateDispositivo() • DeleteDispositivo() • BuscarDispositivo() • EditarDispositivo()	Dispositivos Variables meteorológicas

2.8. Conclusiones del capítulo

En el capítulo se abordó diferentes elementos relacionados con el desarrollo de la propuesta de solución como el modelo conceptual que captura los conceptos claves y las relaciones entre ellos.

Con el análisis y caracterización del dominio se permitió identificar los requisitos funcionales y no funcionales, que establecen las funcionalidades y características generales que debe tener el sistema.

Se encapsularon los requisitos en historias de usuarios, lo que permitió comunicar los requisitos del sistema al equipo de desarrollo.

Se diseño el patrón arquitectónico en el que se basa el framework Django, se realiza un análisis y se ejemplifica el uso de los patrones del diseño, que propiciaron principios y soluciones para el diseño del software.

CAPÍTULO 3

IMPLEMANTACIÓN Y PRUEBA

En este capítulo se desarrollan las dos últimas fases que plantea la metodología XP: Desarrollo y Pruebas. En el caso de la primera se da inicio con las tareas de ingeniería, donde cada HU se convierte en tareas a cumplir por los programadores. En la última fase se realizan pruebas de aceptación y unitarias para verificar el correcto funcionamiento del sistema que se obtuvo como resultado de la presente investigación.

3.1. Fase de implementación

Dentro del ciclo de vida de un sistema informático se encuentra la fase de implementación. Esta es la fase más costosa y que consume más tiempo. Se dice costosa ya que muchas personas, herramientas y recursos están involucrados. La metodología XP propone que las historias de usuario deben ser implementadas en dependencia de la iteración a la que pertenezcan. **(solis2003)**

3.1.1. Tareas de ingeniería para la Iteración

Asociado a cada iteración, se encuentra la planificación de las tareas de ingeniería o programación. Cada una se caracteriza por estar asociada a una historia de usuario y varias tareas de ingeniería pueden pertenecer a una historia de usuario. Para la confección de cada una de las tareas se utilizaron tablas cuyo modelo cuenta con los siguientes campos:

- No. de tarea: numeración continua que identifica a la tarea.
- No. de HU: número de la HU a la que pertenece.
- Nombre de la tarea: identificación literal de la tarea.
- Tipo de tarea: tipo de tarea, dígase diseño, desarrollo, prueba.
- Puntos estimados: representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizará para su realización.
- Fecha inicio: fecha estimada de inicio de realización.
- Fecha fin: fecha estimada de fin de realización.

- Descripción: se describe en qué consiste la tarea y qué elementos deben cumplirse para declarar la tarea terminada.

Seguidamente, se presentan las tareas de ingeniería perteneciente a cada historia de usuario

Tabla 3.1. Tarea de ingeniería # 1

Tarea	
Número de tarea: 1	**Número de Historia de usuario:** 1
Nombre de la tarea: Modificar dispositivo	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.40
Fecha de inicio: 22 de abril de 2024	**Fecha de fin:** 24 de abril de 2024
Programador responsable: Daimarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al administrador modificar un dispositivo con protocolo, variables, descripción y id.	

Tabla 3.2. Tarea de ingeniería # 2

Tarea	
Número de tarea: 2	**Número de Historia de usuario:** 1
Nombre de la tarea: Eliminar dispositivo	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.40
Fecha de inicio: 19 de abril de 2024	**Fecha de fin:** 21 de abril de 2024
Programador responsable: Daimarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al administrador eliminar un dispositivo con protocolo, variables, descripción y id.	

3.2. Diagrama de despliegue

Los diagramas de despliegue se utilizan para visualizar los procesadores/ nodos/dispositivos de hardware de un sistema, los enlaces de comunicación entre ellos y la colocación de los archivos de software en ese hardware.

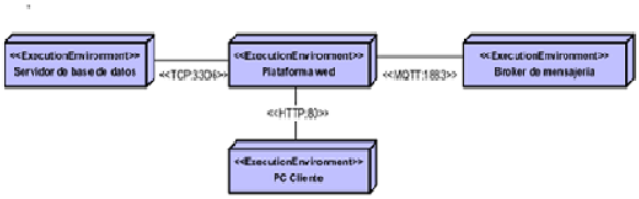


Figura 3.1. Diagrama Despliegue(Fuente: Elaboración propia).

PC-Cliente: Es el nodo que representa la estación de trabajo que permite al usuario mediante el protocolo HTTP y el puerto 8080, acceder a la plataforma.

Plataforma web: Es la estación de trabajo que hospeda el código fuente de la aplicación y que le brinda al usuario las interfaces para realizar los procesos del sistema. Esta estación se comunica con el servidor de base de datos donde se almacenan los datos de la aplicación realizando la comunicación mediante el protocolo TCP (Transmission Control Protocol).

Servidor de Base de Datos (PostgreSQL): Este servidor es el encargado del almacenamiento de los datos del sistema. Se comunica con el servidor de aplicaciones del sistema mediante el protocolo TCP: 5432, posibilitando el acceso mediante el usuario con privilegios para las operaciones determinadas a realizarse en el mismo.

HTTP: Protocolo de transferencia de hipertexto seguro ("Hypertext Transfer Protocol") es un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML, es el que impulsa todo internet. Los navegadores web utilizan este protocolo para solicitar páginas web a los servidores. El servidor devuelve todos los datos necesarios en código HTML para que puedan mostrarse en el navegador, en este caso por el puerto 8080.

TCP: Protocolo de la capa de transporte (Transmission Control Protocol) es un protocolo de comunicación utilizado en redes de computadoras para garantizar la transmisión confiable de paquetes de datos en este caso usando TCP 5432.

Broker de mensajería: Es software encargado de resivir la información de las DAQT y la distribuirla en la aplicación wed para su procesamiento y análisis.

3.3. Estrategia de pruebas

Uno de los pilares de la metodología XP es el proceso de pruebas, pues anima a probar constantemente, o tanto como sea posible. Permitiendo aumentar la calidad de los sistemas, reduciendo el número de errores

no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección .

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática. Y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de com-probar que dicha funcionalidad sea la esperada por el cliente.

Pruebas software

Son propensas a tener fallos. A veces, pueden contribuir al fracaso de cualquier proyecto de software, e impactar de forma negativa en toda una empresa. Los tiempos de desarrollo, los entornos de pro-gramación, las diferencias entre versiones, todo influye para que, incluso con la máxima dedicación, puedan darse fallos que empañen la imagen y a veces la reputación, de una organización. Surge por tanto la necesidad de asegurar en lo posible, la calidad del producto.**(pressman2015)**

Las pruebas de software son las investigaciones empíricas y técnicas cuyo fin es proporcionar información objetiva e independiente sobre la calidad del producto. Esta actividad forma parte del proceso de control de calidad global. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software y dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento del proceso de desarrollo.

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación **(pressman2010)**). La metodología aplicada define que no debe existir ninguna funcionalidad en el programa que no haya sido probada. El equipo de desarrollo estará siempre acompañado por el cliente para convenir los detalles de los requerimientos y así poder implementarlos, probarlos y validarlos. De esta forma se brinda un resultado más completo para un producto final de manera creciente:

- Pruebas internas: se verifica el resultado de la implementación al probar cada construcción, y al incluir tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como diseños de casos de prueba, listas de chequeo y de ser posibles, componentes de prueba ejecutables para automatizar las pruebas.
- Pruebas de liberación: pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
- Pruebas de Aceptación: es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las que el software fue construido.

Las pruebas de software son un elemento crítico para garantizar el correcto funcionamiento de la aplicación. Entre sus metas se encuentra:

- Detectar defectos en el software.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.

- Identificar y asegurar que los fallos encontrados durante el proceso de prueba, se han corregido antes de entregar el software al cliente.

3.4. Pruebas de aceptación

Las pruebas de aceptación, también llamadas pruebas funcionales son supervisadas por el cliente basándose en los requerimientos tomados de las historias de usuario. En todas las iteraciones, cada una de las historias de usuario seleccionadas por el cliente deberá tener una o más pruebas de aceptación, de las cuales deberán determinar los casos de prueba e identificar los errores que serán corregidos. Las pruebas de aceptación son pruebas de caja negra, que representan un resultado esperado de determinada transacción con el sistema. Para que una historia de usuario se considere aprobada, deberá pasar todas las pruebas de aceptación elaboradas para dicha historia. Es importante resaltar la dife-rencia entre las pruebas de aceptación y las unitarias en lo que al papel del usuario se refiere. Mientras que en las pruebas de aceptación juega un papel muy importante seleccionando los casos de prueba para cada historia de usuario e identificando los resultados esperados, en las segundas no tiene ninguna intervención por ser de competencia del equipo de programadores **(pressman2015)**

Las pruebas de aceptación tienen más peso que las unitarias ya que constituyen un indicador de la satisfacción del cliente con la solución además de marcar el final de una iteración y el comienzo de la siguiente. Se recomienda que el cliente sea quien diseñe estas pruebas o que al menos participe de manera activa en el proceso.

Técnica de Iadov

La técnica de Iadov se centra en la creación de casos de prueba basados en la especificación de requisitos y en el comportamiento esperado del sistema. Se basa en la identificación de situaciones clave que reflejan el uso real del sistema.

El cliente debe especificar uno o diversos escenarios para comprobar que una HU ha sido correctamente implementada. Para la representación de las pruebas de aceptación se definieron los siguientes elementos:

- Código de la prueba: representa al caso de prueba, incluye el número de HU y de la prueba.
- Nombre de Historia de Usuario: nombre de la historia de usuario.
- Descripción de la prueba: acción que debe realizar el sistema.
- Condiciones de ejecución: describe las características y elementos que debe contener el siste-ma para realizar el caso de prueba.
- Resultado Esperado: descripción de la respuesta del sistema ante el caso de prueba.
- Resultado Obtenido: respuesta visual del sistema después de realizar el caso de prueba.
- Evaluación de la Prueba: clasificación de la prueba en satisfactoria o insatisfactoria.

Para la primera iteración, se definieron un total de 9 casos de pruebas. Todas enfocadas a evaluar la implementación de algunas funcionalidades del sistema. A continuación se muestran algunos casos de pruebas de la primera iteración.

Tabla 3.3. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_PA1	**Historia de usuario:** 1
Nombre: Autenticar Usuario.	
Descripción: El usuario accede al sistema con su usuario y contraseña.	
Condiciones de ejecución:	
Pasos de ejecución: 1. Introducir usuario y contraseña. 2. Presionar el botón Entrar.	
Resultados esperados: Satisfactorio	

Para la segunda iteración, se definieron un total de 13 pruebas de aceptación. Todas enfocadas a evaluar la implementación de algunas funcionalidades del sistema. A continuación se muestran algunos casos de pruebas de la segunda iteración.

Tabla 3.4. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU5_PA2	**Historia de usuario:** 5
Nombre: Visualizar variables meteorológicas.	
Descripción: Se visualizan las variables meteorológicas.	
Condiciones de ejecución:	
Pasos de ejecución: 1. El usuario se autentica en el sistema. 2. Presiona monitorear y selecciona un dispositivo. 3. Se visualizan las variables meteorológicas en tiempo real.	
Resultados esperados: Satisfactorio	

Para la tercera iteración, se definieron un total de 4 casos de prueba. Todas enfocadas a evaluar la implementación de las ultimas funciones del sistema.

Como resultado de las pruebas se realizaron un total 26 pruebas de aceptación para una primera iteración se realizaron un total de 9 pruebas de aceptación donde se obtuvo 1 no conformidad N/C asociada con errores ortográficos la cual fue solucionada satisfactoriamente. En una segunda iteración se realizaron un total de 13 prueba de aceptación arrojando 1 N/C asociada a la visualización de las variables meteorológicas que no se mostraban correctamente N/C que fue solucionada. En una tercera iteración se realizaron un total de 4 prueba de aceptación donde fue detectada una 1 N/C asociada con editar el perfil del usuario que fue solucionada.

En la siguiente figura se muestra el total de pruebas realizadas, las no conformidades y las no conformidades resueltas.

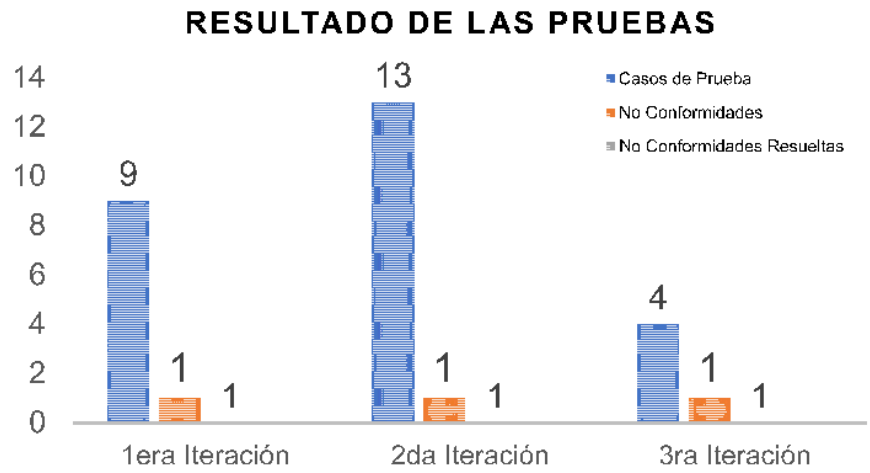


Figura 3.2. Resultado Prueba Aceptación(Fuente: Elaboración propia).

3.5. Pruebas unitarias

Las pruebas unitarias de software, también conocidas como unit testing, incluyen un conjunto de características y propiedades que permiten su funcionamiento, de modo que una de las principales metas de este tipo de pruebas es que permiten garantizar que cada una de las unidades de software analizadas se encuentran funcionando de la forma que deberían e independientemente. Cabe destacar que las pruebas unitarias de software funcionan mediante el mecanismo de aislar una parte determinada del código fuente, con el objetivo de asegurar su funcionamiento; es decir, son pruebas pequeñas que validan el comportamiento de un fragmento del código.

Para la realización de dichas pruebas se utilizó el marco de trabajo Django, en el siguiente código se refleja un segmento realizado a las funcionalidades. A continuación, se muestra la realización de estas pruebas: En esta prueba, vamos a verificar la funcionalidad de un sistema de gestión que permite gestionar dispositivos, usuarios, provincias, variables y municipios. Utilizaremos el framework de pruebas de Django para asegurarnos de que cada componente funcione correctamente.

Utilizando el comando: python manage.py test.

La prueba consta de varias clases heredadas de TestCase, cada una representando una entidad diferente del modelo:

1. TestVariableModel: Pruebas para el modelo Variable
2. TestProvinciaModel: Pruebas para el modelo Provincia
3. TestMunicipioModel: Pruebas para el modelo Municipio
4. TestDispositivoModel: Pruebas para el modelo Dispositivo

5. TestUsuarioModel: Pruebas para el modelo Usuario

6. TestRegistroVariableModel: Pruebas para el modelo RegistroVariable

Estas pruebas unitarias están diseñadas para asegurar la integridad y funcionalidad de los modelos Django utilizados en el sistema. Cubren aspectos clave como:

Creación correcta de objetos

Edición de atributos

Búsqueda eficiente de objetos

Eliminación de objetos

La prueba cubre varios modelos importantes del sistema:

Variable

Provincia

Municipio

Dispositivo

Usuario

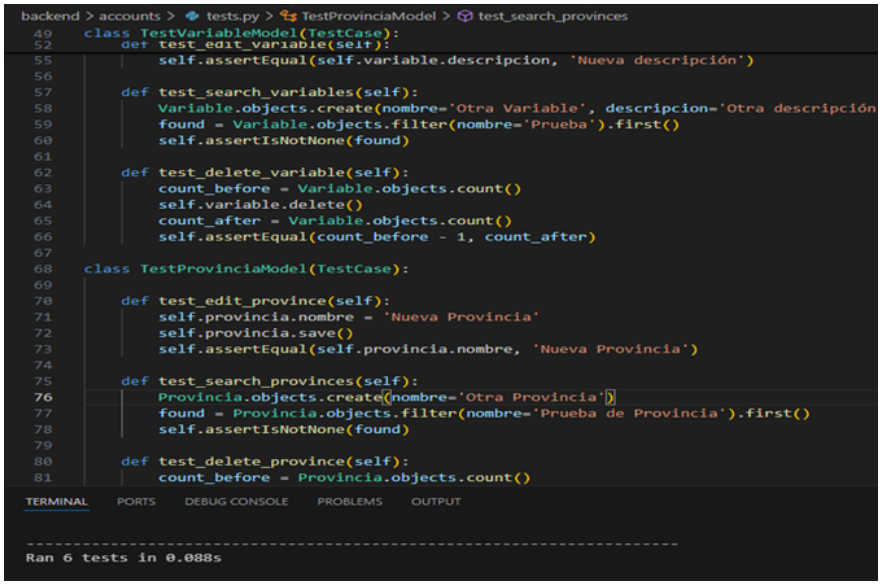


Figura 3.3. Resultado Prueba Unitarias(Fuente: Elaboración propia).

Como resultado se obtuvo que todas las pruebas pasaron, significando que los modelos funcionan correctamente y que la relación entre ellos se ha establecido como se esperaba, todas las funcionalidades de los modelo resultaron eficientes, porque lo que existe un correcto funcionamiento.

3.6. Pruebas No Funcionales

Los objetivos de las pruebas no funcionales son comprobar que el producto cumple las expectativas del usuario y optimizarlo. Verifica los factores que influyen en la usabilidad, fiabilidad, mantenimiento, porta-

bilidad y eficacia del producto. Probar estos elementos garantiza que el producto que se expone al mercado tiene la calidad adecuada y cumple las expectativas de los usuarios en cuanto a rendimiento, tiempos de carga y capacidad de uso antes de su lanzamiento. Se realizan pruebas de usabilidad, portabilidad, compatibilidad y eficiencia del desempeño en las diferentes etapas de iteraciones. En las pruebas no funcionales de usabilidad se usan herramientas automatizadas adaptadas a la aplicación en prueba. En el caso de los otros tres tipos de prueba no funcionales compatibilidad, portabilidad y eficiencia del desempeño no muestran defectos en las diferentes iteraciones. Cada uno de los criterios de usabilidad, compatibilidad y eficiencia del desempeño, se prueban resultando un producto final que satisface plenamente las expectativas del cliente, el cual queda satisfecho con la aplicación y las pruebas realizadas.**(andrei2023)**

CONCLUSIONES DEL CAPÍTULO

- El plan de entrega fue cumplido acorde al tiempo planificado para el desarrollo de las tareas por cada historia de usuario.
- Las pruebas de aceptación permitieron que el cliente comprobara el estado de cada una de las funcionalidades desarrolladas en el sistema.
- Las pruebas al software permitieron comprobar la calidad del sistema propuesto.
- Las tareas de ingeniería ayudaron al equipo de proyecto a lograr identificar, controlar, rastrear los requisitos y los cambios en cualquier etapa mientras se desarrollaba el proyecto.
- Las realizaciones de las pruebas de aceptación aprobaron el correcto funcionamiento y el cum-plimiento de los requisitos de software definidos para la aplicación.

Conclusiones

- El análisis realizado del estado del arte, fundamentó la necesidad de llevar a cabo el desarrollo de un sistema web que contribuye al proceso de visualización de las variables meteorológicas en La Universidad de las Ciencias Informáticas.
- El uso de la metodología XP como guía del proceso de desarrollo de software, así como las herramientas y tecnologías tales como el framework Django, y el lenguaje de programación Python facilitó el correcto desarrollo de la propuesta solución.
- Los requisitos del sistema, la arquitectura modelo vista plantilla y el uso de los patrones de diseño, permitieron obtener una aplicación web optima que cumple con las expectativas del cliente.
- La validación de la Aplicación Web para la Gestión de la visualización de datos meteorologicos en la estación de bajo costo, a partir de una estrategia de pruebas, permitió corroborar que el sistema satisface las necesidades del cliente.

Recomendaciones

En base a los resultados obtenidos se recomienda:

- Integrar la solución con el servidor NTFY de notificaciones a través de publicación/subcripción.
- Implementar una aplicacion móbil cliente que consuma las notificaciones y alertas meteorologicas.
- Probar sensores de gama media y alta para una mayor presición de las mediciones meteorologicas.

Generado con LaTeX: 30 de noviembre de 2024: 12:35 Noon

Apéndices

Apéndice1 entrevista

Serie de preguntas de la entrevista con el cliente.

Pregunta: ¿Porque es necesario la realización de una aplicación web para la visualización de los datos meteorológicos en la universidad de las ciencias informáticas ?

Respuesta: Es necesario para un mayor manejo de los microclimas existente en la universidad que ocasionan daños en la vida cotidiana y el medio ambiente provocando variaciones en la temperatura, humedad y velocidad del viento perjudicando la salud de las personas.

Pregunta: ¿Cuáles son las principales ventajas que ves en utilizar una estación meteorológica de bajo costo basada en IoT, en comparación con las estaciones tradicionales?

Respuesta: Las estaciones tradicionales suelen ser costosas y requieren mantenimiento regular. Un sistema IoT de bajo costo puede ofrecer mayor accesibilidad, permitiendo una mayor densidad de puntos de medición, mejorando la calidad de los datos para la predicción meteorológica a nivel local.

Pregunta: ¿Qué características consideras esenciales para que una estación meteorológica de bajo costo sea atractiva para los usuarios?

Respuesta: La facilidad de instalación y uso, la precisión de los datos, la disponibilidad de una interfaz web o móvil para la visualización, la capacidad de almacenar y acceder a los datos históricos, y la buena relación costobeneficio.

Pregunta: ¿Qué funcionalidades consideras esenciales para una aplicación web de visualización de datos meteorológicos?

Respuesta: Gráficos claros y fáciles de entender, datos históricos, alertas para eventos climáticos extremos, capacidad de compartir datos y una interfaz intuitiva, ubicación de la estación.

Pregunta: ¿Qué tipo de visualización de datos consideras más apropiada para este proyecto? (Gráficos, mapas, tablas, etc.)

Respuesta: Dependiendo del usuario final, una combinación de gráficos interactivos (líneas, barras, dispersión) y mapas podría ser ideal.

Pregunta: ¿Qué aspectos son importantes para diseñar una interfaz de usuario efectiva para la visualización de los datos meteorológicos?

Respuesta: La claridad, la facilidad de uso, la accesibilidad para diferentes dispositivos, la legibilidad de los datos y una estética visualmente atractiva son aspectos clave.

Pregunta: ¿Qué aspectos de la seguridad son importantes para una aplicación web que maneja datos meteorológicos de una estación IoT?

Respuesta: Proteger la comunicación entre la estación y el servidor con protocolos seguros (HTTPS), autenticar a los usuarios, validar los datos recibidos para prevenir ataques, y realizar copias de seguridad regulares de los datos y utilizar el protocolo de comunicación MQTT para el intercambio de información.

Apéndice2 Historias de Usuarios

Tabla 5. Historia de usuario # 3

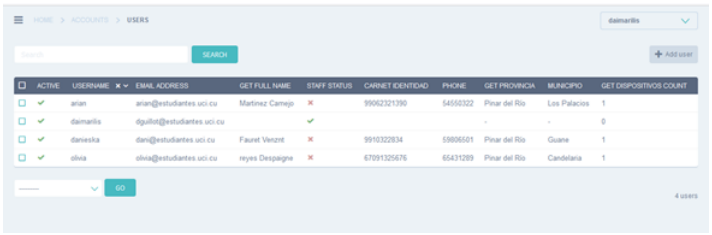
Historia de usuario	
Número: 3	**Nombre:** Gestionar dispositivos
Usuario: Daimarilis Guillot Reyes	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Daimarilis Guillot Reyes	
Descripción: El sistema debe permitirle al usuario con el rol de administrador después de haber iniciado sesión en el sistema con el usuario y contraseña correspondientes poder realizar funciones de insertar, modificar, buscar y eliminar los dispositivos de una estación meteorológica con cada unos de sus parámetros: • Identificador. • Descripción. • Municipio. • Provincias. • Protocolo. • latitud. • Longitud.	
Observaciones:	
Interfaz: 	

Tabla 6. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Gestionar usuarios
Usuario: Daimarilis Guillot Reyes	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Daimarilis Guillot Reyes	

Continúa en la próxima página

Tabla 6. Continuación de la página anterior

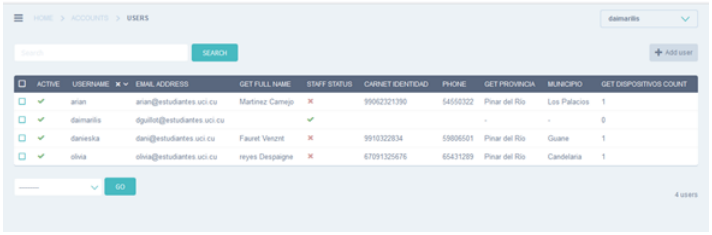
Descripción: El sistema debe permitirle al usuario con el rol de administrador después de haber iniciado sesión en el sistema con el usuario y contraseña correspondientes poder realizar funciones de insertar, modificar, buscar y eliminar los usuarios de una estación meteorológica con cada unos de sus parámetros: • Nombre. • Apellidos. • Usuario. • Carnet de identidad. • Teléfono. • Provincia. • Municipio.
Observaciones:
Interfaz: 

Tabla 7. Historia de usuario # 5

Historia de usuario	
Número: 5	**Nombre:** Gestionar Provincias
Usuario: Daimarilis Guillot Reyes	
Prioridad en negocio: Media	**Riesgo en desarrollo:** media
Puntos estimados: 6	**Iteración asignada:** 2
Programador responsable: Daimarilis Guillot Reyes	
Descripción: El sistema debe permitirle al usuario con el rol de administrador después de haber iniciado sesión en el sistema con el usuario y contraseña correspondientes poder realizar funciones de insertar, modificar, buscar y eliminar las provincias de una estación meteorológica con cada unos de sus parámetros: • Nombre.	
Observaciones:	

Continúa en la próxima página

Tabla 7. Continuación de la página anterior

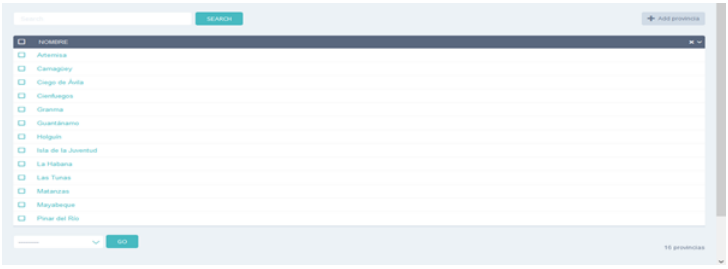
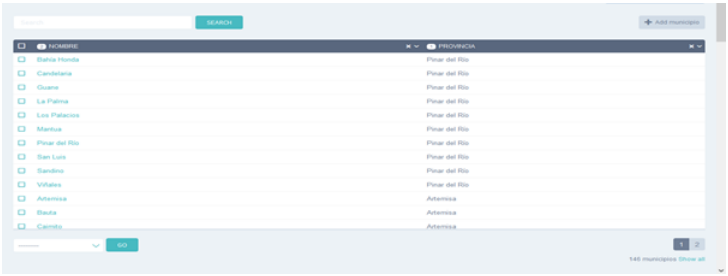
Interfaz: 

Tabla 8. Historia de usuario # 6

Historia de usuario	
Número: 6	**Nombre:** Gestionar Municipios
Usuario: Daimarilis Guillot Reyes	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 7	**Iteración asignada:** 2
Programador responsable: Daimarilis Guillot Reyes	
Descripción: El sistema debe permitirle al usuario con el rol de administrador después de haber iniciado sesión en el sistema con el usuario y contraseña correspondientes poder realizar funciones de insertar, modificar, buscar y eliminar los municipios de una estación meteorológica con cada unos de sus parámetros: • Nombre. • provincia.	
Observaciones:	
Interfaz: 	

Apéndice3 Tareas de Ingeniería

- HU_11 .

Tabla 9. Tarea de ingeniería # 3

Tarea	
Número de tarea: 3	**Número de Historia de usuario:** 1
Nombre de la tarea: Atenticar Usuario.	
Tipo de tarea: desarrollo	**Puntos estimados:** 1
Fecha de inicio: 10 de abril de 2024	**Fecha de fin:** 14 de abril de 2024
Programador responsable: Daiamarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al usuario autenticarse en el sistema para la visualización de las variables meteorológicas.	

- HU_1 Autenticar Usuario.
- HU_1 .

Tabla 10. Tarea de ingeniería # 4

Tarea	
Número de tarea: 4	**Número de Historia de usuario:** 3
Nombre de la tarea: Modificar usuario.	
Tipo de tarea: desarrollo	**Puntos estimados:** 1
Fecha de inicio: 6 de abril de 2024	**Fecha de fin:** 7 de abril de 2024
Programador responsable: Daiamarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al administrador modificar un usuario con nombre, apellidos, CI, provincia y municipio.	

- HU_3 Modificar usuario .
- HU_3 .

Tabla 11. Tarea de ingeniería # 5

Tarea	
Número de tarea: 5	**Número de Historia de usuario:** 3
Nombre de la tarea: Buscar usuario.	
Tipo de tarea: desarrollo	**Puntos estimados:** 1
Fecha de inicio: 8 de abril de 2024	**Fecha de fin:** 9 de abril de 2024
Programador responsable: Daiamarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al administrador buscar un usuario con nombre, apellidos, CI, provincia y municipio.	

- HU_3 Buscar usuario .

- HU_3 .

Tabla 12. Tarea de ingeniería # 6

Tarea	
Número de tarea: 6	**Número de Historia de usuario:** 4
Nombre de la tarea: Eliminar dispositivo.	
Tipo de tarea: desarrollo	**Puntos estimados:** 2
Fecha de inicio: 15 de abril de 2024	**Fecha de fin:** 18 de abril de 2024
Programador responsable: Daiamarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al administrador eliminar un dispositivo con protocolo, variables, descripción y id.	

- HU_4 Eliminar dispositivo.

- HU_.

Tabla 13. Tarea de ingeniería # 7

Tarea	
Número de tarea: 7	**Número de Historia de usuario:** 4
Nombre de la tarea: Insentar dispositivo.	
Tipo de tarea: desarrollo	**Puntos estimados:** 2
Fecha de inicio: 19 de abril de 2024	**Fecha de fin:** 22 de abril de 2024
Programador responsable: Daiamarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al administrador insertar un dispositivo con protocolo, variables, descripción y id.	

- HU_4 Insertar dispositivo.

- HU_4.

Tabla 14. Tarea de ingeniería # 8

Tarea	
Número de tarea: 8	**Número de Historia de usuario:** 4
Nombre de la tarea: Modificar dispositivo.	
Tipo de tarea: desarrollo	**Puntos estimados:** 2
Fecha de inicio: 23 de abril de 2024	**Fecha de fin:** 26 de abril de 2024
Programador responsable: Daiamarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al administrador Mofificar un dispositivo con protocolo, variables, descripción y id.	

- HU_4 Modificar dispositivo.
- HU_4.

Tabla 15. Tarea de ingeniería # 9

Tarea	
Número de tarea: 9	**Número de Historia de usuario:** 4
Nombre de la tarea: Buscar dispositivo.	
Tipo de tarea: desarrollo	**Puntos estimados:** 2
Fecha de inicio: 26 de abril de 2024	**Fecha de fin:** 29 de abril de 2024
Programador responsable: Daiamarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al administrador buscar un dispositivo con protocolo, variables, descripción y id.	

- HU_4 buscar dispositivo.
- HU_4.

Tabla 16. Tarea de ingeniería # 10

Tarea	
Número de tarea: 10	**Número de Historia de usuario:** 5
Nombre de la tarea: Visualizar Variable meteorológicas.	
Tipo de tarea: desarrollo	**Puntos estimados:** 1
Fecha de inicio: 19 de abril de 2024	**Fecha de fin:** 22 de abril de 2024
Programador responsable: Daiamarilis Guillot Reyes	
Descripción: Esta funcionalidad permitirá al usuario visualizar las variables meteorológicas en tiempo real.	

- HU_5 Visualizar Variable meteorológicas.
- HU_5.

Apéndice4 Calidad del producto de software

Figura 4. Calidad del producto de software (Fuente: Elaboración propia).

Apéndice5 Resultados de las pruebas de Aceptación

Figura 5. Pruebas de Aceptación (Fuente: Elaboración propia).

ApéndicePruebas de aceptación

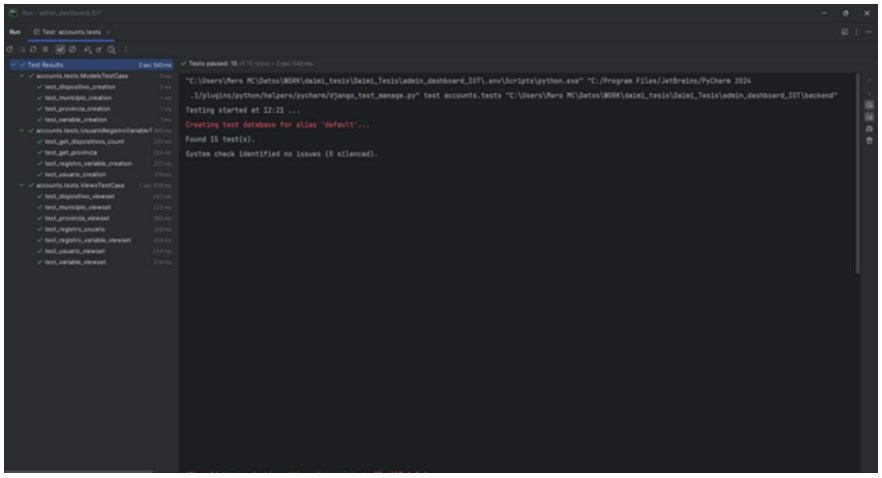


Figura 6. Pruebas de Aceptación (Fuente: Elaboración propia).