Universidad Autónoma de Ciudad Juárez

Instituto de Ingeniería y Tecnología
Departamento de Eléctrica y Computación
Maestría en Cómputo Aplicado

“Framework para manejo de datos heterogéneos climáticos”

Tesis para obtener el grado de
Maestría en Cómputo Aplicado

Alicia Margarita Jiménez Galina

“Becada por el Consejo Nacional de Ciencia y Tecnología”

Bajo la Dirección del
Maestro Fernando Estrada Saldaña
Y la Codirección del
Dr. Jesús Israel Hernández Hernández

Ciudad Juárez, Chihuahua, marzo del 2020

Agradecimientos

A Dios, por permitirme vivir y disfrutar de este momento. A mi amada familia, mi esposo e hijo, por su amor incondicional, apoyo y comprensión en todo momento y por creer siempre en mi y mis capacidades.

A mis padres, hermanos, cuñados y sobrinos, por su motivación constante y su gran amor.

A mi director por su confianza depositada en mi para este proyecto, sus consejos, su guía y dedicación, sin la que no hubiera sido posible lograr culminar exitosamente el desarrollo de esta tesis.

A mi co-director, revisores y lectores, por su constante apoyo, transmisión de sus conocimientos, comentarios y observaciones que permitieron mejorar el presente documento.

A mis maestros, por su paciencia, enseñanzas, consejos y apoyo.

A mis compañeros y amigos, por compartir conocimiento, apoyo y motivación durante este trayecto.

A Consejo Nacional de Ciencia y Tecnología, por su apoyo económico durante la realización de este proyecto.

El camino que he recorrido para lograr este proyecto no ha sido nada sencillo, pero nunca es tarde para cumplir sueño, con dedicación, esfuerzo y apoyo de todas las personas que han estado a mi lado en esta travesía he logrado que este sueño se haga realidad. Mil gracias por estar en mi vida.

Índice de Contenidos.

1. Planteamiento 9
1.1. Antecedentes 9
1.2. Definición del problema 14
1.3. Objetivos 14
1.3.1. Objetivo general 14
1.3.2. Objetivos específicos 14
1.4. Justificación 14
2. Marco Referencial 16
2.1. Marco Teórico 16
2.1.1. Calidad del aire 16
2.1.2. Meteorología 16
2.1.3. Climatología 16
2.1.4. NOAA (National Oceanic and Atmospheric Administration) 16
2.1.5. Servicio Meteorológico Nacional 16
2.1.6. Centro de Ciencias Atmosféricas y Tecnologías Verdes (CECATEV) 17
2.1.7. Sistema Internacional de Unidades 17
2.1.8. Estación meteorológica 17
2.1.8.1. Tipos de estaciones meteorológicas 17
2.1.8.2. Estación meteorológica Campbell Scientific 18
2.1.8.3. Estación Davis Vantage Pro2 19
2.1.8.4. *Datalogger* 19
2.1.8.5. Sensor 19
2.1.9. Variables climáticas 20
2.1.9.1. Presión atmosférica 21
2.1.9.2. Temperatura 22
2.1.9.3. Humedad del aire 22
2.1.9.4. Tasa de precipitación 23
2.1.9.5. Viento 23
2.1.9.6. Radiación solar 24
2.1.9.7. Radiación ultravioleta 24
2.1.10. Estación de monitoreo de calidad del aire 25
2.1.10.1. Teledyne API 25
2.1.11. Variables de contaminantes 26

2.1.11.1. Dióxido de azufre (SO_2) 26
2.1.11.2. Monóxido de carbono (CO) 26
2.1.11.3. Dióxido de carbono (CO_2) 27
2.1.11.4. Dióxido de nitrógeno (NO_2) 27
2.1.11.5. Ozono 27
2.1.11.6. Partícula atmosférica 27
2.1.11.7. Conversiones 28
2.2. Marco Tecnológico 29
2.2.1. Sistema de *big data* 29
2.2.2. Data warehouse 29
2.2.3. Análisis y visualización de datos 29
2.2.4. Extracción de datos 30
2.2.5. Metadatos 30
2.2.6. XML 30
2.2.7. JSON 30
2.2.8. Python 31
2.2.9. Django 31
2.2.10. Node.js 31
2.2.11. Express.js 31
2.2.12. Flask 31
2.2.13. Weewx 31
2.3. Marco Conceptual 32
3. Metodología 35
3.1. Minería de datos 35
3.2. Análisis 35
3.3. Estructura de metadatos 36
3.4. Estructura JSON de SQL a MongoDB 40
3.5. Desarrollo 41
3.6. Funcionalidad 52
3.7. Homogeneización 53
3.8. Pruebas 54
4. Resultados 56
5. Conclusiones 65
Índice de Figuras. 67

Índice de Tablas. 69
Referencias 70
Anexo I – Tablas software *weewx*. 74
Anexo II – Metadatos archivo XML. 76
Anexo III – Librerías del *framework*. 100
homoframe.py 100
conversor.py 108
engine.py 111
frame_clima.py 113
views.py 117

Introducción

La Universidad Autónoma de Ciudad Juárez (UACJ), a través del Centro de Estudios Atmosféricos y Tecnologías Verdes (CECATEV), ha recolectado información desde 1996, sin embargo, utiliza procesos manuales para crear los archivos que deben servir al acervo de datos climatológicos de nuestra región lo cual limita la calidad y cantidad de información que se produce. Además, debido a los compromisos que cuenta la UACJ por medio del CECATEV con diversas universidades dentro y fuera del país, se debe compartir la información climática a diferentes usuarios, entre universidades y expertos en la materia.

Los datos atmosféricos son heterogéneos debido a que son recolectados por estaciones climatológicas de diferentes tipos, marcas y modelos, así como analizadores de gases y aerosoles, lo que implica un reto en la visualización de los datos de una manera que facilite el análisis y la elaboración de productos relevantes para la toma de decisiones y la mejor comprensión de los procesos radiativos y contaminantes de nuestra región.

Si se considera además del tiempo que se emplea en concentrar la información manualmente, que los datos incluidos no están homogeneizados y lo que implicaría un procedimiento más complejo y tardado querer combinar diferentes estaciones y sensores en un solo archivo, resulta indispensable disponer de una herramienta que permita el tratamiento de los datos, minimizando tiempos y homogenizándolos para que puedan ser asimilados en modelos predictivos de la atmósfera.

El presente reporte de desarrollo tecnológico construye la administración de datos heterogéneos, iniciando por los procesos creados para su almacenamiento, así como el tratamiento para extracción y homogeneización de estos. Para este caso se utilizaron datos climáticos provenientes de estaciones meteorológicas y de calidad del aire, las cuales recolectan información de diferentes sensores y en diferentes unidades de medida. Para lograr lo anterior se construye un *framework* con estas características.

Se crea una estructura de metadatos que contienen el perfil de las estaciones y sus sensores, también se utiliza una base de datos NoSQL que permite la gestión de los datos en una estructura dinámica. Utilizando los metadatos y de acuerdo con la elección del usuario, permite combinar estaciones y sensores, rango de fechas y la unidad de salida deseada. Para finalmente a través de los parámetros elegidos extraer y homogeneizar la información, generando un *dataset* que pueda utilizarse para análisis de datos.

En el primer capítulo se describe el dominio de la aplicación analizado para decidir las estrategias a utilizar. Se establece la problemática, se define el objetivo general, los objetivos específicos, así como la justificación. En el segundo capítulo se refiere al marco referencial compuesto por el marco teórico, marco tecnológico y marco conceptual. En el tercer capítulo se

describe las estrategias desarrolladas para la construcción del *framework*, la creación de los metadatos, la estructura dinámica para el almacenamiento de los datos y el conjunto de librerías que forman el *framework*. En el cuarto capítulo se evalúan los resultados del *framework* en comparación con el ambiente manual. En el último capítulo, el quinto se presentan las conclusiones de los resultados del *framework*, así como áreas de oportunidad detectadas para mejoras futuras.

Palabras clave: Datos heterogéneos, homogeneización, NoSQL, MongoDB, ETL

1. Planteamiento

1.1. Antecedentes

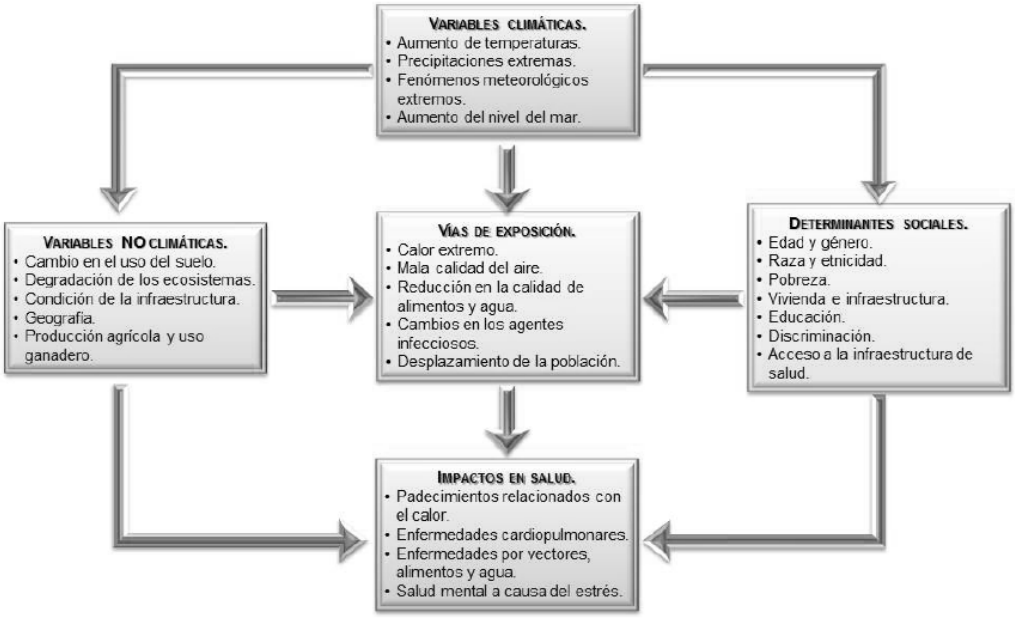
En el año 1996, de acuerdo con [1], la Universidad Autónoma de Ciudad Juárez (UACJ), fundó el Laboratorio de Meteorología, formando parte del Departamento de Ingeniería Civil y Ambiental del IIT de la UACJ. En el año 2007, se moderniza el laboratorio, ahora conocido como Centro de Estudios Atmosféricos y Tecnologías Verdes (CECATEV). Actualmente el CECATEV se encuentra habilitando una red de estaciones climatológicas (Red-Clima UACJ), con más de10 estaciones que permitirá generar datos climatológicos e hidrológicos de las cuencas hidrológicas de Ciudad Juárez y los municipios de Práxedis G Guerrero, Casas Grandes, Ascensión y Cuauhtémoc en el estado de Chihuahua.

El CECATEV ha sido designado como laboratorio de referencia científica del programa de calidad del aire de la cuenca atmosférica de Ciudad Juárez y como parte de un convenio de colaboración entre la UACJ y el Instituto Nacional de Ecología y Cambio Climático. El CECATEV en [1], refiere que dentro de sus funciones se encuentra el mantenimiento a la red climatológica y estudio de la contaminación atmosférica mediante la generación de bases de datos, en las cuales se concentran variables climáticas de la región como temperatura, dirección y velocidad del viento, humedad relativa, evaporación, precipitación pluvial y radiación solar en todos sus componentes.

El nuevo paradigma para el estudio del cambio climático se basa en no compartamentalizar los sistemas ambientales de tal forma que la evaluación del impacto se integre por ciclos, como son el ciclo del carbono, del agua, del azufre etc. Bajo este nuevo paradigma los impactos a la salud asociados a la mala calidad del aire se estudian como parte de los mismos procesos que impactan sobre el cambio climático. Esto implica que las escalas de los procesos han dejado de tener el peso que en antaño se les asignaba. La salud por ejemplo a partir de diversos estudios que muestran diferentes enfermedades relacionadas con las exacerbaciones inducidas por el cambio climático, tales como: enfermedades infecciosas asociadas al transporte y la generación de energía y que además de generar forzantes climáticos generan contaminantes atmosféricos asociados a los procesos inflamatorios cardiovasculares, aumento de la morbilidad y la mortalidad asociados el cambio de las fronteras que delimitan la presencia de vectores de enfermedades tropicales así como la enfermedad de Lyme en Canadá, el Vibrio en el norte de Europa, así como desnutrición y el colapso de la seguridad alimentaria en varias regiones del mundo. Por lo que la IPCC (The Intergovernmental Panel on Climate Change), en [2], concluye que el cambio climático generará importantes riesgos de enfermedades, lesiones, desnutrición y muertes, debido al incremento de las olas de calor e incremento de incendios más severos. Tomando en cuenta el [3], informe de la Comisión Estatal para la Protección Contra Riesgos Sanitarios de Chihuahua, al respecto de los cambios climáticos en la

región, "El calentamiento del planeta será gradual, pero la creciente frecuencia e intensidad de los fenómenos meteorológicos extremos, como tormentas torrenciales, olas de calor, sequías e inundaciones, se manifestarán de manera abrupta y las consecuencias se percibirán de forma aguda", por lo anterior se puede inferir que es necesaria más información climática, que permita apoyar a la población en la toma de decisiones por ejemplo, cual producto se debe sembrar y en que lugar sembrarlo en base a la información existente, por mencionar algunos. En la Figura **1** se muestra el diagrama de los factores que influyen en el impacto en la salud.

Figura 1. **Diagrama de las diferentes variables, rutas de exposición y determinantes sociales que influyen en los impactos en la salud**



Fuente: "Impactos del cambio climático en la salud" [3]

Como se mencionó anteriormente, a partir de 1996 la UACJ por medio del CECATEV (antes del CECATEV a cargo del Laboratorio de Climatología y Calidad del Aire) ha contribuido con la recolección de información climática de manera importante, en la actualidad dicha información se ha incorporado paulatinamente al repositorio institucional. Esta información resulta útil para generar pronósticos, alertas, decisiones, entre otras actividades que benefician a todo el estado, como se describe en [1]. Para maximizar los beneficios otorgados, la UACJ ha trabajado en proyectos para incrementar las estaciones meteorológicas instaladas en la localidad, así como la creación de sistemas de *big data*.

Sin embargo, aunque se cuenta con grandes volúmenes de información en el CECATEV, aún no existe una herramienta que permita la recuperación de la información heterogénea para compartirla con los expertos, entre universidades para el análisis correspondiente o solicitantes diversos.

En base a lo expuesto es necesario crear una base de datos confiable, escalable, que pueda contener datos heterogéneos y que pueda ser consultada de una manera sencilla. Para lograr esto primero se deberá concentrar los datos existentes y al mismo tiempo la información que se vaya generando se integre en tiempo real a la base de datos generada.

Los procesos de extracción, transformación y carga (*Extract-Transform-Load ETL*), se utilizan generalmente cuando se refieren al movimiento y transformación de datos. Estos procesos permiten mover los datos de diferentes fuentes, transformarlos o reformatearlos y cargarlos en otra base de datos llamada *data warehouse* para poder ser analizados. Por lo que se propone utilizar un sistema ETL.

Originalmente *big data* fue asociado a los conceptos de volumen, variedad y velocidad, sin embargo, *big data* incluye la captura, almacenamiento, análisis, búsqueda, intercambio, transferencia, visualización, consulta, actualización, privacidad y fuente de los datos, así como la veracidad y su valor. Actualmente este término rara vez se refiere al tamaño del conjunto de datos, sino al análisis predictivo, de comportamiento del usuario o análisis de datos que extraen valor de los datos.

Por lo anterior se entiende que el proyecto es de *big data* principalmente por el origen heterogéneo de los datos, su crecimiento exponencial, así como la necesaria veracidad y valor de estos, siendo necesario aplicar herramientas que nos permitan acceder a los datos de una manera eficiente, obteniendo la información en los términos deseados.

El enfoque del presente documento incluye exclusivamente la recuperación de datos heterogéneos de una manera sencilla, en el contexto *big data* con el uso de procesos ETL. Esta problemática ha sido abordada por diversos autores, con varios enfoques como se menciona a continuación.

Papa Senghane Diouf, Aliou Boly, Samba Ndiaye, miembros de la universidad de Cheikh Anta Diop ubicada en Dakar, Senegal, en el artículo [4], realizan una revisión de distintas soluciones de ETL para aplicar en el contexto de *big data* en la nube. Uno de los enfoques sugiere soluciones basadas en programación o en la paralelización de tareas. El segundo enfoque se basa en una nueva arquitectura para eliminar la "zona de amortiguamiento", con lo que se puede reducir hasta en un 50% el espacio utilizado en disco, además de reducir el tiempo del procesamiento de los datos. Por lo que sugieren estudiar más profundamente aspectos como elasticidad, dinamismo y el costo de los recursos para lograr mejores implementaciones de ETL.

Otro artículo el [5], de los mismos autores, explican el proceso ETL para la variedad de datos en la nube, en el cual se basan en las características principales del *big data* (velocidad, variedad y volumen), en el tema de la variedad de los datos se encuentran soluciones que utilizan tecnologías

semánticas, sin embargo, sigue pendiente los temas de elasticidad y dinamismo, los cuales apoyarían en el problema del costo del modelo de la nube "pago por uso".

Lunan Li, de la Universidad Donghua, resalta en su artículo [6], la importancia del ETL en la construcción de DW (*Data Warehousing*). La base de los procesos ETL es la administración de metadatos, al contar con un buen diseño de metadatos se mejora la eficiencia del ETL. El documento propone administrar el ETL mediante un repositorio de metadatos. Primero diseñan un *framework* ETL, luego un modelo de metadatos (para los metadatos desordenados y no sistemáticos), con lo que componen el repositorio de metadatos, para el final administran los metadatos por medio del repositorio optimizando ETL.

La idea presentada en los artículos [4] [5] [6] anteriores sobre el proceso ETL fue utilizada en este proyecto ya que se trabajó en el CECATEV por medio de la implementación eficiente de este proceso. Adicionalmente se manejaron las relaciones de conceptos, tomando en cuenta los metadatos para este propósito, de tal manera se conoce al momento de extraer la información que tipo de conversiones se deben realizar y así proporcionar la consulta solicitada por el usuario.

Zhong, de *Wuhan University* en China y Lui de *Carleton University* en Canadá, proponen en su artículo [7] un Semi-Structured Search Engine (3SE) con un lenguaje de consulta (3SQL), por medio del cual el usuario puede redactar consultas semiestructuradas de datos heterogéneos, para este caso de diferentes fuentes de información, simplificando las consultas al usuario.

Jaybal, Ramanathan y Rajagopalan, miembros del International Institute of Information, en su artículo [8], presentan un prototipo de un *framework* llamado HDSAnalytics, el cual utiliza datos de diferentes fuentes y formatos. El *framework* que proponen, proporciona a los modelos de análisis los datos heterogéneos tal como están, sin ningún tipo de integración. Para este *framework* se utiliza el dominio de transporte público, se genera una tabla de las fuentes de datos relacionada con conceptos claves, a través de esta tabla se puede identificar de donde se van a extraer los datos solicitados por el usuario, de tal manera que el trabajo de localizar los datos heterogéneos es transparente para el usuario, una vez que se tiene el resultado de la consulta se usa un modelo de análisis para localizar la ruta más corta o se aplica el modelo deseado.

Dao y Xettsu, del National Institute of Information and Communications Technology, en el artículo [9], presentan un *framework* a través del cual detecta eventos médicos correlacionando sensores físicos (temperatura presión del aire, viento y lluvia), partículas en suspensión y un sensor social. Además de utilizar información histórica para generar las tendencias.

Un grupo de científicos de diversas empresas y universidades en el artículo [10], proponen resolver conflictos entre múltiples fuentes de tipos de datos heterogéneos. Utilizan un *framework,* el cual a través de diversos algoritmos verifican la confiabilidad de las fuentes de datos. Siendo su

objetivo minimizar el riesgo a las empresas al utilizar fuentes falsas y por ende basarse en datos erróneos para sus análisis, y consecuentemente pérdidas financieras.

En base a los artículos que se mencionaron anteriormente, correspondientes al *framework HDSAnalytics* en [8] y el *framework* de eventos médicos con sensores físicos en [9], en los cuales utilizan *frameworks* para el desarrollo de sus proyectos, se concluyó que para el desarrollo del proyecto descrito en este documento se puede crear un *framework* para homogeneizar la información, al incluir diferentes librerías con tareas específicas que ayuden a este propósito. Para el caso del artículo [7], correspondiente al motor de consultas semiestructuradas optimiza las consultas en SQL de acuerdo con su modelo, por lo que se puede considerar en el proyecto del presente documento que se debe tener precaución de revisar los mecanismos que ofrece una base de datos NoSQL para los procesos de recuperación de los datos y como optimizar estos procesos. Sin embargo, en el artículo [10], correspondiente al *framework* que resuelve conflictos entre múltiples fuentes de tipos de datos heterogéneos no se puede considerar directamente para este proyecto debido a que las fuentes, aún que contienen datos heterogéneos, provienen de fuentes plenamente identificadas y por lo tanto confiables, se pudiera considerar para trabajos futuros para expandir el alcance del proyecto.

El artículo [11], correspondiente a los diferentes estándares de consultas y manejo de los metadatos en bases de datos, se relaciona con el proyecto de este documento ya que se pretende utilizar los metadatos para agilizar las consultas e integración de los datos, además de utilizar los mismos para la estandarización de los resultados.

En el artículo [12] en el cual se administran datos espaciales en MongoDB, analizando estrategias de operaciones CRUD basadas en R-tree, se resuelve creando una estructura de tipo árbol que permite gestionar los datos eficazmente.

Miembros de la Tongji University en Shanghai, China en su artículo [13], presentan un método de visualización llamado Graph-Tree, el cual describe las relaciones entre conjuntos de datos en bases de datos. Este método propuesto supone que es más fácil interpretar los datos mostrados en gráficas o imágenes. La herramienta RMine se basa en este método. La estructura visual de RMine es definida por el usuario, en donde agregan un cambio dinámico relacional y mejoran la eficiencia de la consulta de los datos. Los datos heterogéneos que se usan para este método provienen de diferentes fuentes. Sin embargo, para utilizar esta herramienta es necesario que se realicen adecuaciones manuales de los datos para poder representarlos gráficamente. El éxito de esta herramienta radica en hacer más eficiente la extracción de datos en comparación con consultas en SQL.

Debido a que el proyecto de este documento maneja base de datos no relacionales se tomaron en cuenta los artículos referentes al uso de metadatos en [11] y operaciones en MongoDB en [12],

para aplicarlos en el uso eficiente de la información almacenada en la base de datos, debido a que maneja datos climáticos, bases de datos no relacionales y metadatos. Además, para cuidar los aspectos de la integridad de la información se toma en cuenta artículo [13] donde realiza adecuaciones que ayudan en la extracción de los datos, de tal manera que a través de un *framework* en conjunto con los metadatos, permita seleccionar la información que se va a entregar.

De acuerdo con la información investigada se concluye que se debe implementar el proceso ETL eficientemente cuidando la integridad de los datos y crear un *framework* con diversas librerías que permitan homogeneizar la información y generar un *dataset* que pueda ser utilizado por los expertos en diferentes modelos para su análisis correspondiente.

1.2. Definición del problema

El CECATEV para poder compartir la información climática recolectada con las instituciones de las cuales forma parte y diferentes usuarios, actualmente debe realizar procedimientos manuales para reunir los datos que se encuentran en diferentes formatos y unidades. Por lo que es necesario contar con una herramienta propia que pueda extraer la información y generar *datasets* con datos homogéneos de acuerdo con los requerimientos de los solicitantes.

1.3. Objetivos

1.3.1. Objetivo general

Implementar un *framework* para la extracción de datos heterogéneos de múltiples fuentes, existentes en el CECATEV mediante la homologación de estos en *datasets* utilizables por diferentes tipos de usuarios.

1.3.2. Objetivos específicos

Desarrollar un prototipo de interfaz para probar el *framework*, mediante el desarrollo de diferentes librerías que homogenicen la información del *data warehose*, de acuerdo con los diferentes indicadores solicitados, así como el desarrollo de las librerías correspondientes para la generación del *dataset*.

1.4. Justificación

El clima se refiere a la estadística del tiempo atmosférico en un periodo de 30 años. Su medición incluye variables climáticas como temperatura, humedad, presión atmosférica, viento, precipitación, cuenta de partícula atmosférica y otras, en una región sobre un periodo largo de tiempo.

El CECATEV de la UACJ, cuenta con información a partir de septiembre del año 2007, aunque no se cuenta con los 30 años de datos para el modelaje de la normal climática por mencionar alguno, sin embargo, existen otros modelos que pueden utilizar desde 5 años para su análisis. Este tipo de modelos utilizan rangos de información dependiendo de la proyectividad y normatividad.

El *framework* de este proyecto contribuirá de manera importante, al disminuir los tiempos considerablemente de la extracción de datos y el análisis de estos en el CECATEV de la UACJ, de igual manera se podrá compartir a la comunidad científica, universidades, etc., la información almacenada mediante procedimientos sencillos y fiables, para apoyar en el análisis y predicciones climáticas. Esta herramienta podrá ser utilizada en conjunto con otras utilerías no contempladas en este proyecto para realizar el almacenamiento automático al repositorio y así permitir extraer los datos climáticos.

2. Marco Referencial

2.1. Marco Teórico

El presente capítulo contiene los conceptos relacionados con el dominio del problema.

2.1.1. Calidad del aire

La calidad del aire se refiere a la concentración de contaminantes en el aire. El índice de la calidad del aire es una cifra que refleja la cantidad de esos contaminantes en el aire. Cuando la calidad del aire es mala, es decir contiene más contaminantes, entonces es nocivo para la salud y puede producir afectaciones respiratorias, irritaciones oculares y problemas cardiacos, por lo que las autoridades pueden tomar ciertas medidas para intentar disminuir la emisión de contaminantes y ciertas prohibiciones de actividades al aire libre, lo anterior de acuerdo con [14].

2.1.2. Meteorología

La meteorología, en [15], es la ciencia que estudia los fenómenos y propiedades atmosféricas en un momento especifico y lugar determinado.

2.1.3. Climatología

Conforme a [15], la climatología es el estudio del clima y sus variaciones en el tiempo. El clima se refiere a la estadística del tiempo atmosférico en un periodo de 30 años. Su medición incluye variables climáticas como temperatura, humedad, presión atmosférica, viento, precipitación, cuenta de partícula atmosférica y otras, en una región sobre un periodo largo de tiempo.

Se debe distinguir entre el clima y el tiempo atmosférico, mientras el tiempo describe las condiciones actuales de las variables meteorológicas en una región, el clima está generado por el sistema climático (atmósfera, hidrósfera, criósfera, litósfera y biósfera), es decir se refiere a condiciones medias en periodos más largos. Por lo que los datos atmosféricos son obtenidos de estaciones meteorológicas y son usados para definir el clima, lo anterior descrito en [16].

2.1.4. NOAA (National Oceanic and Atmospheric Administration)

El NOAA, de acuerdo con [17], es la agencia científica de Estados Unidos que pertenece al Departamento de Comercio, es la encargada de emitir alertas meteorológicas peligrosas, preparar cartas de mares y de cielos, guías sobre el uso y la protección de los recursos oceánicos y costeros, y conducir los estudios para entender y gestionar el ambiente.

2.1.5. Servicio Meteorológico Nacional

El Servicio Meteorológico Nacional (SMN) es el organismo dependiente de la Comisión Nacional del Agua (CONAGUA) que a su vez depende de Secretaría de Medio Ambiente y Recursos Naturales (SEMARNAT), que proporciona información sobre el estado del tiempo nacional y local. El SMN tiene como objetivo identificar los fenómenos meteorológicos que puedan afectar a la población o a alguna actividad económica del país.

También, como se describe en [18], es el encargado de revisar, depurar y ordenar la información climatológica nacional en el Banco Nacional de Datos Climatológicos, que se encuentra disponible al público. Dentro de las funciones de este organismo se encuentra la difusión sobre las condiciones meteorológicas y climáticas al Sistema Nacional de Protección Civil y al público, así como realizar estudios climatológicos o meteorológicos.

2.1.6. Centro de Ciencias Atmosféricas y Tecnologías Verdes (CECATEV)

La Universidad Autónoma de Ciudad Juárez (UACJ) desde agosto del 2016 forma parte de la Aerosol *Robotic Network* (Aeronet), a través del Centro de Estudios Atmosféricos y Tecnologías Verdes (CECATEV) de la UACJ, la Aeronet, como se refiere en [17], es una red meteorológica de la Administración Nacional de la Aeronáutica y el Espacio (NASA). Por lo que la UACJ enviará a esa agencia los datos recolectados sobre las condiciones atmosféricas de esta región.

2.1.7. Sistema Internacional de Unidades

En busca de estandarizar las unidades de medida en el mundo, en el año 1875 se creó la Oficina Internacional de Pesos y Medidas (Bureau International Des Poids Et Mesures - BIPM, por sus siglas en francés) por el Comité Internacional de Pesos y Medidas (CIPM), que a su vez está bajo la autoridad de la Conferencia General de Pesos y Medidas (CGPM). Siendo el 30 de diciembre de 1980 cuando México se une a este tratado, conforme a [19]. La función del BIPM es asegurar la unificación mundial de las mediciones, de acuerdo con [20].

En México las unidades de medida de uso legal y obligatorio se establecen en la norma NOM-008-SCFI-2002 Sistema General de Unidades de Medida (última reforma publicada DOF 30-04-2009), descrito en [21]. Este sistema se integra con las unidades del Sistema Internacional de Unidades (SI), que es la modernización del sistema métrico.

Las unidades base del SI, como se indica en [22], son las conocidas como el metro, el kilogramo, el segundo, el ampere, el kelvin, el mol y la candela, a partir de estas se establecen las unidades derivadas.

2.1.8. Estación meteorológica

Las estaciones meteorológicas, de acuerdo con [23], están formadas por instrumentos meteorológicos (sensores), los cuales son semejantes a la piel humana, en todo momento están detectando la variable climática, por ejemplo, el higrómetro mide la humedad del aire, entonces a través del *datalogger*, que es el que le indica el intervalo de tiempo en que se debe almacenar esa lectura, se van almacenando los datos atmosféricos adquiridos.

2.1.8.1. Tipos de estaciones meteorológicas

Una estación meteorológica es una instalación que mide y registra constantemente variables meteorológicas.

Esta información recabada puede ser utilizada tanto para elaborar estudios climáticos como para predicciones meteorológicas.

De acuerdo con la Organización Meteorológica Mundial (OMM), las estaciones meteorológicas se clasifican de la siguiente manera:

- De acuerdo con sus finalidades sinópticas: climatológicas, agrícolas, aeronáuticas y especiales.
- De acuerdo con la magnitud de las observaciones: principales, ordinarias y auxiliares.
- De acuerdo con el nivel de observación: superficie y altitud.
- Y por el lugar de observación: terrestre, aéreas y marítimas

De igual manera se pueden clasificar en convencionales, cuando requieren de un operador para recopilar y transmitir la información y las automáticas, las cuales registran los datos y los transmiten de manera automatizada.

Dentro de las estaciones climatológicas se tiene la Estación Meteorológica Automática (EMA), que contiene sensores que miden los parámetros meteorológicos, y estos datos son almacenados y procesados en su *datalogger*. Una EMA contiene principalmente sensores, un sistema central de procesamiento (*datalogger*) y equipo periférico, lo anterior se refiere en [24].

2.1.8.2. Estación meteorológica Campbell Scientific

Una estación meteorológica Campbell Scientific se puede utilizar para investigación climática como meteorología y calidad del aire. Estas estaciones meteorológicas se basan en un *datalogger* programable, por medio del cual toma las lecturas de los sensores y son almacenados esos registros. En el *datalogger* se pueden configurar las unidades de medida en el que se almacenan los datos, así como los intervalos en que se procesan las muestras, de acuerdo con [25]. En la Figura **2** se muestra esta estación.

Figura 2. **Estación meteorológica Campbell Scientific**

Fuente: "Campbell Scientific" [25]

2.1.8.3. Estación Davis Vantage Pro2

La estación Davis, conforme a [26], es una estación meteorológica que a través de una consola registra automáticamente los datos atmosféricos medidos por los sensores y son almacenados en un *datalogger*, para posteriormente usando un software *Weatherlink* y un PC, puede acceder y/o descargar la información, o enviarla de manera remota. En la Figura **3** se muestra una estación Davis con su *datalogger*.

Figura 3. **Estación Davis con *datalogger***

Fuente: "Estación Davis" [26]

2.1.8.4. *Datalogger*

El *datalogger*, de acuerdo con [27], es un instrumento por medio del cual se puede registrar y guardar datos o información en un lapso determinado. Los datos pueden ser variables meteorológicas como temperatura, humedad, presión, viento, etc. Los componentes principales de un *datalogger* son el procesador, la memoria externa y/o interna, reloj de tiempo real y el sensor. En la Figura **4** se muestra un *datalogger* de la marca *Campbell Scientific*.

Figura 4. ***Datalogger***

Fuente: Campbell Scientific [27]

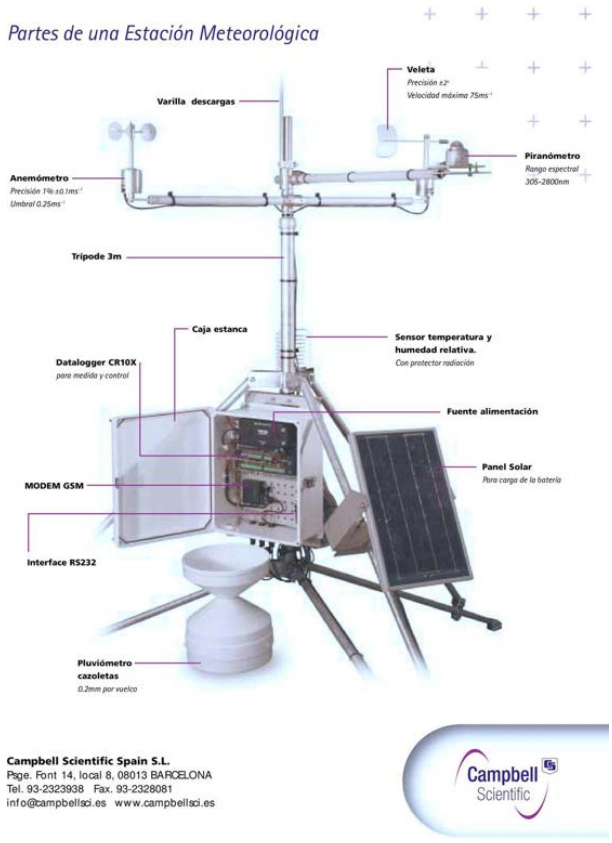
2.1.8.5. Sensor

Los sensores son dispositivos que se encuentran en las estaciones meteorológicas, por medio de los cuales se miden y registran las variables meteorológicas, estos datos se registran en el

datalogger. Existen sensores activos como por ejemplo una celda solar y sensores pasivos como una resistencia fotosensible LDR, lo anterior descrito en [28]. En la Figura 5 se muestran algunos sensores contenidos en las estaciones meteorológicas y pueden ser:

- Pluviómetro para medir la precipitación (lluvia)
- Termómetro para medir la temperatura
- Anemómetro para medir la velocidad del viento
- Piranómetro para medir la radiación solar
- Barógrafo para medir presión atmosférica
- Higrógrafo para medir humedad
- Sensor UV

Figura **5**. **Sensores en una Estación Meteorológica**



Fuente: Campbell Scientific [29]

2.1.9. Variables climáticas

Una variable climática es el dato atmosférico medido o registrado por un instrumento meteorológico. El estado del tiempo se define por el conjunto de variables climáticas en un corto

plazo, mientras que el clima es definido por el mismo conjunto de variables, pero en periodos más largos.

Las variables climáticas pueden ser medidas o registradas a través de los instrumentos meteorológicos de las estaciones meteorológicas, como se indica en [30], y algunas de estas variables meteorológicas son:

- Presión atmosférica, es el peso del aire sobre la tierra.
- Temperatura, es el grado de calentamiento debido a la radiación solar.
- Humedad, es el vapor de agua que tiene el aire.
- Precipitación, es el agua que proviene de la atmósfera.
- Viento, es la circulación del aire, se mide la velocidad y dirección.

2.1.9.1. Presión atmosférica

La presión atmosférica es la presión que ejerce el aire sobre algún punto de la capa gaseosa que se encuentra rodeando el planeta (atmósfera). Para calcular correctamente la presión atmosférica se debe tomar en cuenta la densidad del aire, la cual disminuye o aumenta con relación al disminuir o incrementar la altitud. Por lo que entre más alto está un punto sobre el nivel del mar menos capa de aire tiene por encima. La unidad de medida es la atmósfera, y es definida como la "cantidad de peso que ejerce una columna de mercurio de 760 milímetros de altura a una latitud de 45º, al nivel del mar y a una temperatura de 0º centígrados", en meteorología se usan los milibares o los milímetros de mercurio y su relación es la siguiente: 1 atmósfera son 1.013,2 milibares o 760 milímetros de mercurio. Lo descrito anteriormente en base a [31]. En la Tabla **1** se muestran los factores de conversión para la presión atmosférica.

De acuerdo con el SI, en [20], la unidad base de la presión es el Pascal, para este proyecto se utilizan los milibares (mb), ya que es la unidad con la que la mayoría de los sensores recolectan las lecturas, y se incluyen sus diferentes factores de conversión a Pa (Pascal), en psi o pound per square inch (libras de fuerza por pulgada cuadrada), atm (atmósfera estándar), pulgadas Hg (pulgada de mercurio), hPa (hectopascal), mm Hg (miligramos de mercurio) y torr denominada así en homenaje a Evangelista Torricelli, conforme a [32].

En la estación meteorológica Davis Vantage Pro2, la presión atmosférica puede mostrarse en unidades de pulgadas de mercurio (in Hg), milímetros de mercurio (mm Hg), milibares (mb) o hectoPascals (hPa). Si se configura en la estación la altitud de la localidad, ésta busca un valor para convertir la presión atmosférica en presión barométrica. Por lo que la presión barométrica es una medición valiosa en previsiones del tiempo, lo anterior de acuerdo con [33] [32].

Tabla 1. **Factores de conversión para la presión.**

Unidades	Mb (milibares)
Pa (Pascal)	**mb = Pa * 0.01**
Atm (atmósfera)	**mb = Atm * 1013.25**
in Hg (pulgadas de mercurio)	**mb = in Hg * 33.86**
hPa (hectopascal)	**mb = hPa * 1**
mm Hg (mm de mercurio)	**mb = mm Hg * 1.33**
Psi (libra por pulgada cuadrada)	**mb = Psi * 68.95**
Torr	**mb = Torr * 1.33**

Fuente: Elaboración propia

2.1.9.2. Temperatura

La temperatura, de acuerdo con su definición, es una propiedad de la materia que mide el grado o nivel de calor de los cuerpos o del ambiente, en el SI su unidad es el kelvin (K). En la estación meteorológica Davis Vantage Pro2, la temperatura se expresa en grados centígrados o grados Fahrenheit, conforme a [33]. En la Tabla **2** se observan los factores de conversión entre las unidades para la temperatura.

Tabla 2. **Factores de conversión de la temperatura**

Unidades	Grado Celsius	Grado Fahrenheit	Rankine
Kelvin	K=K	K=C+273.15	K=(F+459.67)*59
Grado Celsius o centígrado	C=K−273.15	C=C	C = (F - 32) *59
Grado Fahrenheit	F=K*9/5 – 459.67	F = C*9/5 + 32	F=F
Rankine	Ra=K*9/5	Ra = (C + 273.15) *9/5	Ra=F+459.67

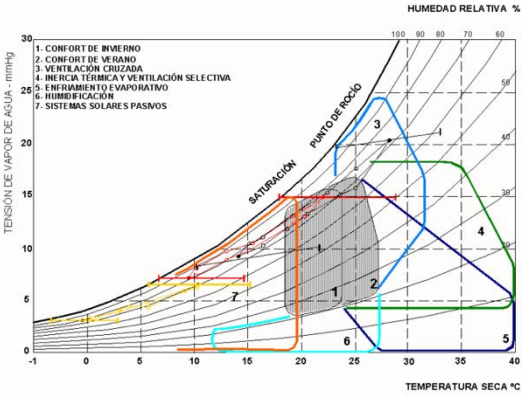
Fuente: Elaboración propia

2.1.9.3. Humedad del aire

La humedad ambiental o atmosférica es la cantidad de vapor de agua que se encuentra presente en el aire, la cual puede ser visualizada como humedad relativa o humedad absoluta y depende de la temperatura y la presión. La humedad relativa expresada como porcentaje; por lo tanto, si es alto el porcentaje la mezcla de aire y agua es más húmeda. Cuando la humedad relativa es 100% significa que el aire se encuentra saturado de agua y se dice que está en su punto de rocío. Las estaciones meteorológicas Davis Vantage Pro2, expresan la humedad relativa ambiente en porcentaje (%), lo anterior se refiere en [33]. Debido a que la humedad se mide en función de la temperatura y presión, no se requieren factores de conversión para su homogeneización. En la Figura **6**, se muestra

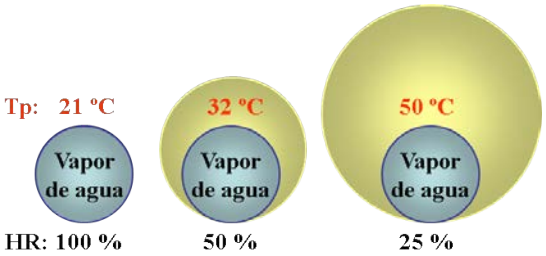
el Climograma de B. Givoni aplicado a los climas húmedos de la Argentina y en la Figura **7** se muestran los cambios en la humedad relativa (HR) de una masa de aire con una humedad absoluta de 20 g / m3 en función de la temperatura (Tp), de acuerdo con [33]. Se indican desde un clima muy cálido a uno muy frío. Del climograma se pueden extraer pautas diseño bioclimático para una arquitectura sustentable, descrito en [32].

Figura 6. **Climograma de B. Givoni**



Fuente: Arquitectura sustentable [34]

Figura 7. **Cambios en la humedad relativa**



Fuente: Cambios en la humedad

2.1.9.4. Tasa de precipitación

La precipitación es el agua sólida o líquida procedente de la atmósfera, y que cae sobre la superficie de la tierra, puede ser lluvia, llovizna, nieve, aguanieve o granizo. En las estaciones meteorológicas Davis Vantage Pro2, se puede medir la intensidad de lluvia en pulgadas por hora (in/hr) o milímetros por hora (mm/hr), lo anterior se refiere en [33], mostrados en la Tabla **3**.

Tabla 3. **Factor de conversión de precipitación**

Unidad	**in/h**
mm/h	0.0394

Fuente: Elaboración propia

2.1.9.5. Viento

De acuerdo con [35], el viento es la corriente o flujo de aire en masa que es producido por diferencias de presión atmosférica. Se puede medir su dirección, que es de donde procede el viento, por medio de una veleta, así como la velocidad del viento con el anemómetro.

La dirección del viento se representa en grados, en dirección de las manecillas del reloj y va de 0º a 360º. Para medir la velocidad se expresa en metros por segundo (m/s), esta es la unidad del viento en el SI, o en kilómetros por hora (km/h), millas por hora (mph) o en nudos (kt), lo anterior conforme a [20]. En la Tabla **4**, se muestran los factores de conversión para la velocidad del viento.

Tabla 4. **Factores de conversión de velocidad del viento**

Unidades	km/h	m/s	mph
1 nudo (kt)	1.852	0.515	1.15

Fuente: Elaboración propia

2.1.9.6. Radiación solar

La radiación solar, con base en [36], son ondas electromagnéticas que son producidas por el Sol. Estas ondas se desplazan desde el Sol a la Tierra a una velocidad de 299.792 km/s.

De acuerdo con el SI, la radiación solar se mide en una unidad de potencia en valores de watts por metro cuadrado (W/m^2), pero si se incluye acumular a esta medida un periodo de tiempo (10 minutos, por ejemplo), entonces la unidad es Joules por metro cuadrado (J/m^2), como se indica en [20]. La radiación solar se visualiza en la estación meteorológica Davis Vantage Pro2 en watts por metro cuadrado (W/m^2), de acuerdo con [33]. Los factores de conversión para la radiación se observan en la Tabla **5**.

Tabla 5. **Factores de conversión de energía de radiación acumulada**

Unidades	$W/h/m^2$	$Btu/pies^2$	$Joules/m^2$
langley	11.622	3.687	41840

Fuente: Elaboración propia

2.1.9.7. Radiación ultravioleta

La radiación ultravioleta, conforme a [36], es la energía electromagnética emitida a longitudes de onda menores que la correspondiente a la visible por el ojo humano, pero mayor que la que caracteriza a los rayos X, cubre el rango espectral desde los 100 a los 400 nm y se divide en:

- Ultravioleta C de 100 a 280 nm. La cual se absorbe completamente por el ozono.
- Ultravioleta B de 280 a 320 nm. Es absorbida en parte por el ozono.
- Ultravioleta A de 320 a 400 nm. Se absorbe muy poco por el ozono.

La intensidad de la radiación ultravioleta es expresada por un indicador llamado índice UV, además el índice UV también representa las lesiones que pudiera producir en la piel la radiación UV. Este índice es estandarizado por las organizaciones siguientes: la Organización Mundial de la Salud, la Organización Meteorológica Mundial, el Programa de las Naciones Unidas para el Medio Ambiente y la Comisión Internacional de Protección contra la radiación no ionizante. Por lo anterior no se identificaron factores de conversión para esta variable, por lo que solo se maneja esta unidad. En la Tabla **6** , se muestra la medición del índice UV, lo anterior en base a [37].

Tabla 6. **Sistema estándar de medición del índice UV incluyendo colores**

Color	Riesgo	Índice UV
Verde	**Bajo**	< 2
Amarillo	**Moderado**	3-5
Naranja	**Alto**	6-7
Rojo	**Muy Alto**	8-10
Violeta	**Extremadamente alto**	> 11

Fuente: Ultraviolet Immunosuppression [37]

2.1.10. Estación de monitoreo de calidad del aire

Una estación de medición o monitoreo de la calidad del aire también es conocida como estación de gases y partículas suspendidas y se encuentra formada por varios sensores encargados de medir las concentraciones de los contaminantes del aire, de acuerdo con [38].

Estas estaciones tienen la capacidad de muestrear las concentraciones de contaminantes, siendo estas las que pueden producir algún daño a la salud, como se describe en [39].

2.1.10.1. Teledyne API

Dentro de los productos Teledyne, existen estaciones especializadas en la calidad del aire y monitoreo de emisiones, los modelos instalados en la localidad y de los que se tienen lecturas registradas son los de modelos para monitoreo de emisiones de las series T100, T200, T300, T400, T700 y el modelo T640 mostrado en la Figura **8** especializado en monitoreo de partículas ambientales, como se refiere en [40]. En la Tabla **7** se muestran las unidades de medida y los rangos de lectura de las estaciones Teledyne correspondientes a las series T, conforme a [40].

Figura **8**. **Teledyne modelo T640**

Fuente: Teledyne API [40]

Tabla 7. **Modelos Teledyne serie T**

Instrumentos de compuestos de azufre				
Gas	**Modelo**	**Rangos (Min/Max)**	**Software Numaview™**	**Aplicaciones**
SO2 Dióxido de azufre	T100	0-50 ppb / 0-20 ppm	Estándar	Nivel ambiental; Dilución CEM
Instrumentos de compuestos de nitrógeno				
Gas	**Modelo**	**Rangos (Min/Max)**	**Software Numaview™**	**Aplicaciones**
NO, NO2, NOX Óxido de nitrógeno, Dióxido de nitrógeno	T200	0-50 ppb / 0-20 ppm	Estándar	Nivel ambiental; Dilución CEM
Instrumentos de compuestos de carbono				
Gas	**Modelo**	**Rangos (Min/Max)**	**Software Numaview™**	**Aplicaciones**
CO Monóxido de carbono	T300	0-1 ppm / 0-1,000 ppm	Estándar	Nivel ambiental
Instrumentos de compuestos de oxígeno				
Gas	**Modelo**	**Rangos (Min/Max)**	**Software Numaview™**	**Aplicaciones**
O3 Ozono	T400	0-100 ppb / 0-10 ppm	Estándar	Nivel ambiental
Calibradores de gas y generadores de aire cero				
Gas	**Modelo**	**Rangos (Min/Max)**	**Software Numaview™**	**Aplicaciones**
Todos los gases	T700	Disponible MFC's: 0-10 cc/min hasta 0-20 LPM	Estándar	Calibrador de gas
Instrumentos de partículas				
Gas		**Modelo**	**Rangos (Min/Max)**	
Monitoreo en tiempo real de concentración de partículas PM		T640	PM2.5, PM10, PM10-2.5	

Fuente: Elaboración propia con información de valores estación Teledyne [40]

2.1.11. Variables de contaminantes

Las estaciones de monitoreo miden contaminantes como, por ejemplo: dióxido de azufre (SO_2), monóxido de carbono (CO), dióxido de nitrógeno (NO_2), ozono (O_3), partículas en suspensión menores a 10 y 2.5 micrómetros (PM_{10} y $PM_{2.5}$, respectivamente).

2.1.11.1. Dióxido de azufre (SO_2)

Es un gas incoloro, pero de olor irritante, su fórmula química es SO_2. Este gas se encuentra en combustibles como el carbón, gas natural, diésel, entre otros, mismo que se libera en procesos de combustión. Es considerado un contaminante ambiental, que si rebasa las concentraciones permisibles puede producir problemas cardiacos y respiratorios. De acuerdo con la NOM-022-SSA1-2010 la concentración de este gas no debe exceder los 288 $\mu g/m^3$ o 0.110 ppm del promedio en 24 horas, lo anterior, descrito en [41]. La unidad de medida que utiliza la estación Teledyne para recolectar el valor de este gas es en partes por billón (ppb).

2.1.11.2. Monóxido de carbono (CO)

El monóxido de carbono es un gas inoloro, altamente tóxico, su fórmula química es CO. Este gas se produce por una deficiencia en los procesos de combustión de compuestos de carbono. Aparatos domésticos con alguna falla y que utilizan sustancias combustibles de este tipo, automotores con el motor encendido, procesos de mala combustión en la industria con estas sustancias, incendios forestales, son algunos ejemplos de procesos que pueden liberar este gas. Por su naturaleza inolora, no es fácil de detectar y puede ser un asesino silencioso. De acuerdo con la NOM-021-SSA1-1993,

establece que la concentración del monóxido de carbono no debe rebasar el valor de 11.00 ppm, que equivale a 12,595 µg/m3 del promedio en ocho horas, de acuerdo con [42]. El valor de este gas en la estación Teledyne, se recolecta en la unidad de medida de partes por millón (ppm).

2.1.11.3. Dióxido de carbono (CO_2)

Su fórmula química es CO_2, el dióxido de carbono es un compuesto químico indispensable para la vida en la Tierra. Cuando respiramos se libera el CO_2 que es usado en la fotosíntesis de las plantas, siendo su función mantener el calor en la atmósfera, para que no se congele la Tierra. Sin embargo, por fenómenos naturales y no naturales en su mayoría, se están produciendo grandes concentraciones atmosféricas de CO_2 por lo que se ha producido el calentamiento global y en consecuencia alteraciones en el cambio climático, como se refiere en [43]. Este gas es medido en la estación Teledyne en partes por millón (ppm).

2.1.11.4. Dióxido de nitrógeno (NO_2)

Este contaminante es liberado en los procesos de combustión, su fórmula química es NO^2, es un gas tóxico e irritante. Se concentra mas en zonas urbanas, ya que se produce en la industria y por los vehículos automotores. El riesgo en la salud por la acumulación de esta contaminante afecta principalmente el sistema respiratorio, como se describe en [44]. Este contaminante es recolectado en la estación Teledyne en partes por billón (ppb).

2.1.11.5. Ozono

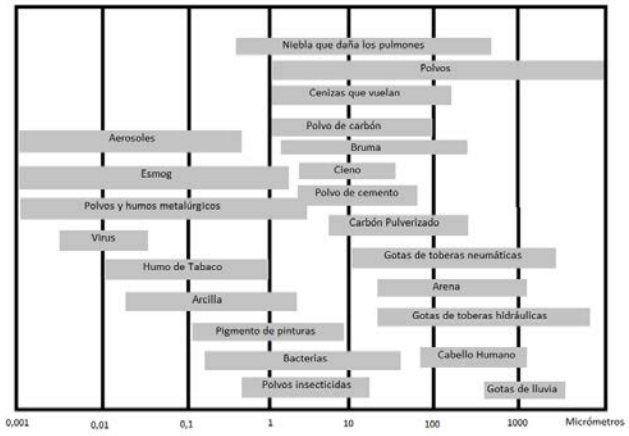
El ozono es una sustancia cuya molécula está compuesta por tres átomos de oxígeno y se forma al disociarse los dos átomos que componen el gas de oxígeno, su fórmula química es O_3. El ozono atmosférico en estado puro en diferentes concentraciones se encuentra entre los 10 y los 40 km sobre el nivel del mar, su concentración más alta se encuentra en la estratosfera. La función del ozono en la atmosfera es para depurar el aire y filtrar los rayos ultravioletas, la llamada “capa de ozono”. Si se respira en cantidades grandes, este contaminante puede provocar irritación en ojos y garganta, de acuerdo con [45].

El ozono es medido en la estación Teledyne en partes por billón (ppb).

2.1.11.6. Partícula atmosférica

Las partículas atmosféricas también se conocen como aerosol atmosférico y son producidas por factores naturales como actividad volcánica, o no naturales como la quema de carbón para producir electricidad. Por lo anterior, la composición de estas partículas es diversa y pueden ser nocivas para la salud y alterar las propiedades de la atmósfera. En la Figura **9** se muestra la distribución de partículas en el aire medidas en micrómetros, lo anterior de acuerdo con [46][47].

Figura 9. **Distribución de partículas en el aire; medidas en micrómetros**



Fuente: Distribución de partículas en el aire [47]

Las partículas en suspensión medidas en la estación Teledyne son medidas en unidades de microgramos entre metro cúbico ($\mu g/m^3$).

2.1.11.7. Conversiones

La estación Teledyne mide las partículas en suspensión en unidades de microgramos por metro cúbico ($\mu g/m^3$), el monóxido de carbono (CO) en partes por millón (ppm), dióxido de nitrógeno (NO_2) en partes por billón (ppb), el ozono (O_3) en partes por billón (ppb) y el dióxido de azufre (SO_2) en partes por billón (ppb) siendo sus factores de conversión los siguientes:

Para hacer conversión de unidades de concentración de partes por millón (ppm) en volumen (ppmv o simplemente ppm en este documento), a microgramos entre metro cúbico ($\mu g/m^3$) se aplicará la siguiente ecuación:

2.1

$$\mu g/m^3 = \frac{ppm\ x\ PM}{24.5}\ x10^3$$

$\mu g/m^3$ = Concentración del contaminante por peso por unidad de volumen de aire.

ppm = Concentración del contaminante por volumen por unidad de volumen de aire.

PM = Peso molecular del agente contaminante conocido.

La ecuación para conversión de unidades se establece para las condiciones de referencia 25°C (298 K) y 760 mm (101.325 kPa) de presión.

Tabla 8. **Tabla de pesos moleculares (PM)**

Gas común	Peso molecular PM (g/mol)
Ozono (O_3)	47.998
dióxido de azufre (SO_2)	64.0638
dióxido de nitrógeno (NO_2)	46.0055
monóxido de carbono (CO)	28.01

Fuente: Elaboración propia

La ecuación de conversión de partes por millón (ppm) a microgramos entre metro cúbico (μg/m^3) es:

$$ppm = \frac{\mu g/m^3}{PM * \left(\frac{10^3}{24.5}\right)} \quad 2.2$$

La equivalencia entre ppm y partes por billón en volumen (ppbv o ppb) es: 1 ppm = 1000 ppb.

2.2. Marco Tecnológico

2.2.1. Sistema de *big data*

Los sistemas de *big data* se definen como un conjunto de datos masivos, heterogéneos, que manejan datos estructurados y no estructurados en el orden de *Terabyte* o *Petabyte*, por lo que dificulta manejarlos de manera convencional con bases de datos relacionales y estadísticas comunes. Por lo que se crearon nuevas tecnologías como Hadoop y NoSQL para la gestión efectiva de la información, lo anterior se describe en [48]. La importancia de los sistemas de *big data* no es solo su capacidad de gestionar grandes cantidades de datos, si no, lo que se puede hacer con esa información. Estos datos pueden ser analizados, en algunos casos combinándolos con bases de datos relacionales, mediante nuevas tecnologías, para poder entender mejor la información almacenada. De esta manera, apoya a las organizaciones a tomar mejores decisiones o estrategias, basados en sus datos, así como implementar nuevas ideas, de acuerdo con [49].

Las características de *big data* generalmente se conocen como modelo de múltiples V. El volumen está relacionado con el "Big" de *big data*, en referencia al tamaño de los datos y el crecimiento y escalabilidad de estos. La variedad, es debida a que *big data* es aplicable a todo tipo de dato, como texto, audio, correos electrónicos, documentos electrónicos, llamadas telefónicas, etc. La velocidad, se refiere a el tiempo de respuesta del análisis de los datos es muy importante y tiene que poderse adaptar a la cantidad de datos ingresados. Veracidad, es qué cantidad de datos correctos dada la confiabilidad de su fuente. Y el valor de los datos que mide la utilidad de los datos al tomar decisiones. La "ciencia analítica" abarca el poder predictivo del *big data*, de conformidad con [50].

2.2.2. Data warehouse

El almacenamiento de los datos ha tenido que evolucionar para responder a la necesidad de guardar los datos masivos. Se debe tomar en cuenta tanto la recolección, almacenamiento y análisis. Por lo que se han empleado nuevas tecnologías para el almacenamiento, siendo algunas de ellas MapReduce/Hadoop, NoSQL, herramientas de *reporting* y RDBMS (sistemas de gestión de bases de datos relacionales), lo anterior en correspondencia con [51].

2.2.3. Análisis y visualización de datos

Una vez que se cuenta con la información, ahora se debe saber qué hacer con ella e interpretarla correctamente. Debido a que se maneja una enorme cantidad de datos es necesario contar

con herramientas que permitan asegurar la calidad de la información. Algunas de estas técnicas de análisis son las siguientes: Aprendizaje automático o *machine learning* que trata de clasificar y realizar predicciones a partir de los datos almacenados. *Data mining* o minería de datos, que su base es extraer conocimiento a partir de los datos por medio de reconocimiento de patrones, usando técnicas de *machine learning*, técnicas estadísticas y utilizando *scripts* en para Java, Scala o Python. Aprendizaje supervisado, los cuales detectan patrones en los datos a través de algoritmos especializados. Así como, el aprendizaje no supervisado y *topic modeling*, el cual, por medio de procedimientos inductivos, extrae temáticas a partir de cuerpos de documentos, lo anterior de acuerdo con [51].

2.2.4. Extracción de datos

La minería de datos, en correspondencia con [50], es un software con herramientas para analizar y procesar datos. Esta técnica permite encontrar dependencias o patrones entre los campos en grandes volúmenes de bases de datos. Por lo que, utilizando las herramientas correctas, se puede analizar, categorizar e identificar relaciones que permitan a los usuarios tener un panorama completo para una mejor toma de decisiones. Para los sistemas de *big data* el mayor desafío es tener la capacidad de usar los grandes volúmenes de información para explorar y buscar registros específicos, que arrojen información útil para los usuarios.

2.2.5. Metadatos

Un metadato, como se refiere en [52], es un dato sobre datos o recursos en su concepción original, sin embargo y al paso del tiempo se han incorporado atributos a estos datos que amplían su contexto y aportan información que describe mas a detalle al recurso. Algunas de las funciones de estos atributos pueden ser mostrar relaciones entre los datos, gestión de estos, permitir su búsqueda, entre otras.

2.2.6. XML

El XML (eXtensible Markup Language), es un metalenguaje derivado del SGML (Standard Generalized Markup Language) que es el estándar ISO 8879 para el tratamiento de la información, utilizado para crear una estructura en forma de árbol de los metadatos tanto generales como específicos, de tal forma que se puede acceder a estos por medio de su sistema de etiquetado, de acuerdo con lo indicado en [52].

2.2.7. JSON

JSON (JavaScript Object Notation) es un formato utilizado para el intercambio de datos, esta basado en el Standard ECMA-262 3rd Edition, debido a que es simple su lectura y escritura para el humano e interpretarlo y generarlo para las máquinas es un lenguaje muy conocido y utilizado por

muchos desarrolladores. Se construye en dos estructuras universales, colección de pares (nombre/valor) y listas, en correspondencia con [53].

2.2.8. Python

Python, de acuerdo con [54], es conocido como un lenguaje de programación de alto nivel, creado por Guido van Rossum en 1991, desarrollado como software libre, sus características son: lenguaje interpretado al utilizar fuertemente tipado y tipado dinámico, multiplataforma al no depender de ningún sistema operativo, multiparadigma al permitir estilos de programación como programación imperativa, orientación a objetos y funcional, además con formato de código estructural.

2.2.9. Django

De acuerdo con [55], Django es un *framework* de código abierto para aplicaciones web escrito en Python.

2.2.10. Node.js

Node.js conforme a [56], es un entorno multiplataforma de código abierto en un entorno de desarrollo homogéneo en JavaScript entre el cliente y servidor.

2.2.11. Express.js

En base a [57], Express.js, es un *framework* de aplicaciones web para Node.js de código abierto, utiliza MongoDB como su base de datos NoSQL estándar.

2.2.12. Flask

De conformidad con [58], Flask es un *microframework* web escrito en Python, permite desarrollar aplicaciones web sin librerías o herramientas particulares.

2.2.13. Weewx

Software de código abierto escrito en Python utilizado por muchas estaciones meteorológicas para producir gráficos, reportes y paginas HTML. Incluye la mayoría de los controladores para las diferentes marcas y modelos de las estaciones meteorológicas, así como también en una Raspberry Pi. *Weewx* tiene opción para ejecutarse directo o como demonio, lo anterior se refiere en [59]. Este software utiliza únicamente una base de datos estándar para todas las estaciones y solo se registran valores de acuerdo con los sensores que contiene la estación. Las estaciones meteorológicas instaladas por el CECATEV utilizan el software *weewx*. En la Tabla **9** se muestra la estructura de archivo que utiliza *weewx* formada por 53 campos, esta estructura es en la que se almacena la información de las estaciones meteorológicas del CECATEV en SQL.

Tabla 9. **Tipo de archivo usado por weewx**

Tipo de archivo weewx				
altimeter	**hail**	**leafWet1**	**rxCheckPercent**	**usUnits**
barometer	**hailRate**	**leafWet2**	**soilMoist1**	**UV**
consBatteryVoltage	**heatindex**	**outHumidity**	**soilMoist2**	**windvec**
dateTime	**heatingTemp**	**outTemp**	**soilMoist3**	**windBatteryStatus**
dewpoint	**heatingVoltage**	**outTempBatteryStatus**	**soilMoist4**	**windDir**
ET	**inHumidity**	**pressure**	**soilTemp1**	**windGust**
extraHumid1	**inTemp**	**radiation**	**soilTemp2**	**windGustDir**
extraHumid2	**inTempBatteryStatus**	**rain**	**soilTemp3**	**windSpeed**
extraTemp1	**interval**	**rainBatteryStatus**	**soilTemp4**	**windchill**
extraTemp2	**leafTemp1**	**rainRate**	**supplyVoltage**	
extraTemp3	**leafTemp2**	**referenceVoltage**	**txBatteryStatus**	

Fuente: Archivo weewx [60]

2.3. Marco Conceptual

En esta sección se definen los conceptos utilizados en el presente proyecto para interpretar correctamente la información.

Término	Depende de:	Interpretación
Factor		Es el nombre que recibe un grupo de unidades de medida que son equiparables entre sí, las que se pueden normalizar a una sola unidad de salida. Se utiliza el SI para la conversión de unidades.
Homogeneización		Es el procedimiento que agrupa los datos de sensores que pertenecen al mismo factor y realiza las conversiones necesarias entre estos para producir un dato en la unidad de salida elegida por el experto.
Estaciones		Se refiere a estaciones meteorológicas o de gases y partículas suspendidas que a través de sus sensores recolectan información heterogénea, esta información es construida como documento para integrarla a MongoDB.
Nombre	estaciones	Las estaciones tienen un atributo nombre que contiene el apelativo designado a esa estación.
sistemamt	estaciones	Sistema métrico con el que se configuró el *datalogger* de la estación para la recolección de las lecturas de las variables climáticas.

Término	Depende de:	Interpretación
intervalo	estaciones	Corresponde al tiempo en minutos en el que es almacenado el promedio de las muestras de las variables en el *datalogger*.
altura	estaciones	Altura en la que se ubica la estación con respecto a su base.
tablasql	estaciones	Tabla en la que se encuentra almacenada la información de la estación en el servidor SQL.
url	estaciones	Dirección *url* de la estación en CECATEV.
ubicación	estaciones	Localización física de la estación, la cual contiene descripción, domicilio, municipio y estado.
descripción	ubicación	Localización rápida del lugar en donde se ubica la estación (Bomberos anapra, por ejemplo).
domicilio	ubicación	Lugar donde se localiza la estación.
municipio	ubicación	Municipio donde se localiza la estación.
estado	ubicación	Estado en donde se ubica la estación.
geolocalización	estaciones	Ubicación geográfica en coordenadas de la estación, contiene latitud, longitud y altitud.
latitud	geolocalización	Coordenada de posición horizontal de la ubicación de la estación (norte, sur).
longitud	geolocalización	Coordenada de posición horizontal de la ubicación de la estación (este, oeste).
altitud	geolocalización	Distancia de la estación respecto al nivel del mar.
marca	estaciones	Marca de la estación (Davis, Campbell, entre otras).
modelo	estaciones	Modelo de la estación (Vantage Pro2, Teledyne API, etc).
tipo	estaciones	Tipo de la estación, meteorológica, de gases y partículas suspendidas u otro.
fechains	estaciones	Fecha de instalación de la estación.
fechaact	estaciones	Fecha de la última actualización de la estación.
datalogger	estaciones	Marca del *datalogger* instalado en la estación.
sw	estaciones	Software con el que opera el *datalogger* de la estación.
características	estaciones	Alguna característica adicional de la estación.
imagen	estaciones	Fotografía de la estación.
Sensores		Cada sensor que conforman a la estación.
unidades	sensores	Unidad de medida con la que toma la muestra el sensor.
mínimo	sensores	Valor mínimo de lectura del sensor para que se encuentre dentro del rango de parámetros normales.
máximo	sensores	Valor máximo de lectura del sensor para que se encuentre dentro del rango de parámetros normales.
marca	sensores	Marca del sensor instalado en la estación.
modelo	sensores	Modelo del sensor instalado en la estación.
sql	sensores	Nombre del campo en el cual se almacena la información del sensor en la tabla SQL.

Término	Depende de:	Interpretación
columna	sensores	Nombre de la columna utilizado en el encabezado de los *dataset* en formato csv.
factor	sensores	Grupo de conversión al que pertenece el sensor, si no tiene este atributo significa que el valor del sensor no tiene equivalencia a otra unidad de medida y debe quedar en su unidad recolectada.
Dataset		Conjunto de datos filtrados del repositorio en MongoDB, de acuerdo con los parámetros elegidos por el experto.
Control de calidad		Proyecto realizado en paralelo al proyecto referido en este documento, también es parte del sistema de *big data* de la UACJ.
weewx		Software utilizado por las estaciones meteorológicas para producir gráficos, reportes y páginas HTML
timestamp		Es la marca de tiempo actual de Unix, en este caso se representa en tiempo universal coordinado (UTC/GMT)
R		RStudio. IDE de Código abierto para lenguaje R, dedicado al análisis estadístico de datos y gráficos.
State (estado)		Es el estado en el que se encuentran los documentos (registros) almacenados en MongoDB.
state 0		Indica que el documento se encuentra sin haber pasado por ningún proceso de control de calidad.
state 1		El documento pasó por el proceso de control de calidad y no se detectó ninguna lectura sospechosa en algún sensor contenido en el documento, cambia el valor del atributo *state* a 1.
state 2		Al pasar el documento por el proceso de control de calidad, uno o varios de los sensores contienen lecturas fuera de sus rangos de máximo o mínimo, por lo que se consideran sospechosos, cambia el valor del atributo *state* a 2 y se agregan dos atributos en el nivel del valor de sensor, *pred* que contiene el valor sugerido a cambiar y *newvalue* que contiene el mismo valor de lectura original.
state 3		Una vez que el experto decide el nuevo valor que debe tener la lectura del sensor que se consideró sospechoso, cambia el valor del atributo *newvalue* por el nuevo valor y el *state* cambia a 3. En ningún proceso se altera la información original lo que permite garantizar la integridad del repositorio.
ppb		Es una abreviación de partes por billón, es la medida de una sustancia en líquido o gas.
ppm		Es una abreviación de partes por millón, es la medida de una sustancia en líquido o gas.

3. Metodología

En esta sección se detalla el diseño y realización de el *framework* de este proyecto. Pasando por el análisis, diseño y creación de este, las opciones evaluadas para su desarrollo y como es que se tomaron las decisiones para llegar al producto terminado.

3.1. Minería de datos

Se utilizó un sistema ETL para este desarrollo, la fase de extracción constó de un análisis de la información existente, identificando las diferentes fuentes que proveen los datos, las cuales ayudaron para la creación de los metadatos. Se crearon los metadatos necesarios para la extracción y transformación.

En la fase de transformación, se crearon las librerías que se encargarán de homogeneizar los datos, de acuerdo con las variables seleccionadas y generar los entregables solicitados. Para esta fase se combinan los metadatos correspondientes a las conversiones con los datos en MongoDB para normalizar las unidades de salida.

Para la fase de carga, se entregarán los *datasets* resultantes incluyendo las variables solicitadas en la fase de transformación para poder ser utilizadas por los expertos en herramientas para su análisis avanzado.

3.2. Análisis

En el inicio del proyecto se realizaron reuniones continuas con los expertos del laboratorio y asesores del proyecto para realizar elicitación de requisitos.

Durante estas reuniones se explicó el funcionamiento de las estaciones meteorológicas y las estaciones de gases y partículas suspendidas principalmente, el proceso por medio del cual los sensores recolectan las lecturas de las muestras de temperatura, lluvia, humedad, presión atmosférica, velocidad del viento, radiación solar, radiación ultravioleta, así como contaminantes ambientales como dióxido de azufre, monóxido de carbono, dióxido de carbono, dióxido de nitrógeno, ozono, y partículas atmosféricas, estas variables son almacenadas en el *datalogger* de cada estación, posteriormente y de acuerdo con el intervalo de tiempo, el cual es definido al momento de configurar el *datalogger* de cada estación, se calcula un promedio de las muestras y ese conjunto de datos en un solo registro se sube a una tabla en el servidor SQL. En este servidor se encuentra la mayor parte de la información con la que cuenta el CECATEV.

El esquema de la tabla en SQL es una estructura predefinida por el software *weewx*, sin embargo, esta estructura esta formada por 53 campos, de los cuales solo se llenan los que utiliza cada estación, un aproximado de 20 campos dependiendo de los sensores con los que cuente la estación.

Por lo que los campos que no almacenan valores se encuentran ocupando espacio, representando uno de los inconvenientes de utilizar este tipo de base de datos. En el Anexo I se muestran las estructuras weewx para la estación 26 y la estación Teledyne, indicando los campos que no se utilizan para esas estaciones.

Además de la información que ya se encuentra almacenada en SQL, existen datos en otros archivos de tipo texto en su mayoría, los cuales podrían ser incorporados al repositorio posteriormente.

Por lo anterior y de acuerdo con el análisis realizado, se detectaron diferentes requerimientos, uno de los más importantes fue el manejo de los datos heterogéneos, en este sentido era necesario contar con una herramienta que permitiera almacenar la información de una manera eficiente y optimizada, entonces se decidió utilizar una base de datos NoSQL, considerando que este tipo de base de datos tiene una estructura descentralizada, lo que permite utilizar esquemas distribuidos y almacenar cada registro de SQL como documento independiente. De esta manera se logra optimizar la base de datos ya que solo se almacenan los campos con información por cada estación eliminando el resto de los campos no necesarios.

Para elegir la base de datos NoSQL que se utiliza en el proyecto del sistema de *big data* de la UACJ se consideraron aspectos como la forma en que se almacenan los datos utilizando archivos en formato tipo JSON, lo que permite que las consultas para grandes cantidades de datos están optimizadas, además de ser una de las más populares por lo que hay mucha documentación disponible, de acuerdo con [61].

Otra necesidad que tenían es debido a los compromisos que cuenta la UACJ por medio del CECATEV con diversas universidades dentro y fuera del país, se debe compartir la información climática a diferentes usuarios, entre universidades y expertos en la materia. Para esto es necesario agrupar los datos, homogeneizarlos, es decir, que se realicen las conversiones necesarias de acuerdo con la unidad de medida de la lectura del sensor de la estación, que permita entregar un *dataset* que contenga la información homologada de los datos existentes para uso de los expertos.

3.3. Estructura de metadatos

Inicialmente se diseñaron y crearon los metadatos contenidos en un archivo XML, para que las diferentes aplicaciones de los sistemas de la UACJ pudieran acceder, hospedado en un *web service*, el cual se encuentra formado en 4 grupos. Más información en el Anexo II.

El primer grupo está formado por las configuraciones que contiene la IP del servidor de MongoDB.

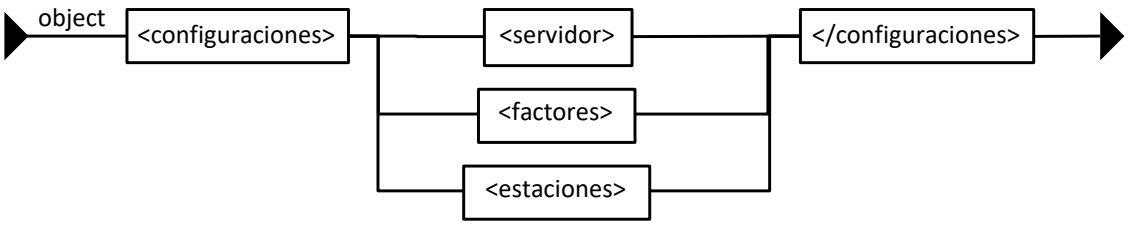
El segundo grupo contiene los factores de conversión y sus unidades de medida, es decir, para una unidad de medida en cuales otras unidades de medida se pueden normalizar. A cada sensor se le agrega como un atributo a que factor de conversión pertenece, en caso de no tener este atributo significa que es un índice o porcentaje que son unidades que no tienen conversión a otras unidades.

En el tercer grupo se forma por los metadatos de las estaciones, como nombre, sistema métrico que utiliza, el intervalo de escritura a SQL, la altura de la estación con respecto a su base, su correspondiente tabla en SQL, su ubicación física como domicilio, municipio y estado, su geolocalización como latitud, longitud y altitud, marca, modelo, tipo de estación pudiendo ser meteorológica o gases y partículas suspendidas, también se encuentra información como la fecha instalación, fecha de actualización, tipo de *datalogger*, el *software* del *datalogger*, y alguna característica especial que tenga la estación, así como su imagen.

En el cuarto grupo se encuentran los sensores que corresponden a cada una de las estaciones, con su correspondiente valor máximo, mínimo, la unidad de medida en la que se recolecta su lectura, marca, modelo, su campo correspondiente en SQL y el factor de conversión al que pertenece en caso de que su unidad de medida permita alguna conversión a otra unidad.

El la Figura **10** se muestra el grupo 1 de los metadatos en el archivo XML.

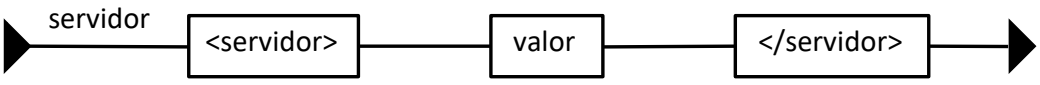
Figura 10. **Metadatos grupo 1**



Fuente: Elaboración propia

En la Figura **11** se indica la IP del servidor que hospeda la base de datos en MongoDB.

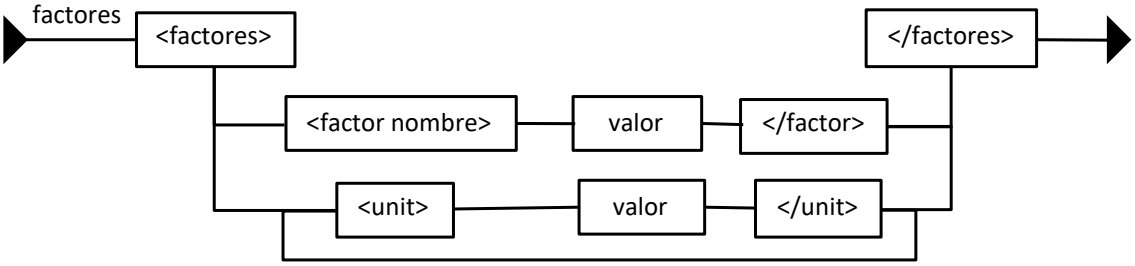
Figura 11. **Metadatos - IP del servidor**



Fuente: Elaboración propia

En la Figura **12** se muestra el metadato para los factores de conversión.

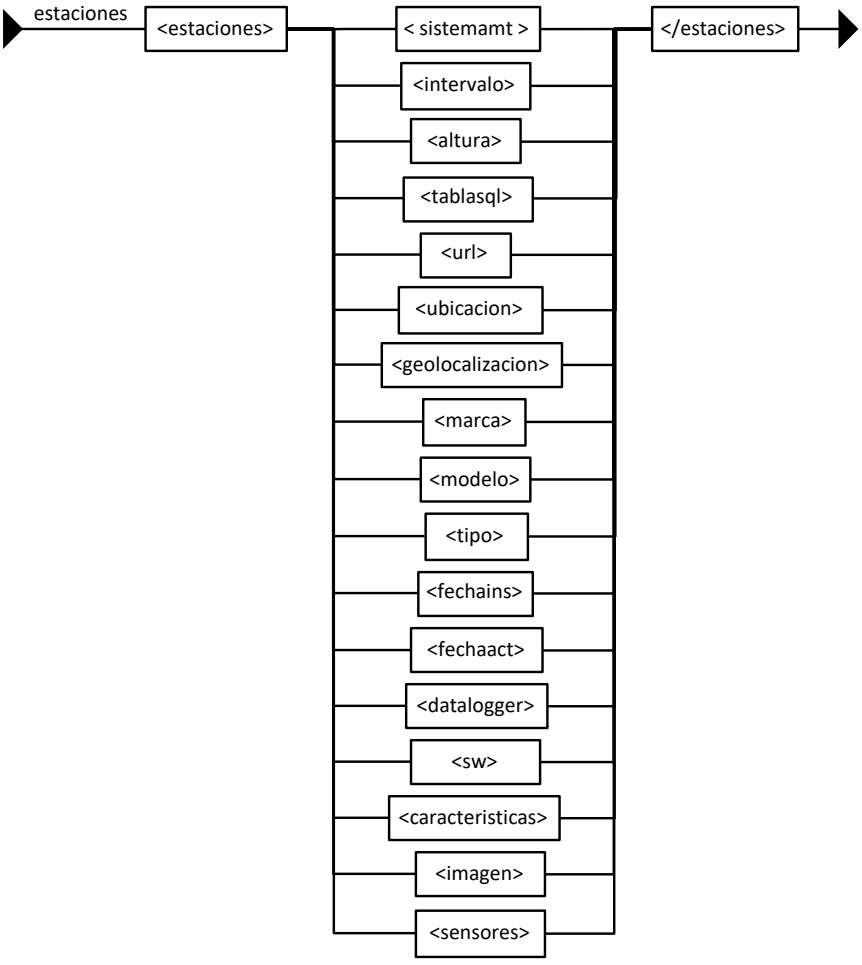
Figura 12. **Metadatos factores de conversión**



Fuente: Elaboración propia

En la Figura **13** se representan los metadatos de las estaciones y su información.

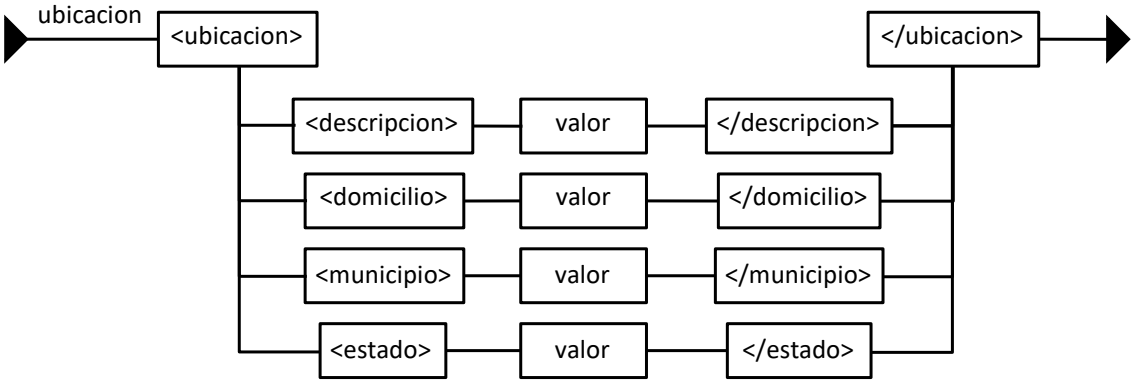
Figura 13. **Metadatos estaciones**



Fuente: Elaboración propia

En la Figura **14** se muestran los metadatos de la ubicación física de las estaciones.

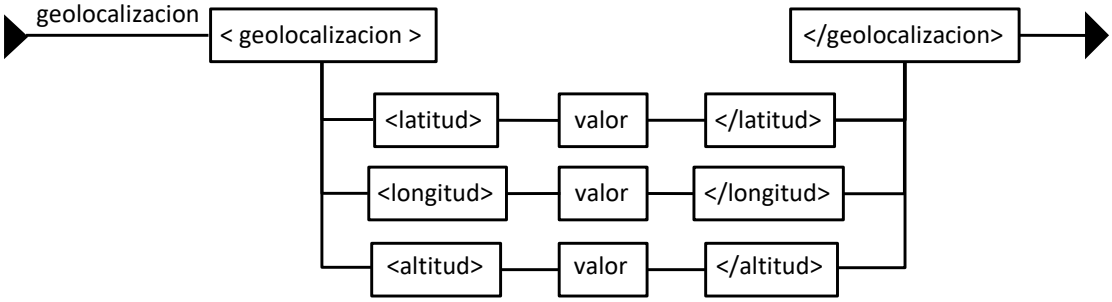
Figura 14. **Metadatos de ubicación**



Fuente: Elaboración propia

En la Figura **15** se observan los metadatos correspondientes a la geolocalización de las estaciones.

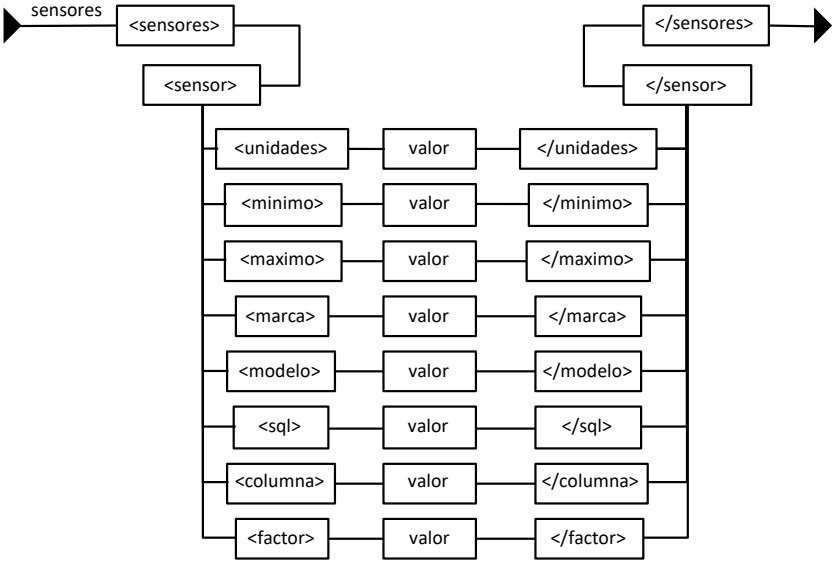
Figura 15. **Metadatos de geolocalización**



Fuente: Elaboración propia

En la Figura **16** están representados los metadatos de los sensores que pertenecen a la estación.

Figura 16. **Metadatos de sensores**

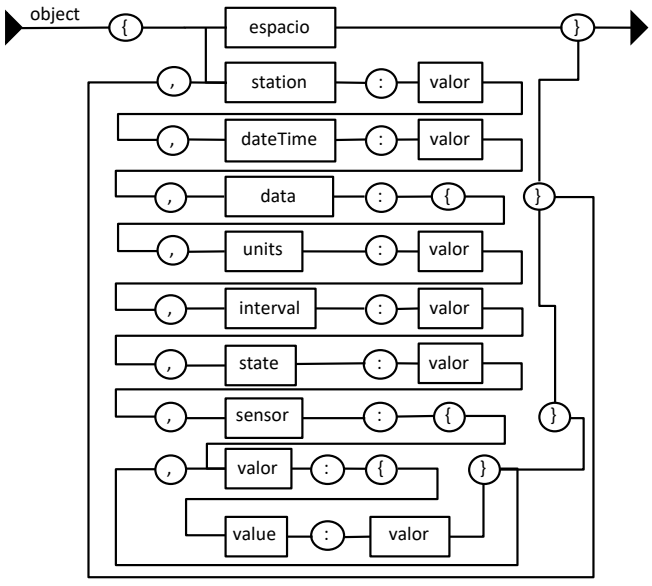


Fuente: Elaboración propia

3.4. Estructura JSON de SQL a MongoDB

Después de definir los metadatos, se diseñó la estructura del documento JSON con el que se importan los registros del servidor SQL al servidor de MongoDB. Se construye el documento JSON únicamente con la información que se almacena de la estación, de acuerdo con los metadatos descritos en el archivo XML. En la Figura **17** se observa la representación del esquema del documento JSON a importar.

Figura 17. **Esquema de documento JSON**



Fuente: Elaboración propia

3.5. Desarrollo

Terminando con la fase de elicitación de requisitos, se analizaron posibles desarrollos de herramientas que solucionen las necesidades del CECATEV.

Una de las herramientas que se decidió desarrollar consiste en un *framework* formado por un conjunto de librerías, las cuales realizan funciones definidas, por lo que en conjunto permiten generar un *dataset* con información homogénea para apoyo de los expertos.

Para el diseño del *framework* se decidió utilizar Python por sus ventajas en el uso de funciones matemáticas y la compatibilidad con MongoDB.

Primero se creó una colección *users* para gestión de usuarios en MongoDB que permite el manejo ordenado de las sesiones. En esta colección se almacenan los usuarios y las colecciones que usan en su sesión.

La librería para el inicio de sesión a su vez esta formada por otras librerías *register, create_session, login* y *logout*. En la función *register* se valida que el usuario no exista para crearlo y manda llamar a la librería *create_session*, para almacenar en la colección *users* el correo del usuario, contraseña y dos colecciones con las que va a operar en el *framework*. La función *login* autentica al usuario y carga las colecciones con las que estaba trabajando. Y *logout* cierra la sesión del usuario en el *framework*.

En la función *create_session*, se crean las dos colecciones correspondientes a la sesión del usuario, la primera colección almacena los parámetros de la consulta deseada (*parametersX*) y la otra (*sessionX*) contiene el *dataset* resultante para que en esta se apliquen las variantes elegidas por el usuario. Estas dos colecciones en la generación de su nombre tienen una X al final (*sessionX* y *parametersX*), donde X es un número aleatorio entre 1 y 1000, primero genera el número aleatorio y revisa que no exista alguna colección con ese número para evitar colisiones con otros usuarios, cuando se valida la colección, regresa el número para que pueda crear las colecciones. Estas colecciones se crean para permitir al usuario guardar su consulta y realizar cambios a la misma sin volver a cambiar todos los parámetros seleccionados. Las colecciones se crean al registrar el usuario y se conservan durante toda su sesión y se eliminan al dar de baja al usuario o en un proceso de depuración en un tiempo determinado. Una muestra de la colección *users* en MongoDB se puede observar en la Figura **18**.

Figura 18. **Colección *users* MongoDB**


```
_id: ObjectId("5e1663c590f42926a24ee39d")
email: "prueba1@hotmail.com"
password: "+N5JER1O8pg/B0nTGsuJ5A8ISoOUU1pPGxS27a7f0kHbJU0883JiiAF0j74luSx4"
session: "session861"
parameters: "parameters861"

_id: ObjectId("5e1685df90f4293870a0cf96")
email: "prueba2@hotmail.com"
password: "rNSb2sYYE4m5Ac/geyWcCVThFghtL0dQAFus9tEFLbrT/Ia+YW5TJrVkY6Lu9r7o"
session: "session78"
parameters: "parameters78"

_id: ObjectId("5e1686c890f429393e721b52")
email: "prueba3@hotmail.com"
password: "AXuNCAOircsFoy0Q733bQSfTBWrb6Idkhri3FhOUVrwETlZdYRVl3/lg4Ql7FnYM"
session: "session778"
parameters: "parameters778"
```


Fuente: Elaboración propia

Una vez iniciada la sesión el usuario en la librería *list_stations* el usuario elige una o varias estaciones de las cuales se va a extraer la información. Las estaciones las extrae de los metadatos definidos en el archivo XML. Al elegir las estaciones se almacenan en la colección de *parametersX* para su posterior acceso. En la Figura **19** muestra el archivo que le corresponde a la sesión y se observan las estaciones disponibles en el archivo XML, se seleccionan las estaciones Estación 25, Estación 26 y Teledyne.

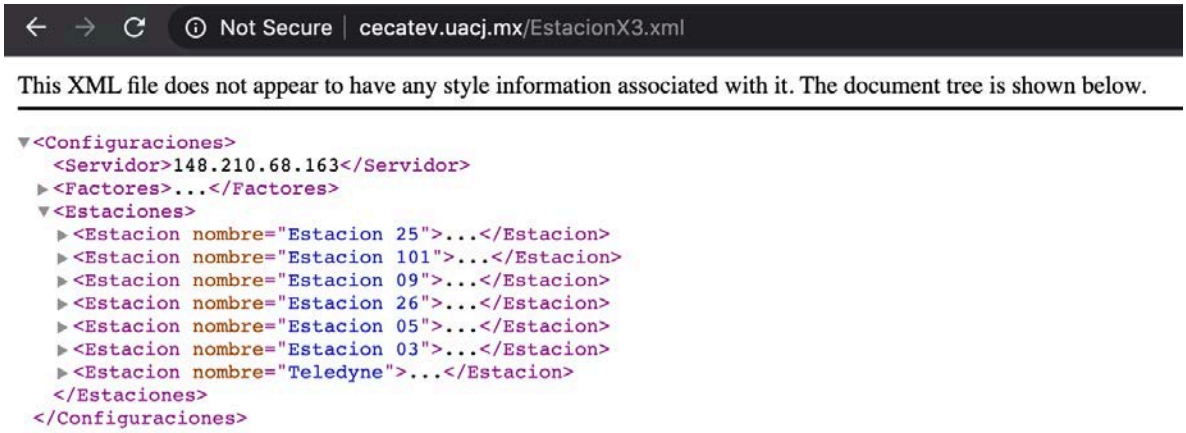
Figura 19. **Inicio de sesión y selección de estaciones en el *framework* (1,4 y 7), librería *init* y *list_stations***

```
Sesion:  session904

Estaciones disponibles

1 - Estacion 25 -- Meteorologica
2 - Estacion 101 -- Meteorologica
3 - Estacion 09 -- Meteorologica
4 - Estacion 26 -- Meteorologica
5 - Estacion 05 -- Meteorologica
6 - Estacion 03 -- Meteorologica
7 - Teledyne -- Gases y Particulas suspendidas

Elija las estaciones de la lista, separando por coma cada estación (1,2,3,..) o (0) para todas las estaciones
Presione enter para finalizar
>1,4,7
```

Fuente: Elaboración propia

A continuación, en la Figura **20** se muestran las estaciones que se encuentran en el archivo XML y se observa su coincidencia con lo mostrado en la Figura **19**.

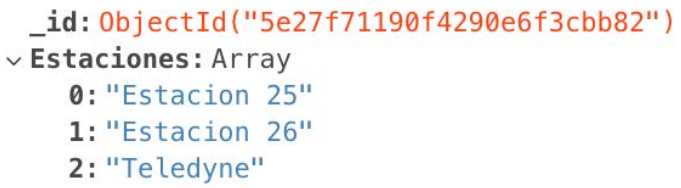
Figura 20. **Metadatos de estaciones en archivo XML**



Fuente: Elaboración propia

Una vez elegidas las estaciones se almacenan en la colección *parametersX* de MongoDB como se muestra en la Figura **21**.

Figura 21. **Colección *parametersX* en MongoDB**



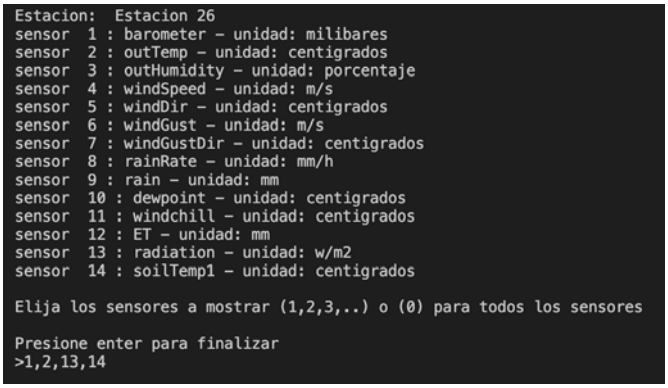
Fuente: Elaboración propia

Otra librería llamada *list_sensors* consulta las estaciones seleccionadas en la colección *parametersX* y muestra los sensores que corresponden a esas estaciones para ser elegidos, en este caso se muestran los ejemplos para las estaciones: Estación 25, Estación 26 y Teledyne, mostrados en la Figura **22**, Figura **23** y Figura **24**. Estos sensores son consultados de los metadatos definidos en el archivo XML, tal como se muestra en la Figura **25**, para la estación 25. Los sensores elegidos por estación son almacenados en la colección de *parametersX*, y se genera el encabezado que se utiliza para el *dataset* en formato csv, mismo que se almacena en la colección *parametersX* Figura **26**, además en esta librería se graban los metadatos de unidad de medida, peso molecular y el grupo de factor al que pertenece, como se muestra en la Figura **27**, y al mismo tiempo busca los factores de conversión que corresponden a los sensores seleccionados, estos últimos también se almacenan en la misma colección Figura **28**. En la Figura **29** se observan los factores y el encabezado generado por el *framework* para grabarlos en la colección.

Figura 22. **Sensores que pertenecen a la estación 25, extraído del archivo XML, librería *list_sensors***

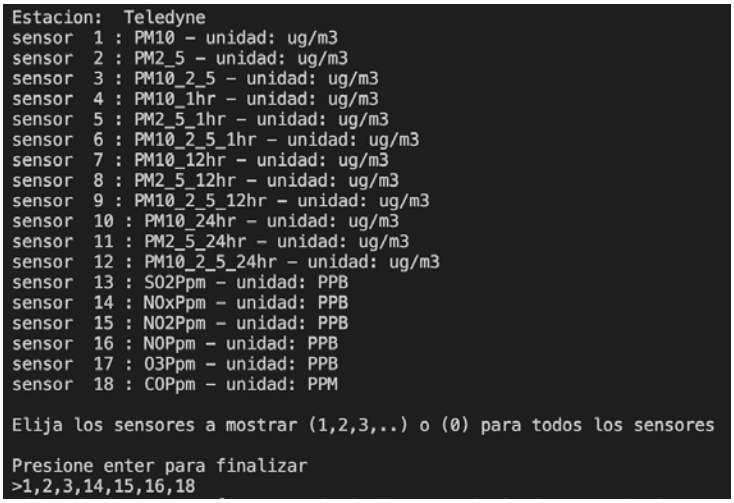
Fuente: Elaboración propia

Figura 23. **Sensores que pertenecen a la estación 26, extraído del archivo XML**



Fuente: Elaboración propia

Figura 24. **Sensores que pertenecen a la estación Teledyne, extraído del archivo XML**



Fuente: Elaboración propia

Figura 25. **Metadatos de sensores de la estación 25, archivo XML**

```
<Estacion nombre="Estacion 25">
  <!--  Davis  -->
  <nombre>Estacion 25</nombre>
  <sistemamt>16</sistemamt>
  <intervalo>5</intervalo>
  <altura>1.50</altura>
  <tablasql>weewx25</tablasql>
  <url>url</url>
  <ubicacion>...</ubicacion>
  <geolocalizacion>...</geolocalizacion>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <tipo>Meteorologica</tipo>
  <fechains>Sep-07</fechains>
  <fechaact>xfechaA</fechaact>
  <datalogger>ISS</datalogger>
  <sw>WaterLink</sw>
  <caracteristicas>xcaracteristicas</caracteristicas>
  <imagen>ximagen</imagen>
  <sensores>
    <barometer>...</barometer>
    <pressure>...</pressure>
    <altimeter>...</altimeter>
    <inTemp>...</inTemp>
    <outTemp>...</outTemp>
    <inHumidity>...</inHumidity>
    <outHumidity>...</outHumidity>
    <windSpeed>...</windSpeed>
    <windDir>...</windDir>
    <windGust>...</windGust>
    <windGustDir>...</windGustDir>
    <rain>...</rain>
    <rainRate>...</rainRate>
    <dewpoint>...</dewpoint>
    <windchill>...</windchill>
    <heatindex>...</heatindex>
    <ET>...</ET>
    <radiation>...</radiation>
    <UV>...</UV>
  </sensores>
</Estacion>
```

Fuente: Elaboración propia

Figura 26. **Documento para encabezados en la colección *parametersX***

```
_id: ObjectId("5e2898f890f4290e6f3cbb9b")
Encabezados: Array
  0: "dateTime"
  1: "Estacion 25"
  2: "barometer"
  3: "pressure"
  4: "altimeter"
  5: "UV"
  6: "Estacion 26"
  7: "barometer"
  8: "outTemp"
  9: "radiation"
  10: "soilTemp1"
  11: "Teledyne"
  12: "PM10"
  13: "PM2_5"
  14: "PM10_2_5"
  15: "NOxPpm"
  16: "NO2Ppm"
  17: "NOPpm"
  18: "COPpm"
```

Fuente: Elaboración propia

Figura 27. **Documento valores almacenado en la colección *parametersX***

Fuente: Elaboración propia

Figura 28. **Documento *Factors* en la colección *parametersX***

Fuente: Elaboración propia

Figura 29. **Documento valores almacenado en la colección *parametersX***

```
Lista de factores de conversión: ['Barometrico', 'Concentracion ppb', 'Concentracion ppm', 'Radiacion', 'Temperatura']

Encabezados: ['Estacion 25', 'barometer', 'pressure', 'altimeter', 'UV', 'Estacion 26', 'barometer', 'outTemp', 'radiation',
 'soilTemp1', 'Teledyne', 'PM10', 'PM2_5', 'PM10_2_5', 'NOxPpm', 'NO2Ppm', 'NOPpm', 'COPpm']
```

Fuente: Elaboración propia

La siguiente librería *list_factors* es la que se encarga de solicitar al usuario las unidades de salida que desea para el *dataset* de acuerdo con los factores de los sensores elegidos. Los factores son consultados de la colección *parametersX* y las diferentes unidades de salida las extrae de los metadatos definidas en el archivo XML. En la Figura **30**, se observa cómo se realiza la selección de unidades de medida para los grupos de factores que corresponden a los sensores elegidos.

Figura 30. **Selección de unidades de salida para los grupos de factor de conversión en el *framework*, librería *list_factors***

```
Los sensores seleccionados pueden generar su salida en las siguientes unidades:
Factor de conversion :  Barometrico
1 - milibares
2 - Pa
3 - Atm
4 - inHg
5 - hPa
6 - mmHg
7 - Psi
8 - Torr

Elija la unidad de salida (1...) o (0) para la unidad base del sensor
>2

Los sensores seleccionados pueden generar su salida en las siguientes unidades:
Factor de conversion :  Concentracion ppb
1 - ppb
2 - ppm
3 - ug/m3

Elija la unidad de salida (1...) o (0) para la unidad base del sensor
>2

Los sensores seleccionados pueden generar su salida en las siguientes unidades:
Factor de conversion :  Concentracion ppm
1 - ppm
2 - ppb
3 - ug/m3

Elija la unidad de salida (1...) o (0) para la unidad base del sensor
>2

Los sensores seleccionados pueden generar su salida en las siguientes unidades:
Factor de conversion :  Radiacion
1 - w/m2
2 - langley
3 - btu/ft2
4 - joules/m2

Elija la unidad de salida (1...) o (0) para la unidad base del sensor
>4

Los sensores seleccionados pueden generar su salida en las siguientes unidades:
Factor de conversion :  Temperatura
1 - centigrados
2 - fahrenheit
3 - kelvin

Elija la unidad de salida (1...) o (0) para la unidad base del sensor
>2
```

Fuente: Elaboración propia

En la Figura **31**, se indica como quedan las unidades de salida para los grupos de factores de conversión y se actualiza la colección de *parametersX*, como se muestra en la Figura **32**.

Figura 31. **Unidades de salida elegidos para los grupos de factores**

```
{'Barometrico': 'Pa', 'Concentracion ppb': 'ppm', 'Concentracion ppm': 'ppb', 'Radiacion': 'w/m2', 'Temperatura': 'fahrenheit'}
```

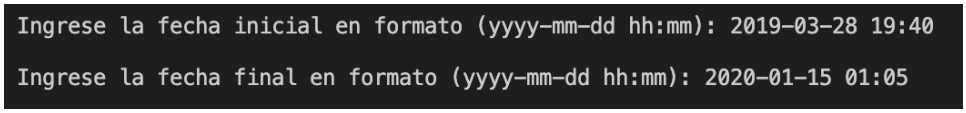
Fuente: Elaboración propia

Figura 32. **Documento Factors en la colección *parametersX***

Fuente: Elaboración propia

En otra librería *range_date* se define el rango de fechas para la información que va a contener el *dataset*, las fechas se manejan con el tiempo universal coordinado (UTC/GMT) tal como se almacena en SQL, este rango de fechas se almacena de igual manera en la colección *parametersX*. La Figura **33**, representa la pantalla para la captura del rango de fechas en el *framework*.

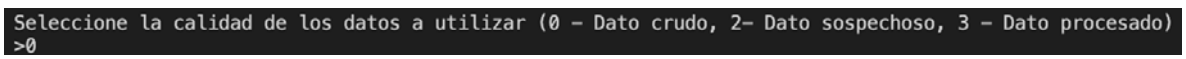
Figura 33. **Librería *range_date* en el *framework*, para selección de rango de fechas**



Fuente: Elaboración propia

Para elegir la calidad de los datos existe la librería *quality_data*, como se muestra en la Figura **34**, la cual permite seleccionar datos originales que incluyen datos con *state* 0 o recién insertados y 1 que pasaron por el proceso de calidad y son datos correctos. Datos sospechosos que incluyen datos *state* 2 los cuales pasaron por el proceso de calidad y se encuentran fuera de los rangos de permitidos de máximo y mínimo. Y datos corregidos, los cuales incluyen los datos *state* 1 que son los que pasaron por el proceso de calidad y son correctos y los datos con *state* 3, que son aquellos que pasaron por el proceso de calidad y fueron corregidos por los expertos. La calidad de los datos seleccionada se almacena en la colección de *parametersX*.

Figura 34. **Librería *quality_data* para calidad de datos**



Fuente: Elaboración propia

El dato que se almacena en la colección de *parametersX* es 1 como se muestra en la Figura **35**, que indica que el *dataset* debe incluir los valores con estado 0 y 1 explicados anteriormente.

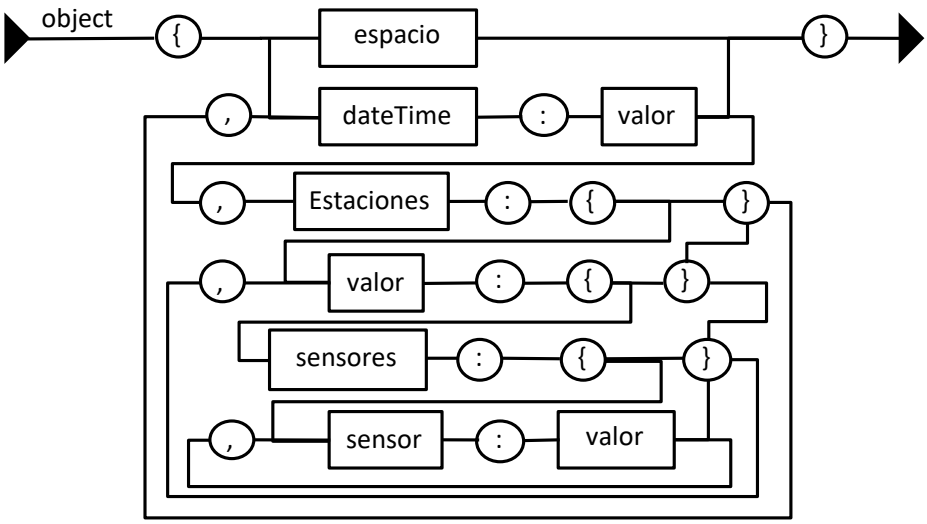
Figura 35. **Calidad de datos almacenada en *parametersX***

```
_id: ObjectId("5e2b5d6b4ca6fd66430a5d09")
State: "1"
```

Fuente: Elaboración propia

Al terminar de seleccionar los parámetros, la librería *save_session* utiliza estos parámetros almacenados en la colección *parametersX*, para realizar la consulta en el repositorio y por cada uno de los sensores manda llamar la librería *homogeneizar* que normaliza el valor del sensor de acuerdo con la unidad de salida elegida correspondiente al grupo del factor de conversión que pertenece y va grabando esos documentos de acuerdo a la estructura representada en la Figura **36**, dentro de la colección *sessionX*.

Figura 36. **Estructura documento JSON para almacenar en *session*X**



Fuente: Elaboración propia

La librería *homogeneizar* se compone de un conjunto de librerías que se encargan de realizar las conversiones entre las unidades equiparables, recibe el valor del sensor, su unidad de medida original, unidad de salida a la que va a normalizar, peso molecular si es que tiene y factor de conversión. El peso molecular se utiliza en algunas fórmulas para realizar la conversión a otra unidad. En caso de no tener factor de conversión, no ingresa a la librería *homogeneizar* y almacena el valor del sensor en su unidad original. Es importante mencionar que en los casos que el valor del sensor se encuentre NULL o vacío, de esta manera es almacenado en la colección, no se sustituye por cero ni se altera de ninguna forma su valor.

La última librería *switch_data* es la encargada de exportar la colección *sessionX* en el formato de salida elegida por el usuario, puede ser en formato JSON, csv y XML, además de incluir un archivo

readmeX, el cual contiene información de metadatos correspondiente a los parámetros elegidos por el usuario y contenido en el archivo de salida. Cada uno de los formatos de salida se realiza en funciones independientes. En la Figura **37** se muestra la librería *switch_data* para selección del formato de salida.

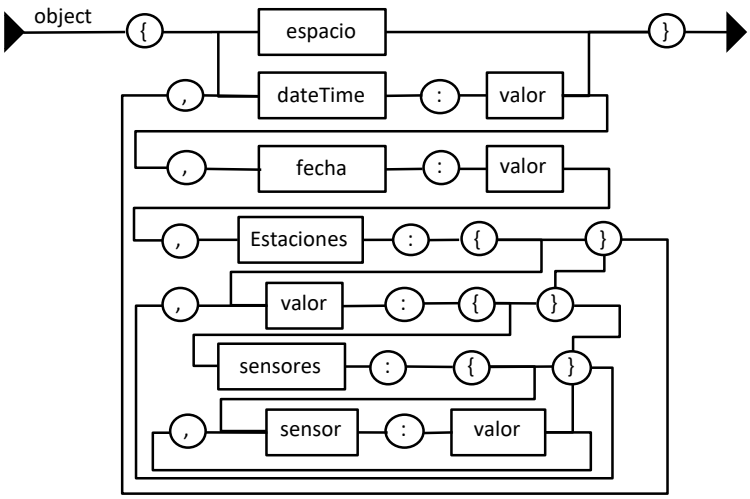
Figura 37. **Librería *switch_data framework***

```
Elija el formato para el dataset (1-Json, 2-csv, 3-xml)

Presione enter para finalizar
>1
Dataset en json
```

Fuente: Elaboración propia

A continuación, en la Figura **38** se representa la estructura de *dataset* para salida en formato JSON.

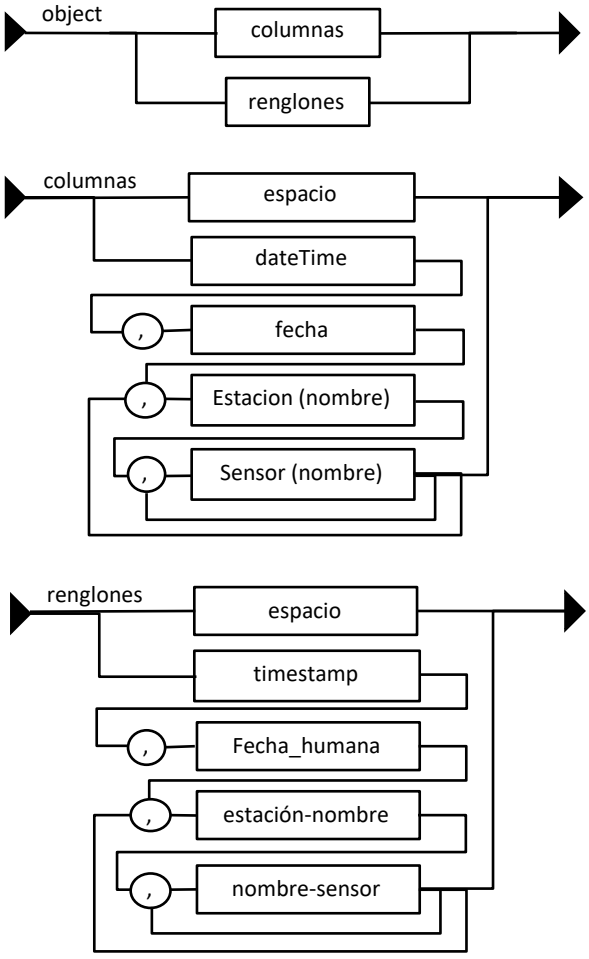
Figura 38. Estructura formato JSON



Fuente: Elaboración propia

En la Figura **39**, se muestra la estructura para el formato de salida en csv.

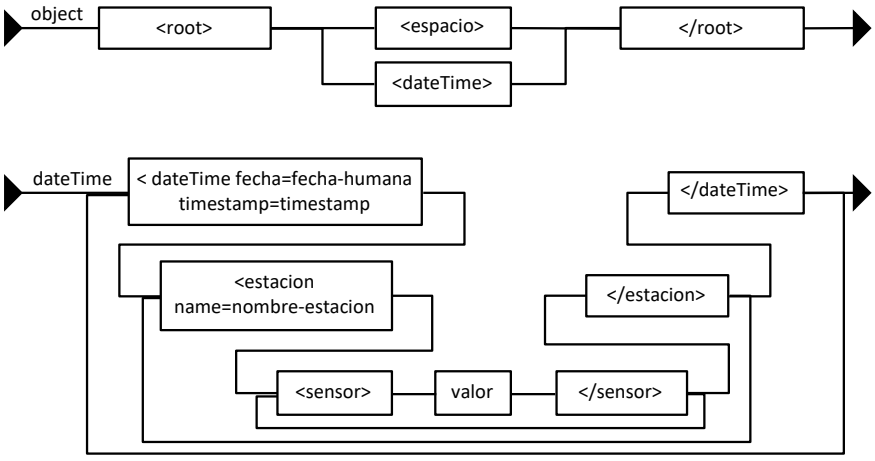
Figura 39. **Estructura formato csv**



Fuente: Elaboración propia

En la Figura **40** se muestra la estructura para el formato de salida en XML.

Figura 40. **Estructura formato XML**



Fuente: Elaboración propia

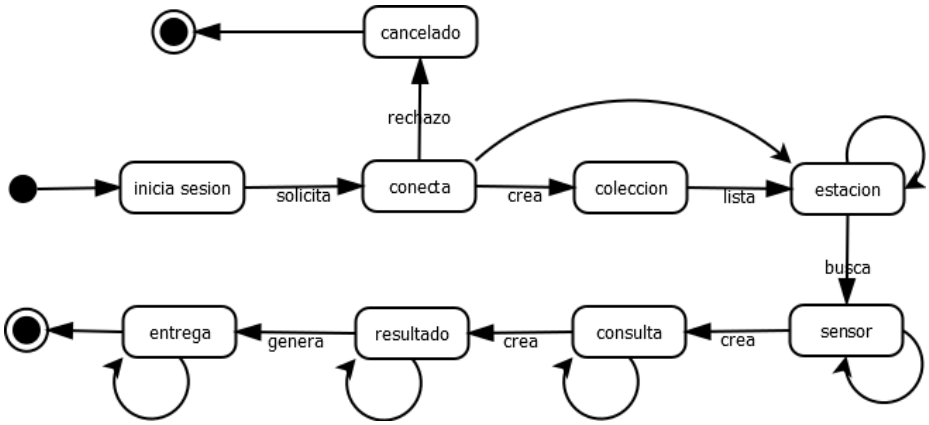
Fue necesario asegurar la conexión a MongoDB, cuando se trabajan con más de 30,000 documentos y debido a la velocidad con la que Python manda la instrucción de grabar el documento en MongoDB, se detectó que aproximadamente 20 documentos no se insertaban, por lo que para asegurar el almacenamiento de la totalidad de documentos, se agregó recursividad a la función de inserción obtiene el documento que no se grabó y se intenta volver a grabar, hasta que se asegura su almacenamiento.

Al terminar de grabar la colección en MongoDB, fue necesario indexarla por el *timestamp* para optimizar la consulta ya que se utiliza ordenamiento por este atributo, para conjuntos de datos menores a 32Mb no es requerido el índice.

3.6. Funcionalidad

En la Figura **41** se muestra el diagrama de los estados del *framework*. En algunos de los estados se muestra recursividad, esto se debe a que una vez que se obtienen los resultados, es posible regresar a cambiar la selección de estaciones, sensores, unidades de salida de los grupos de factores, rango de fechas, calidad de los datos y formato de salida del *dataset*.

Figura 41. **Diagrama de estados del *framework***



Fuente: Elaboración propia

Las librerías diseñadas fueron las siguientes:

1. *init* – Inicio de sesión, conecta base de datos y crea las colecciones *sessionX* y *parametersX*
2. *list_stations* – Listado de las estaciones extraídas del archivo XML
3. *list_sensor* – Listado de los sensores y unidades de las estaciones seleccionadas
4. *list_factors* – Listado de factores de conversión, para elección de unidad de salida
5. *range_date* – Rango de fechas para generar la consulta
6. *quality_data* – Calidad de los datos a incluir en el *dataset*
7. *save_sesion* – Almacena el resultado de la consulta en la colección *sessionX*

8. *homogeneizar* – Realiza el proceso de homogeneización de los valores de los sensores de la unidad de medida original a la unidad de medida de salida seleccionada
9. *switch_data* – Genera el *dataset* en el formato solicitado y genera el archivo *readmeX*
10. *close* – Cierra la sesión y desconecta la base de datos

3.7. Homogeneización

La librería *homogeneizar* encargada de realizar la normalización entre las unidades de medida de entrada y salida funciona con las siguientes conversiones:

La Tabla **10** se aplica para la conversión de las lecturas tomadas por los sensores *barometer, pressure* y *altimeter* de las estaciones meteorológicas.

Tabla 10. **Conversión de unidades para el factor barométrico**

Unidades	Mb (milibares)
Pa (Pascal)	mb = Pa * 0.01
Atm (atmósfera)	mb = Atm * 1013.25
in Hg (pulgadas de mercurio)	mb = in Hg * 33.86
hPa (hectopascal)	mb = hPa * 1
mm Hg (mm de mercurio)	mb = mm Hg * 1.33
Psi (libra por pulgada cuadrada)	mb = Psi * 68.95
Torr	mb = Torr * 1.33

Fuente: Elaboración propia de acuerdo con las conversiones del SI

Para las conversiones de las lecturas de los sensores *inTemp, outTemp, winDir, winGusDir, dewpoint* y *windchill* de las estaciones meteorológicas se utiliza la Tabla **11**.

Tabla 11. **Conversión de unidades para el factor de temperatura**

Unidades	Grado Celsius	Grado Fahrenheit	Kelvin
Kelvin	C = K – 273.15	F = (9(K-273.15)/5) + 32	K = K
Grado Celsius o centígrado	C = C	F = (9*C/5) + 32	K = C + 273.15
Grado Fahrenheit	C = 5(F - 32)/9	F = F	K = (5(F-32)/9)+32

Fuente: Elaboración propia de acuerdo con las conversiones del SI

Para el sensor de velocidad del viento en las variables *windSpeed y winGust*, se utiliza la Tabla **12** de conversión:

Tabla 12. **Conversión de unidades para el factor de velocidad**

Unidades	**km/h**	**m/s**	**Mph**
1 nudo (kt)	1.852	0.515	1.15

Fuente: Elaboración propia de acuerdo con las conversiones del SI

En el caso del sensor de radiación en la variable *radiation* se utilizan las conversiones en la Tabla **13**.

Tabla 13. **Conversión de unidades para el factor de radiación**

Unidades	**W/h/m^2**	**Btu/pies^2**	**Joules/m^2**
langley	11.622	3.687	41840

Fuente: Elaboración propia de acuerdo con las conversiones del SI

El sensor de precipitación con sus variables *rain*, *rainRate*, y *ET*, utiliza la conversión de la Tabla **14**.

Tabla 14. **Conversión de unidades para el factor de acumulación**

Unidades	**in/h**
mm/h	0.0394

Fuente: Elaboración propia de acuerdo con las conversiones del SI

También existen variables que no cuentan con factor de conversión por su naturaleza de lectura, en el caso de las estaciones meteorológicas las variables *inHumidity* y *outHumidity* utilizan porcentaje y las variables *heatindex* y *UV*, su unidad de medida es un índice por lo que estas variables conservan su valor en la unidad de medida recolectada en todos los casos.

Para los contaminantes se utilizan las conversiones representadas en la Tabla **15**.

Tabla 15. **Conversión de unidades para los factores de concentración de microgramos, ppm y ppb**

base	**µg/m^3**	**ppm**	**ppb**
µg/m^3	1	µg/m^3 / (float(pm) * (pow(10.3)/24.5))	(µg/m^3 / (float(pm) * (pow(10.3)/24.5)))*1000
ppm	((ppm * float(pm))/24.5) * pow(10.3)	1	ppm*1000
ppb	(((ppb/1000) * float(pm))/24.5) * pow(10.3)	ppb/1000	1

Fuente: Elaboración propia

3.8. Pruebas

Se organizaron capacitaciones con los usuarios del CECATEV, donde se les explicó el uso del *framework* en entorno web, cada uno hizo uso de este. Durante las pruebas se recolectaron los

tiempos del *framework* para realizar las tablas comparativas. Se realizó la validación de los *dataset,* a través de procedimientos manuales para verificar las conversiones de los datos, se utilizó el *dataset* con formato csv para cargarlo en R-Studio, que es su herramienta de análisis, donde se demostró que los datos contenidos en el archivo se leen correctamente y pueden generar los gráficos deseados. Finalmente se realizaron encuestas con los participantes para recabar sus experiencias al utilizar el *framework*, así como las mejoras que les gustaría se incluyeran.

4. Resultados

El contexto para realizar las pruebas constó de un repositorio de 631,202 documentos correspondientes a 6 estaciones, Estación 05, Estación 09, Estación 25, Estación 26, Estación 101 y Teledyne, distribuidos como se muestra en la Tabla **16**.

Tabla 16. **Distribución de documentos en colección archive en MongoDB**

Estación	Fecha inicial de muestras	Fecha final de muestras	Total documentos por estación
Estacion 05	6/27/19 18:10	1/15/20 01:05	**39,780**
Estacion 09	3/8/19 19:50	11/4/19 18:05	**54,467**
Estacion 25	4/3/17 20:55	1/15/20 00:40	**194,377**
Estacion 26	3/28/19 19:40	1/15/20 00:40	**79,223**
Estacion 101	9/25/18 20:10	8/28/19 18:40	**90,384**
Teledyne	9/1/18 00:00	1/15/20 01:00	**172,971**
		Total de documentos en MongoDB	**631,202**

Fuente: Elaboración propia

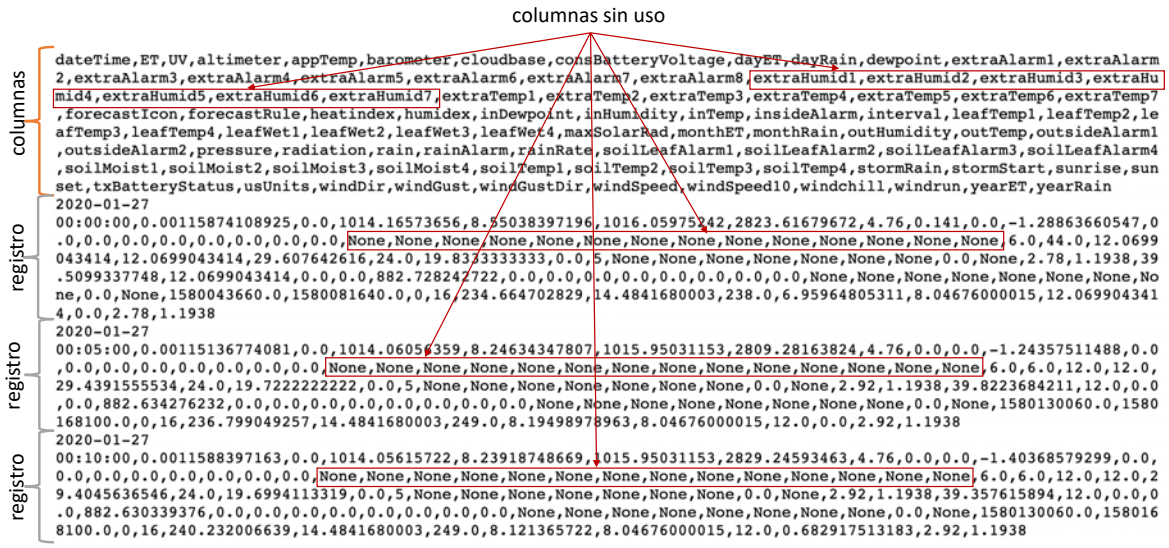
Para demostrar la implementación del *framework* en una aplicación se utilizó Django, para más información revisar el Anexo II que contiene el código fuente del *framework* y el Anexo III contiene la implementación en Django.

El personal del CECATEV, para extraer la información de las estaciones depende de cual estación va a consultar para aplicar el procedimiento correspondiente. Si la estación es meteorológica, acceden a su repositorio en la página de CECATEV, por ejemplo para la estación 101 la liga sería http://cecatev.uacj.mx/Estacion101/csv/ y la información aparece como se muestra en la Figura **42**, y en la Figura **43** se indican algunas de las columnas sin uso y que ocupan espacio.

Figura 42. **Estación 101**

Fuente: Repositorio estación 101 extraído de la liga http://cecatev.uacj.mx/Estacion101/csv/

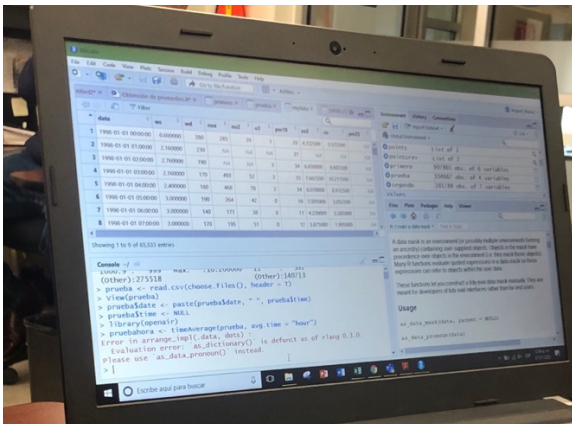
Figura 43. **Archivo csv estación 101 en el repositorio**



Fuente: Elaboración propia

De acuerdo con lo anterior, el tiempo empleado en la descarga, juntar archivos de un mes, eliminar las columnas que no les sirven, revisar que no tengan basura como guiones o algún carácter inválido, todo lo anterior en Excel y por estación, lo realizan en un tiempo aproximado de 40 a 60 minutos por *dataset*, para que les permita trabajarlo en R que es la herramienta que utilizan para graficar y analizar los datos, como se muestra en la Figura **44**.

Figura 44. **IDE RStudio**



Fuente: Elaboración propia

En el *framework*, para la misma estación 101, con todos los sensores, un mes de datos y con un proceso de homogeneización a otra unidad de medida se tardo 37.010108 segundos registrando 8,735 documentos, una vez generada la colección los tiempos de exportación a los diferentes formatos de salida fueron: 0.003999 segundos para el archivo *readmeX.txt*, 1.295802 segundos archivo en JSON, 0.462249 segundos archivo en csv y 0.900712 segundos para el archivo en XML, como se

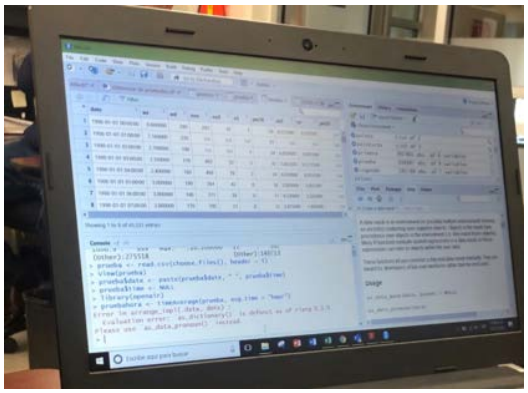
muestra en la Figura **45**. El archivo csv es analizado por el personal del laboratorio directamente en R y cambian el nombre de la columna fecha por date, además de eliminar las columnas con los nombres de las estaciones (el usuario comentó que las eliminan por que no son necesarias para este análisis, pero es correcto que existan en el archivo original) y se demostró que el archivo sube a R exitosamente, como se muestra en la Figura **46** y en la Figura **47** se muestran las gráficas realizadas con el *dataset* que se cargó en R generado por el *framework*.

Figura 45. **Tiempos ejecución Django**

```
Tiempo total grabar session: 37.01010823249817

Total documentos insertados: 8735
[28/Jan/2020 00:45:33] "GET /save HTTP/1.1" 302 0
[28/Jan/2020 00:45:33] "GET /index HTTP/1.1" 200 7731
[28/Jan/2020 00:48:43] "GET /out_dataset HTTP/1.1" 200 3796

Tiempo total generación archivo README: 0.003998994827270508


Tiempo total generación dataset JSON: 1.295802116394043

[28/Jan/2020 00:48:46] "POST /out_dataset HTTP/1.1" 302 0
[28/Jan/2020 00:48:47] "GET /index HTTP/1.1" 200 7592
[28/Jan/2020 00:48:49] "GET /out_dataset HTTP/1.1" 200 3796

Tiempo total generación archivo README: 0.0030479431152343 75


Tiempo total generación dataset CSV: 0.4622490406036377

[28/Jan/2020 00:48:51] "POST /out_dataset HTTP/1.1" 302 0
[28/Jan/2020 00:48:52] "GET /index HTTP/1.1" 200 7592
[28/Jan/2020 00:48:54] "GET /out_dataset HTTP/1.1" 200 3796

Tiempo total generación archivo README: 0.003301858901977539


Tiempo total generación dataset XML: 0.9007117748260498
```

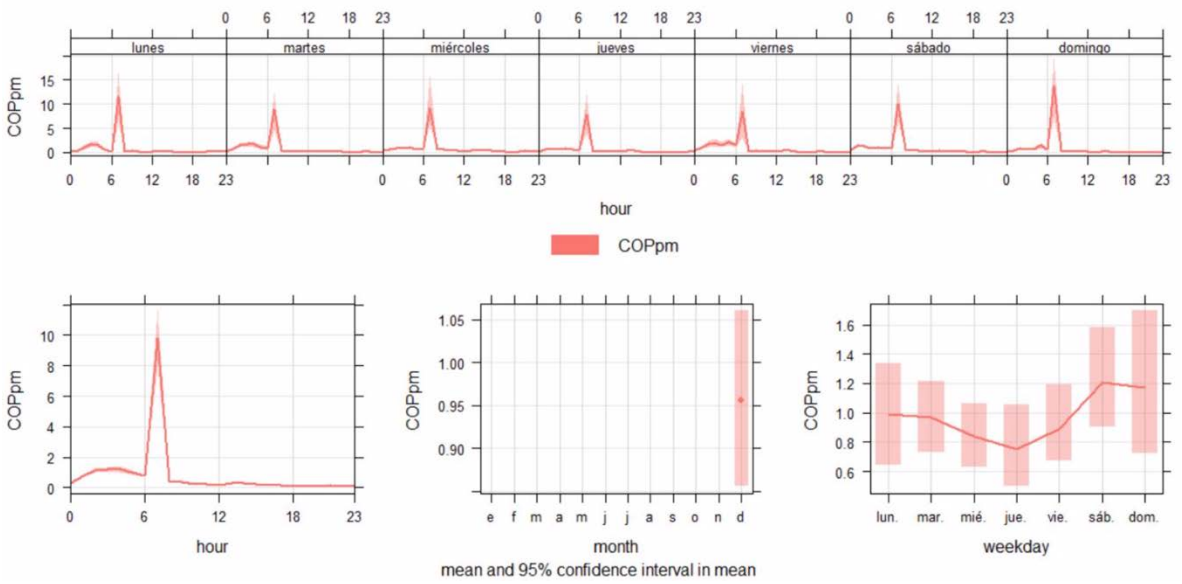
Fuente: Elaboración propia

Figura 46. **Dataset csv del *framework* en R**



Fuente: Elaboración propia

Figura 47. **Gráficas en R del *dataset* generado por el *framework***

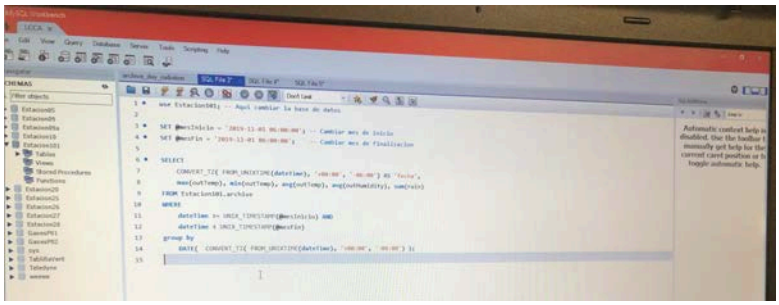


Fuente: Elaboración propia

Otro procedimiento que realizan en el laboratorio para consultar alguna estación en rango mayor a un día, es consultar directamente en SQL, sin embargo, se puede poner en riesgo la

información debido a que el personal no tiene conocimiento de lenguaje SQL y si falla el *query* no saben como arreglarlo, se anexa la pantalla de la consulta que realizan como se muestra en la Figura **48**, exportan manualmente el resultado de la consulta a un archivo con formato csv.

Figura 48. **Query en *sql* por estación**



Fuente: Elaboración propia

En comparación con la consulta en SQL, el *framework* no expone directamente la información cuidando su integridad, incluyendo la mejora de homogeneizar los datos a otra unidad en tiempos muy cortos, además no es necesario que el usuario tenga conocimiento de programación en ningún lenguaje para utilizar el *framework*.

Para extraer la información de las estaciones de gases y partículas suspendidas, el personal del laboratorio genera un archivo como el que se muestra en la Figura **49**, sin embargo deben acceder directamente al servidor dentro del *site* del CECATEV, como se muestra en la Figura **50**.

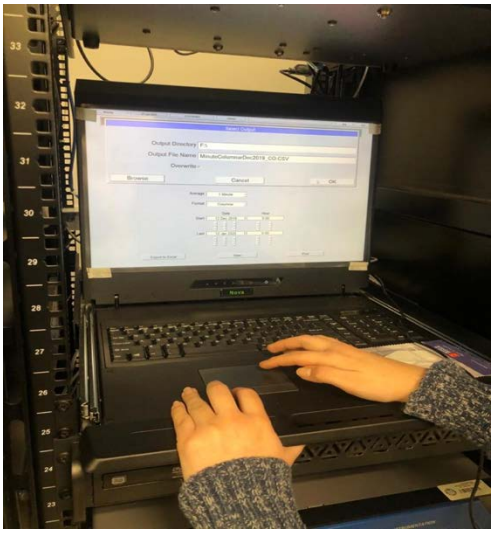
Figura 49. **Archivo csv Teledyne**

	A	B	C	D	E	F	G	H
1	Station: uluarez			Monthly Columnar Data Matrix				
2	Parameter(s): O3 NOX NO NO2 SO2 C			Short Term Averages				
3	Month / Year: January, 2019							
4								
5	Day	Hour	O3	NOX	NO	NO2	SO2	CO
6	12-Jan-19	00:00						
7	12-Jan-19	00:05						
8	12-Jan-19	00:10						
9	12-Jan-19	00:15	10.7					
10	12-Jan-19	00:20						
11	12-Jan-19	00:25		35.4	8.2	27.1	4	0.173
12	12-Jan-19	00:30		23.5	3.1	20.4	3.8	0.14
13	12-Jan-19	00:35	16.1	18.6	1.6	16.9	3.7	0.135
14	12-Jan-19	00:40	15.4	18.9	1.6	17.2	3.7	0.133
15	12-Jan-19	00:45	15.7	18.7	1.6	16.9	3.8	0.122
16	12-Jan-19	00:50	12.6	26.4	4.4	22	3.7	0.132
17	12-Jan-19	00:55	12.6	21.4	1.9	19.3	3.7	0.101
18	12-Jan-19	01:00	12.2	21.7	1.8	19.8	3.7	0.115
19	12-Jan-19	01:05	13.6	22.7	2.7	20	3.8	0.112
20	12-Jan-19	01:10	15.4	20.7	2.6	18	3.7	0.101
21	12-Jan-19	01:15	17.2	18.5	2	16.4	3.7	0.105
22	12-Jan-19	01:20	18.9	17.6	2.8	14.6	3.6	0.103
23	12-Jan-19	01:25	20.3	16	2.7	13.2	3.7	0.099
24	12-Jan-19	01:30	20.8	13.7	1.5	12	3.7	0.087
25	12-Jan-19	01:35	17.4	14.8	1.5	13.1	3.8	0.083
26	12-Jan-19	01:40	13.5	18.5	1.7	16.6	3.7	0.087
27	12-Jan-19	01:45	15.4	20.9	2.9	17.9	3.8	0.102
28	12-Jan-19	01:50	15.2	20.4	3.5	16.8	3.7	0.093
29	12-Jan-19	01:55	10.5	33.9	7.3	26.4	3.7	0.127
30	12-Jan-19	02:00	14.2	23.8	3.2	20.5	3.8	0.101
31	12-Jan-19	02:05	13.8	21.1	3.2	17.8	3.9	0.094
32	12-Jan-19	02:10	14.1	20.4	2.5	17.8	3.7	0.096
33	12-Jan-19	02:15	15.5	18.7	1.6	16.9	3.7	0.098
34	12-Jan-19	02:20	16.7	16.8	1.4	15.1	3.7	0.082
35	12-Jan-19	02:25	17.1	15.4	1.4	13.8	3.6	0.086

ShortTermColumnarJan2019_O3_NOX

Fuente: Elaboración propia

Figura 50. **Site en CECATEV**



Fuente: Elaboración propia

En la Tabla **17**, se detallan los tiempos que se emplearon en estas consultas correspondientes a las pruebas en el *site* del CECATEV para la estación Teledyne, como se puede observar los tiempos son muy cortos e inclusive se puede extraer las lecturas por minuto o cada cinco minutos, esto es

debido a que la información se obtiene directamente del servidor donde esta conectada esa estación. Sin embargo, solo este tipo de estación cuenta con esta funcionalidad y deben ingresar al *site* cada que requieren hacer estas consultas.

Tabla 17. **Estación Teledyne, pruebas *site* CECATEV**

sensores	promedio lectura	Fecha inicial	Fecha final	dias	registros	segundos	minutos
1	por minuto	2019-12-12 00:00	2020-01-12 00:00	31	44,641	5.71	0.09516667
6	por minuto	2019-12-12 00:00	2020-01-12 00:00	31	44,641	12.22	0.20366667
6	por minuto	2019-10-12 00:00	2020-01-12 00:00	92	132,481	36.26	0.60433333
6	cada 5 minutos	2019-01-12 00:00	2020-01-12 00:00	365	105,121	28.61	0.47683333

Fuente: Elaboración propia

En la Tabla **18** se muestra una serie de pruebas con diferentes estaciones, sensores y rangos de fecha, incluyendo homogeneización utilizando el *framework*. Se incluyen pruebas de la estación Teledyne. Como se observa con el *framework* se mejoran los tiempos de proceso para el CECATEV, no solo por la disminución en el tiempo de extracción de datos, si no al incluir la homogeneización de la información a unidades de medida diferentes a la que se tiene almacenada en el repositorio en MongoDB.

Tabla 18. **Tiempos de proceso de datos homogeneizados por estación con el *framework***

Proceso en framework						tiempo en segundos						tiempo total en minutos
Estación	sen	Fecha inicial	Fecha final	dias	doctos MongoDB	dataset	readme	json	csv	xml	tiempo total	
Estacion 05	14	2019-06-27 18:10	2019-10-04 18:05	98.99652778	17,358	75.3255	0.0034	1.9741	0.7102	1.1697	79.1830	1.3197
Estacion 05	14	2019-08-04 18:05	2020-01-15 01:05	163.2916667	31,936	132.5774	0.0041	3.9222	1.3321	2.6404	140.4762	2.3413
Estacion 05	14	2019-06-27 18:10	2020-01-15 01:05	201.2881944	39,780	169.1441	0.0030	4.9403	1.6928	3.2616	179.0418	2.9840
Estacion 09	19	2019-03-08 19:50	2019-06-08 19:50	92	18,980	79.3514	0.0045	2.5711	0.9402	1.7722	84.6394	1.4107
Estacion 09	19	2019-03-08 19:50	2019-09-04 18:05	179.9270833	36,914	156.2165	0.0029	5.1794	1.8674	3.2699	166.5360	2.7756
Estacion 09	19	2019-03-08 19:50	2019-11-04 18:05	240.9270833	54,467	233.9262	0.0029	7.4542	2.6587	4.8130	248.8551	4.1476
Estacion 25	19	2019-10-01 00:00	2019-11-01 00:00	31	8,929	40.2444	0.0038	1.3998	0.4805	0.8378	42.9664	0.7161
Estacion 25	19	2019-01-01 00:00	2020-01-01 00:00	365	80,353	358.0659	0.0030	11.9192	4.3548	7.7943	382.1372	6.3690
Estacion 25	**19**	**2017-04-03 20:55**	**2020-01-15 00:40**	**1016.15625**	**194,377**	**861.0886**	**0.0041**	**29.6684**	**10.6237**	**18.9564**	**920.3411**	**15.3390**
Estacion 26	14	2019-03-28 19:40	2019-06-28 19:40	92	26,460	114.6024	0.0036	3.1081	1.0885	1.9389	120.7416	2.0124
Estacion 26	14	2019-03-28 19:40	2019-10-15 00:40	200.2083333	54,119	232.4990	0.0032	6.5583	2.3220	4.1233	245.5058	4.0918
Estacion 26	14	2019-03-28 19:40	2020-01-15 00:40	292.2083333	79,223	344.7402	0.0034	9.7701	3.3503	6.2492	364.1132	6.0686
Estacion 101	19	2018-09-25 20:10	2018-12-31 20:10	97	25,682	113.9750	0.0047	3.7560	1.3825	2.4682	121.5864	2.0264
Estacion 101	19	2018-09-25 20:10	2019-05-28 18:40	244.9375	64,921	282.4692	0.0030	9.7583	3.4286	6.3382	301.9973	5.0333
Estacion 101	19	2018-09-25 20:10	2019-08-28 18:40	336.9375	90,384	399.1689	0.0039	13.3276	4.8884	8.8727	426.2614	7.1044
Teledyne	18	2018-09-01 00:00	2019-03-01 00:00	181	88,379	391.0676	0.0042	8.3926	3.4936	5.2528	408.2109	6.8035
Teledyne	18	2018-09-01 00:00	2019-08-15 01:00	348.0416667	132,427	589.4341	0.0039	12.3581	5.1125	7.1101	614.0187	10.2336
Teledyne	**18**	**2018-09-01 00:00**	**2020-01-15 01:00**	**501.0416667**	**172,971**	**757.7355**	**0.0044**	**15.9425**	**6.4051**	**9.4482**	**789.5356**	**13.1589**

Fuente: Elaboración propia

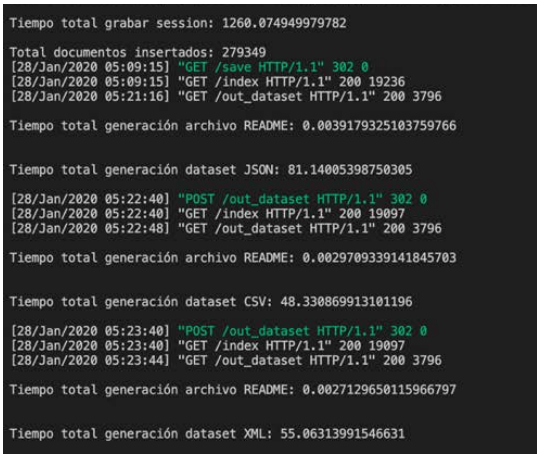
Otras pruebas que se realizaron con el *framework* incluyeron las 6 estaciones (Estación 05, Estación 09, Estación 25, Estación 26, Estación 101 y Teledyne), seleccionado el total de sensores de cada una, que dan un total de 103 sensores y utilizando conversiones en cada uno de ellos, es decir, se eligieron unidades de medida de salida diferentes a las unidades de medida en las que se encuentran almacenados los datos, en periodos mayores a 30 días, como se indica en la Tabla **19**, la última prueba que se indica en la tabla se observa que el tiempo que tardó el *framework* en procesar información mayor a 2 años fue de menos de 25 minutos, en este periodo procesó un total de 631,202 documentos almacenando 279,349 documentos ya que realiza la combinación de los datos por fecha al mismo tiempo que realiza la conversión por sensor. En la Figura **51** se observan los tiempos de ejecución del *framework*. Lo anterior representa la mitad del tiempo que emplea el personal del laboratorio en un proceso para una sola estación, cuando extrae la información del repositorio en la página del CECATEV.

Tabla 19. **Procesos de 6 estaciones 103 sensores homogeneizados en el *framework***

Proceso en framework					tiempo en segundos						tiempo total en minutos
Fecha inicial	Fecha final	dias	documentos MongoDB	documentos empotrados	dataset	readme	json	csv	xml	tiempo total	
2019-10-01 00:00	2019-11-01 00:00	31.0000	43,295	8,929	41.1790	0.0039	4.8485	1.9920	3.3504	51.3739	0.8562
2018-09-01 00:00	2019-03-05 00:00	185.0000	144,188	90,143	391.2329	0.0037	20.6551	15.2575	14.4693	441.6185	7.3603
2019-02-08 02:05	2019-11-04 00:00	268.9132	327,738	77,445	361.3077	0.0040	39.0183	16.4658	27.0040	443.7999	7.3967
2018-09-25 00:00	2019-11-04 00:00	405.0000	444,441	146,663	664.7569	0.0034	57.4077	28.5741	39.5156	790.2577	13.1710
2018-01-27 00:00	2020-01-15 01:00	718.0417	571,358	219,505	991.5286	0.0046	73.1795	39.9118	49.6220	1154.2465	19.2374
2017-04-03 20:55	**2020-01-15 01:05**	**1016.1736**	**631,202**	**279,349**	**1260.0749**	**0.0039**	**81.1401**	**48.3309**	**55.0631**	**1444.6129**	**24.0769**
	Promedio	437.3547	360,370	137,006	618.3467	0.0039	46.0415	25.0887	31.5041	720.9849	12.0164

Fuente: Elaboración propia

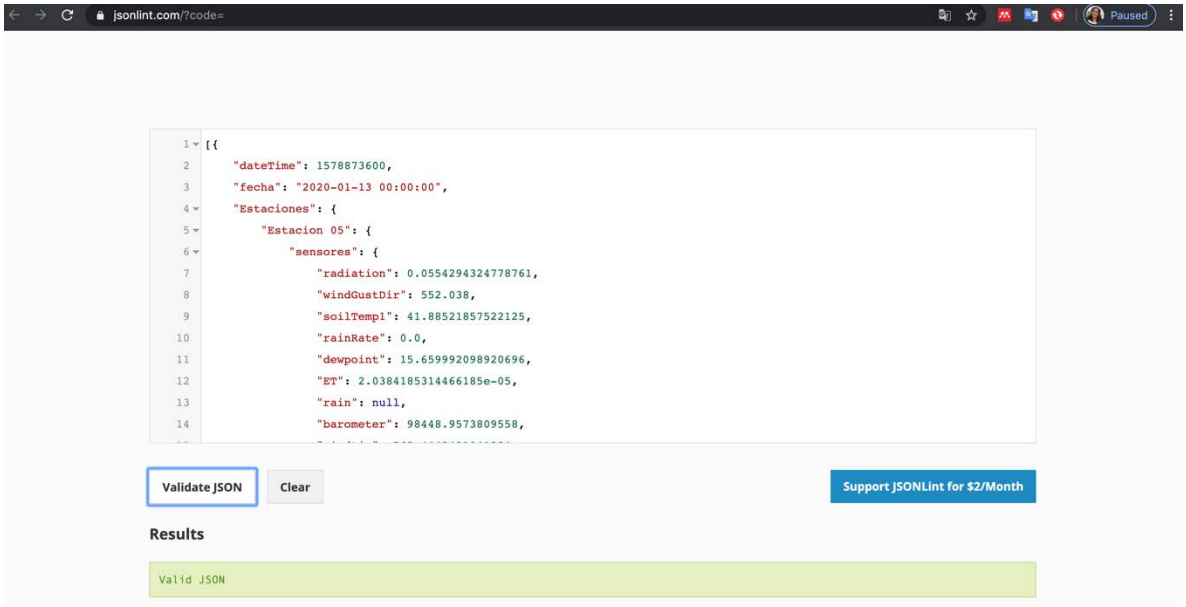
Figura 51. **Tiempos de ejecución *framework***



Fuente: Elaboración propia

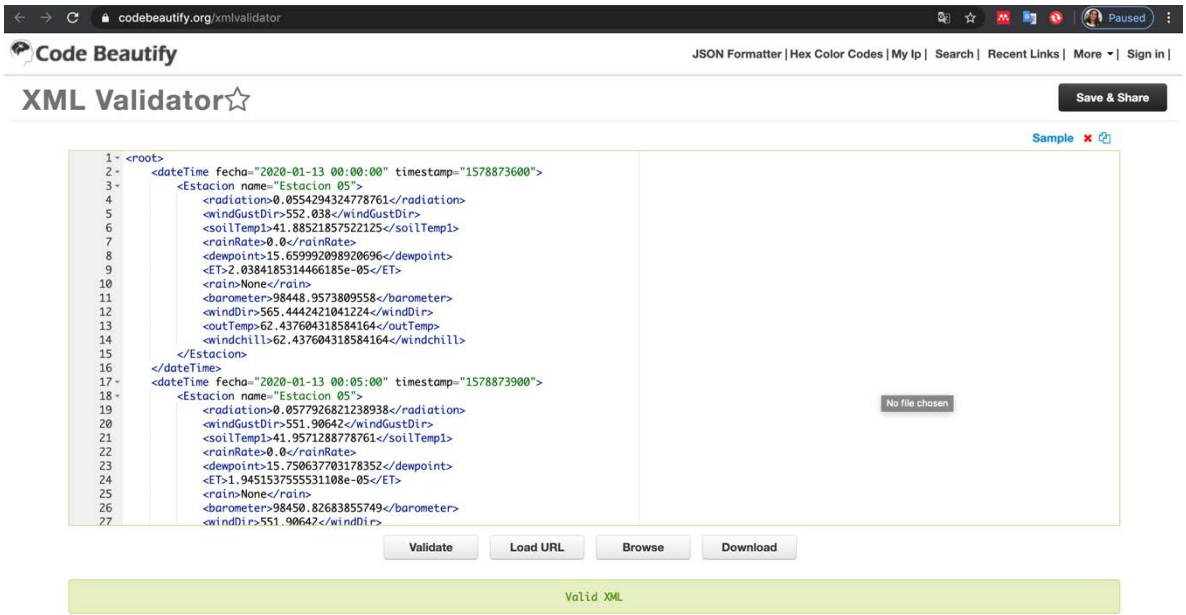
Para validar los archivos de salida en formato JSON y XML, se utilizaron herramientas en línea como se muestra en la Figura **52** y Figura **53**.

Figura 52. **Validación *dataset* en formato JSON**



Fuente: Elaboración propia

Figura 53. **Validación *dataset* en formato XML**

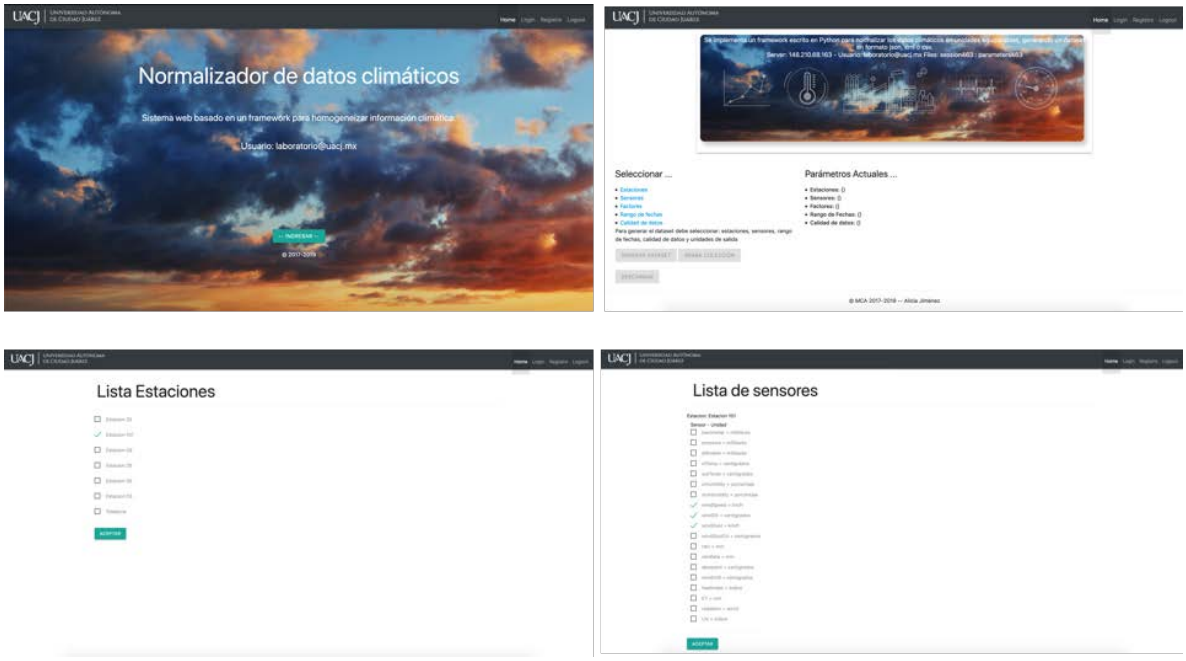


Fuente: Elaboración propia

En las siguientes figuras se observa la implementación del *framework* en Django, inicio de sesión, selección de estaciones y sensores en la Figura **54**, factores de conversión, rango de fechas,

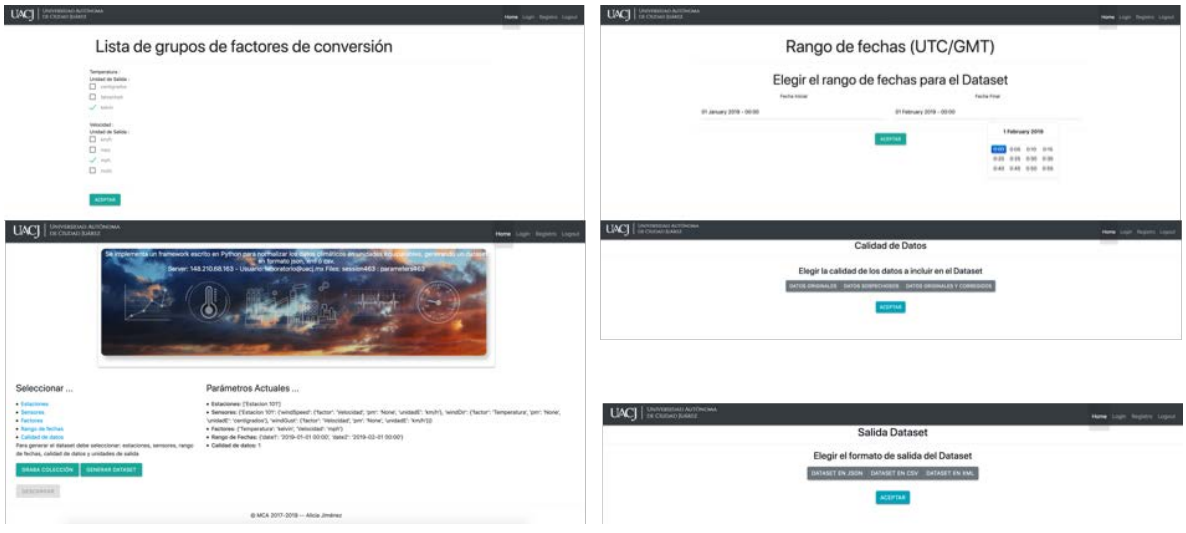
calidad de datos, formato de salida del *dataset*, así como la pantalla donde el usuario puede observar los parámetros de lo que va seleccionando, en la Figura **55**.

Figura 54. **Implementación de librerías del *framework* en Django**



Fuente: Elaboración propia

Figura 55. **Implementación de librerías del *framework* en Django**



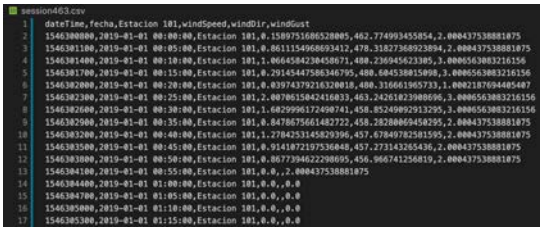
Fuente: Elaboración propia

En la Figura **56**, se muestran los ejemplos de los archivos de salida *readmeX.txt*, *sessionX.json, sessionX.csv* y *sessionX.xml*, generados en Django con la implementación del *framework*.

Figura 56. **Muestra de archivos de salida *readme.txt*, y *dataset* en JSON, csv y XML**

Archivos de salida

readmeX.txt



sessionX.csv

sessionX.json

sessionX.xml

Fuente: Elaboración propia

Se realizó una demostración del *framework* al personal del laboratorio como se muestra en la Figura **57** y se les preguntó sobre sus comentarios al respecto. Las sugerencias fueron en sentido agregar algunas mejoras a la aplicación en Django como manual de usuario o algunas ayudas visuales, sin embargo, en cuanto a los tiempos de procesamiento y calidad de los *dataset* resultantes, todos coincidieron que en definitiva el *framework* representa una mejora sustancial en el tiempo que emplean actualmente en estos procesos.

Figura 57. **Demostración *framework* CECATEV**

Fuente: Elaboración propia

5. Conclusiones

El desarrollar el *framework* formado por librerías como archivo ejecutable en Python por la línea de comando, permite implementarlo como fue demostrado en la sección de resultados en cualquier tipo de aplicación, ya sea web, aplicación móvil entre otras, por lo que la convierte en una herramienta ampliamente portable y adaptable en cualquier entorno.

Para realizar las pruebas se utilizaron datos reales correspondientes a 6 estaciones en operación con un total de 631,202 documentos en MongoDB. Y las comparaciones se realizaron contra los tiempos de consulta en un ambiente manual con personal del laboratorio.

Con los resultados anteriores se demuestra que se alcanzó el objetivo inicial de este proyecto al proporcionar soluciones eficientes. Se identificaron mejoras como disminución considerable en los tiempos de generación del *dataset*, homogeneización de la información, generación de *dataset* con unidades de medida diferentes a las originales, incluir en el *dataset* más de una estación y elegir sensores por cada una. Esta última mejora mencionada, reduce significativamente los tiempos de procesamiento ya que actualmente deben generar *dataset* por estación y combinarlos manualmente, lo que representa mayor tiempo y puede ser susceptible a errores al mezclar grandes cantidades de información. Por lo anterior se demuestra la velocidad de las bases de datos NoSQL, al permitir procesar 6 estaciones con todos sus sensores en un periodo mayor a dos años y homogeneizados en menos de 25 minutos que representa aproximadamente la mitad de lo que se tardan en procesar una sola estación con datos de un mes y sin realizar alguna conversión.

Esta herramienta además de homogeneizar información climática como en este caso, se desarrolló de tal manera que es posible adaptarla con mínimas adecuaciones para la homogeneización de diferentes repositorios de información heterogénea en un ambiente NoSQL.

A lo largo de este proyecto se identificaron elementos básicos que permitieron el desarrollo de este proyecto, como las reglas y el manejo de los metadatos en XML, para conceptualizar los perfiles de las estaciones y las características de los sensores.

También un elemento importante es el manejo de bases de datos no relacionales, en este caso MongoDB, que permiten la gestión de estructuras dinámicas para el manejo de la información, el poder almacenar documentos de diferente estructura y con datos heterogéneos en la misma colección solo se puede hacer en este tipo de bases de datos. En este proyecto, estas estructuras diferentes se pueden construir dinámicamente con los metadatos definidos en XML en conjunto con la selección de parámetros indicados por el usuario.

El uso de la técnica de minería de datos en este proyecto es otro elemento muy importante, debido que permite extraer los datos del repositorio en base a una consulta selectiva, posteriormente se analizan cada uno de los elementos del documento, se transforman, es decir, se homogeneizan y

se crea otro documento con estructura diferente que se almacena en la colección de resultados, para finalmente presentar estos datos en un *dataset*.

A pesar de que se cumplió con el objetivo de este proyecto, se identifican áreas de oportunidad que se pudieran implementar en un futuro.

Durante las pláticas con los expertos se identificaron modelos que aceptan *dataset* en otros formatos especiales, en este proyecto se incluyeron los más conocidos, sin embargo, se pueden implementar librerías que proyecten la salida de la colección resultante a estos nuevos formatos. Agregando estas opciones al *framework*.

Los metadatos definidos proporcionan información correspondiente al perfil de la estación como su geolocalización, su altura respecto a la base, por mencionar algunos, estos metadatos pudieran incluirse en el *dataset*, por medio de otra librería que le permitiera al experto seleccionar que dato quisiera agregar a su conjunto.

El proyecto utiliza el tiempo universal coordinado (UTC), sin embargo, se puede crear una librería que normalice las zonas horarias a la elección del experto.

Se ha mencionado que se pueden agregar estaciones a los metadatos para que el *framework* las incluya en las selecciones, sin embargo, faltaría construir una interfaz que permitiera la edición de estas configuraciones, con las validaciones necesarias que apoyaran en este proceso al usuario sin conocimiento en las reglas para definir XML.

También es posible explorar otros campos de aplicación para el *framework*, considerando que su construcción depende de la definición de los metadatos en el archivo XML, por lo que en este sentido si redefinen los metadatos, se puede aplicar el *framework* para la homogeneización de otro tipo de datos de acuerdo con las conversiones indicadas en el XML.

El desarrollo de este proyecto representó un crecimiento, un reto personal y profesional, el aplicar cada uno de los conocimientos adquiridos durante el curso de la maestría y este desarrollo me permitió lograr el resultado esperado, entregando un producto funcional y eficiente que se suma al esfuerzo de la universidad con proyectos en apoyo a la sociedad.

Índice de Figuras.

Figura 1. Diagrama de las diferentes variables, rutas de exposición y determinantes sociales que influyen en los impactos en la salud 10
Figura 2. Estación meteorológica Campbell Scientific 18
Figura 3. Estación Davis con *datalogger* 19
Figura 4. *Datalogger* 19
Figura 5. Sensores en una Estación Meteorológica 20
Figura 6. Climograma de B. Givoni 23
Figura 7. Cambios en la humedad relativa 23
Figura 8. Teledyne modelo T640 25
Figura 9. Distribución de partículas en el aire; medidas en micrómetros 28
Figura 10. Metadatos grupo 1 37
Figura 11. Metadatos - IP del servidor 37
Figura 12. Metadatos factores de conversión 38
Figura 13. Metadatos estaciones 38
Figura 14. Metadatos de ubicación 39
Figura 15. Metadatos de geolocalización 39
Figura 16. Metadatos de sensores 40
Figura 17. Esquema de documento JSON 40
Figura 18. Colección *users* MongoDB 42
Figura 19. Inicio de sesión y selección de estaciones en el *framework* (1,4 y 7), librería *init* y *list_stations* 42
Figura 20. Metadatos de estaciones en archivo XML 43
Figura 21. Colección *parametersX* en MongoDB 43
Figura 22. Sensores que pertenecen a la estación 25, extraído del archivo XML, librería *list_sensors* 44
Figura 23. Sensores que pertenecen a la estación 26, extraído del archivo XML 44
Figura 24. Sensores que pertenecen a la estación Teledyne, extraído del archivo XML 44
Figura 25. Metadatos de sensores de la estación 25, archivo XML 45
Figura 26. Documento para encabezados en la colección *parametersX* 45
Figura 27. Documento valores almacenado en la colección *parametersX* 46
Figura 28. Documento *Factors* en la colección *parametersX* 46
Figura 29. Documento valores almacenado en la colección *parametersX* 47

Figura 30. Selección de unidades de salida para los grupos de factor de conversión en el *framework*, librería *list_factors* 47
Figura 31. Unidades de salida elegidos para los grupos de factores 47
Figura 32. Documento Factors en la colección *parametersX* 48
Figura 33. Librería *range_date* en el *framework*, para selección de rango de fechas 48
Figura 34. Librería *quality_data* para calidad de datos 48
Figura 35. Calidad de datos almacenada en *parametersX* 49
Figura 36. Estructura documento JSON para almacenar en *session*X 49
Figura 37. Librería *switch_data framework* 50
Figura 38. Estructura formato JSON 50
Figura 39. Estructura formato csv 51
Figura 40. Estructura formato XML 51
Figura 41. Diagrama de estados del *framework* 52
Figura 42. Estación 101 56
Figura 43. Archivo csv estación 101 en el repositorio 57
Figura 44. IDE RStudio 57
Figura 45. Tiempos ejecución Django 58
Figura 46. Dataset csv del *framework* en R 58
Figura 47. Gráficas en R del *dataset* generado por el *framework* 58
Figura 48. Query en *sql* por estación 59
Figura 49. Archivo csv Teledyne 59
Figura 50. Site en CECATEV 59
Figura 51. Tiempos de ejecución *framework* 61
Figura 52. Validación *dataset* en formato JSON 62
Figura 53. Validación *dataset* en formato XML 62
Figura 54. Implementación de librerías del *framework* en Django 63
Figura 55. Implementación de librerías del *framework* en Django 63
Figura 56. Muestra de archivos de salida *readme.txt*, y *dataset* en JSON, csv y XML 64
Figura 57. Demostración *framework* CECATEV 64

Índice de Tablas.

Tabla 1. Factores de conversión para la presión. 22
Tabla 2. Factores de conversión de la temperatura 22
Tabla 3. Factor de conversión de precipitación 23
Tabla 4. Factores de conversión de velocidad del viento 24
Tabla 5. Factores de conversión de energía de radiación acumulada 24
Tabla 6. Sistema estándar de medición del índice UV incluyendo colores 25
Tabla 7. Modelos Teledyne serie T 26
Tabla 8. Tabla de pesos moleculares (PM) 28
Tabla 9. Tipo de archivo usado por weewx 32
Tabla 10. Conversión de unidades para el factor barométrico 53
Tabla 11. Conversión de unidades para el factor de temperatura 53
Tabla 12. Conversión de unidades para el factor de velocidad 54
Tabla 13. Conversión de unidades para el factor de radiación 54
Tabla 14. Conversión de unidades para el factor de acumulación 54
Tabla 15. Conversión de unidades para los factores de concentración de microgramos, ppm y ppb 54
Tabla 16. Distribución de documentos en colección archive en MongoDB 56
Tabla 17. Estación Teledyne, pruebas *site* CECATEV 60
Tabla 18. Tiempos de proceso de datos homogeneizados por estación con el *framework* 60
Tabla 19. Procesos de 6 estaciones 103 sensores homogeneizados en el *framework* 61

Referencias

[1] "Triptico LCCA." Septiembre, 2018.

[2] O. Hoegh-Guldberg, D. Jacob, and M. Taylor, "Special Report on Global Warming of 1.5 °C - Chapter 3: Impacts of 1.5º C global warming on natural and human systems," p. 243, 2018.

[3] C. F. para la P. contra R. Sanitarios, "Impactos del cambio climático en la salud," *31 de diciembre de 2017*. [Online]. Available: https://www.gob.mx/cofepris/acciones-y-programas/impactos-del-cambio-climatico-en-la-salud. [Accessed: 12-Feb-2019].

[4] P. S. Diouf, A. Boly, and S. Ndiaye, "Performance of the ETL processes in terms of volume and velocity in the cloud : state of the art."

[5] P. S. Diouf, A. Boly, and S. Ndiaye, "Variety of data in the ETL processes in the cloud : state of the art," *2018 IEEE Int. Conf. Innov. Res. Dev.*, no. May, pp. 1–5, 2018.

[6] L. Li, "A Framework Study of ETL Processes Optimization Based on Metadata Repository," pp. 125–129, 2010.

[7] M. Zhong and M. Liu, "3Se: a Semi-Structured Search Engine for Heterogeneous Data in Graph Model," *Proceeding 18th ACM Conf. Inf. Knowl. Manag. - CIKM '09*, p. 1405, 2009.

[8] Y. Jaybal, C. Ramanathan, and S. Rajagopalan, "HDSAnalytics: A Data Analytics Framework for Heterogeneous Data Sources," vol. 9, 2018.

[9] M.-S. Dao and K. Zettsu, "Discovering Environmental Impacts on Public Health Using Heterogeneous Big Sensory Data," *2015 IEEE Int. Congr. Big Data*, pp. 741–744, 2015.

[10] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han, "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," *Proc. 2014 ACM SIGMOD Int. Conf. Manag. data - SIGMOD '14*, pp. 1187–1198, 2014.

[11] R. Steinmetz, "Metadata Standards for Web-Based Resources," pp. 70–76, 2001.

[12] L. Xiang, J. Huang, X. Shao, and D. Wang, "A MongoDB-Based Management of Planar Spatial Data with a Flattened R-Tree," 2016.

[13] Q. Liu, X. Guo, H. Fan, and H. Zhu, "A novel data visualization approach and scheme for supporting heterogeneous data," *Proc. 2017 IEEE 2nd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2017*, vol. 2018-Janua, pp. 1259–1263, 2018.

[14] "Dirección de Monitoreo Atmosférico." [Online]. Available: http://www.aire.cdmx.gob.mx/default.php?opc=%27ZaBhnmI=&dc=%27Zw==. [Accessed: 05-Jun-2018].

[15] Semana de la Ciencia y la Tecnología 4o. 2004 Madrid., *Meteorología y climatología : unidad didáctica : Semana de la Ciencia y la Tecnología 2004.* Fundación Española para la Ciencia y la Tecnología, 2004.

[16] "Meteorología y climatología. ¿En qué se diferencian?" [Online]. Available: https://www.solucionesintegralesendesa.com/blog/meteorologia/meteorologia-y-climatologia/. [Accessed: 05-Jun-2018].

[17] "About Our Agency | National Oceanic and Atmospheric Administration." [Online]. Available: http://www.noaa.gov/about-our-agency. [Accessed: 05-Jun-2018].

[18] "Funciones y Objetivos." [Online]. Available: http://smn.cna.gob.mx/es/smn/funciones-y-objetivos. [Accessed: 05-Jun-2018].

[19] "México se rige bajo el Sistema Internacional de Unidades de medida | Secretaría de Economía | Gobierno | gob.mx." [Online]. Available: https://www.gob.mx/se/articulos/mexico-se-rige-bajo-el-sistema-internacional-de-unidades-de-medida. [Accessed: 02-Aug-2019].

[20] "BIPM - SI Brochure." [Online]. Available: https://www.bipm.org/en/publications/si-brochure/. [Accessed: 19-Jul-2019].

[21] C. De Diputados, D. H. Congreso De, L. A. Unión, and N. Ley, "LEY FEDERAL SOBRE METROLOGÍA Y NORMALIZACIÓN."

[22] "CENAM." [Online]. Available: http://www.cenam.mx/siu.aspx. [Accessed: 22-Jul-2019].

[23] "Tipos de estaciones meteorológicas | Guías Prácticas.COM." [Online]. Available: http://www.guiaspracticas.com/estaciones-meteorologicas/tipos-de-estaciones-meteorologicas. [Accessed: 05-Jun-2018].

[24] "Estaciones Meteorologicas." [Online]. Available: http://www.agrometeorologia.inia.gob.ve/index.php/estaciones. [Accessed: 05-Jun-2018].

[25] "Automated Weather Stations: Research-grade stations for reliable..." [Online]. Available: https://www.campbellsci.com/automated-weather-stations. [Accessed: 05-Jun-2018].

[26] D. Pro and D. Pro, "ESTACIÓN METEOROLÓGICA DAVIS INSTRUMENTS 6152 / 6162 VANTAGE PRO2," no. 571.

[27] "¿Que es un datalogger?" [Online]. Available: https://www.campbellsci.es/faqs?v=1. [Accessed: 05-Jun-2018].

[28] "Tutorial básico para el diseño de un Data Logger con Arduino y micro-SD | HeTPro." [Online]. Available: https://hetpro-store.com/tutorial-micro-sd-con-arduino/. [Accessed: 05-Jun-2018].

[29] "Partes de una Estación Meteorológica."

[30] "Elementos del clima: temperatura, humedad, presión." [Online]. Available: http://www.astromia.com/tierraluna/elemclima.htm. [Accessed: 05-Jun-2018].

[31] "Definición de presión atmosférica - Qué es, Significado y Concepto." [Online]. Available:

https://definicion.de/presion-atmosferica/. [Accessed: 05-Jun-2018].

[32] J. D. Wilson and A. J. Buffa, *Física*. Pearson Educación, 2003.

[33] ™ ™, "Vantage Pro2 Manuel de la console Bedienungsanleitung für die Konsole Manual de la consola Pour les stations météo Vantage Pro2 ™ et Vantage Pro2 Plus ™ Für Vantage Pro2 ™ und Vantage Pro2 Plus ™ Wetterstationen Para las estaciones meteorológicas Vantage ."

[34] J. D. Czajkowski, "EDIFICIOS PARA HABITACIÓN HUMANA EN CLIMAS HÚMEDOS. Evaluación y propuesta de medidas para la mitigación del cambio climático.," *Publicado en actas Jornadas Investigación FAU-UNLP, La Plata, Argentina. (Proyecto acreditado UNLP 11/U075)., CC BY-SA 2.5,*. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=2261349. [Accessed: 20-Sep-2007].

[35] P. D. Iten and P. D. Iten, *Laser doppler anemometer*. Oficina de patentes y marcas registradas de los Estados Unidos, 1976.

[36] "LA RADIACIÓN SOLAR."

[37] K. K. Hanneman, K. D. Cooper, and E. D. Baron, "Ultraviolet Immunosuppression: Mechanisms and Consequences," *Dermatol. Clin.*, vol. 24, no. 1, pp. 19–25, Jan. 2006.

[38] "Manual 3 Redes estaciones y equipos de Medición de la calidad del aiRe."

[39] "Equipos para Medición de Partículas - Sanambiente." [Online]. Available: https://www.sanambiente.com.co/index.php/es/equipos-para-medicion-de-particulas. [Accessed: 27-Aug-2019].

[40] "Teledyne API Products." [Online]. Available: http://www.teledyne-api.com/products. [Accessed: 13-Jan-2020].

[41] "NORMA Oficial Mexicana NOM-022-SSA1-2010, Salud ambiental." [Online]. Available: http://www.dof.gob.mx/normasOficiales/4149/salud1/salud1.htm. [Accessed: 15-Jan-2020].

[42] "(No Title)." [Online]. Available: https://sinaica.inecc.gob.mx/archivo/noms/NOM 021 SSA 1993 CO.pdf. [Accessed: 15-Jan-2020].

[43] "¿Qué es el CO2? - Conciencia Eco." [Online]. Available: https://www.concienciaeco.com/2012/02/20/que-es-el-co2/. [Accessed: 05-Jun-2018].

[44] "NOM-023-SSA1-1993."

[45] "Ozono - EcuRed." [Online]. Available: https://www.ecured.cu/Ozono. [Accessed: 05-Jun-2018].

[46] J. Pey, B. Investigador Ramón, C. Igme, and U. De Zaragoza, "PARTÍCULAS ATMOSFÉRICAS: COMPOSICIÓN, ORIGEN Y EFECTOS."

[47] I. E. Roberts Alley & Associates, *Manual de control de la calidad del aire*. 2001.

[48] J. J. Universidad Pedagógica y Tecnológica de Colombia. Facultad de Ingeniería. Centro de Estudios y Educación Contínua., J. F. Camargo-Ortega, and L. Joyanes-Aguilar, *Revista FI-UPTC : publicación del Centro de Estudios y Educación Continua de la Facultad de Ingeniería, CEDEC.*, vol. 24, no. 38. Universidad Pedagógica y Tecnológica de Colombia, 2015.

[49] M. Chen, S. Mao, and Y. Liu, "Big Data : A Survey," pp. 171–209, 2014.

[50] W. Swapnil and Y. Anil, "Big Data: Characteristics, Challenges and Data Mining," *Int. J. Comput. Appl.*, pp. 975–8887.

[51] C. Arcila-Calderón, E. Barbosa-Caro, and F. Cabezuelo-Lorenzo, "TÉCNICAS BIG DATA: ANÁLISIS DE TEXTOS A GRAN ESCALA PARA LA INVESTIGACIÓN CIENTÍFICA Y PERIODÍSTICA Big data techniques: Large-scale text analysis for scientific and journalistic research," vol. 25, no. 4, pp. 1699–2407, 2016.

[52] J. A. Senso, "El concepto de metadato. Algo más que descripción de El concepto de metadato. Algo más que descripción de recursos electrónicos recursos electrónicos Antonio de la Rosa Piñero Antonio de la Rosa Piñero."

[53] "JSON." [Online]. Available: https://www.json.org/json-en.html. [Accessed: 17-Jan-2020].

[54] "Welcome to Python.org." [Online]. Available: https://www.python.org/. [Accessed: 25-Jan-2020].

[55] "The Web framework for perfectionists with deadlines | Django." [Online]. Available: https://www.djangoproject.com/. [Accessed: 25-Jan-2020].

[56] "Introduction to Node.js." [Online]. Available: https://nodejs.dev/. [Accessed: 25-Jan-2020].

[57] "Express - Node.js web application framework." [Online]. Available: http://expressjs.com/. [Accessed: 25-Jan-2020].

[58] "Flask - Full Stack Python." [Online]. Available: https://www.fullstackpython.com/flask.html. [Accessed: 25-Jan-2020].

[59] "WeeWX: open source weather software." [Online]. Available: http://www.weewx.com/. [Accessed: 25-Jan-2020].

[60] "WeeWX: Customization Guide." [Online]. Available: http://www.weewx.com/docs/customizing.htm. [Accessed: 26-Jan-2020].

[61] "La base de datos líder del mercado para aplicaciones modernas | MongoDB." [Online]. Available: https://www.mongodb.com/es. [Accessed: 16-Jan-2020].

Anexo I – Tablas software *weewx*.

En este anexo se muestran las tablas del software *weewx*, para las estaciones Teledyne y Estación 26, en la columna de sensor se marcan los sensores que recolectan lecturas en esas estaciones, por lo que los otros campos no se utilizan.

Estación 26	Valor	Sensor	Teledyne	Valor	Sensor
dateTime	1553802300	*	**dateTime**	1535760000	*
usUnits	16	*	**usUnits**	16	*
interval	5	*	**interval**	5	*
barometer		*	**barometer**		
pressure			**pressure**		
altimeter			**altimeter**		
inTemp			**inTemp**		
outTemp	27.5945133	*	**outTemp**		
inHumidity			**inHumidity**		
outHumidity	10.2061062	*	**outHumidity**		
windSpeed	15.0336747	*	**windSpeed**		
windDir	218.835312	*	**windDir**		
windGust	15.0336747	*	**windGust**		
windGustDir	220.4	*	**windGustDir**		
rainRate	0	*	**rainRate**		
rain	0	*	**rain**		
dewpoint	-6.5048207	*	**dewpoint**		
windchill	27.5945133	*	**windchill**		
heatindex	27.5945133	*	**heatindex**		
ET	0.00196954	*	**ET**		
radiation	0.94471681	*	**radiation**		
UV			**UV**		
extraTemp1			**extraTemp1**		
extraTemp2			**extraTemp2**		
extraTemp3			**extraTemp3**		
soilTemp1		*	**soilTemp1**		
soilTemp2			**soilTemp2**		
soilTemp3			**soilTemp3**		
soilTemp4			**soilTemp4**		
leafTemp1			**leafTemp1**		
leafTemp2			**leafTemp2**		
extraHumid1			**extraHumid1**		
extraHumid2			**extraHumid2**		
soilMoist1			**soilMoist1**		
soilMoist2			**soilMoist2**		
soilMoist3			**soilMoist3**		
soilMoist4			**soilMoist4**		
leafWet1			**leafWet1**		
leafWet2			**leafWet2**		
rxCheckPercent			**rxCheckPercent**		
txBatteryStatus			**txBatteryStatus**		

consBatteryVoltage			**consBatteryVoltage**		
hail			**hail**		
hailRate			**hailRate**		
heatingTemp			**heatingTemp**		
heatingVoltage			**heatingVoltage**		
supplyVoltage			**supplyVoltage**		
referenceVoltage			**referenceVoltage**		
windBatteryStatus			**windBatteryStatus**		
rainBatteryStatus			**rainBatteryStatus**		
outTempBatteryStatus			**outTempBatteryStatus**		
inTempBatteryStatus			**inTempBatteryStatus**		
			PM10		*
			PM2_5		*
			PM10_2_5		*
			PM10_1hr		*
			PM2_5_1hr		*
			PM10_2_5_1hr		*
			PM10_12hr		*
			PM2_5_12hr		*
			PM10_2_5_12hr		*
			PM10_24hr		*
			PM2_5_24hr		*
			PM10_2_5_24hr		*
			SO2Ppm	1	*
			NOxPpm	32.5	*
			NO2Ppm	27.3	*
			NOPpm	5.1	*
			CO2		*
			CO2Ppm		
			O3		*
			O3Ppm		
			CO		
			COPpm		
			particlesDensity		
			particlesPpmperfcf		

Anexo II – Metadatos archivo XML.

Metadatos disponibles en el archivo EstacionX3.XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Configuraciones>
 <Servidor>148.210.68.163</Servidor>
 <Factores>
  <Factor nombre="Barometrico">
   <unit>milibares</unit>
   <unit>Pa</unit>
   <unit>Atm</unit>
   <unit>inHg</unit>
   <unit>hPa</unit>
   <unit>mmHg</unit>
   <unit>Psi</unit>
   <unit>Torr</unit>
  </Factor>
  <Factor nombre="Temperatura">
   <unit>centigrados</unit>
   <unit>fahrenheit</unit>
   <unit>kelvin</unit>
  </Factor>
  <Factor nombre="Velocidad">
   <unit>km/h</unit>
   <unit>mps</unit>
   <unit>mph</unit>
   <unit>nudo</unit>
  </Factor>
  <Factor nombre="Radiacion">
   <unit>w/m2</unit>
   <unit>langley</unit>
   <unit>btu/ft2</unit>
   <unit>joules/m2</unit>
  </Factor>
  <Factor nombre="Acumulacion">
   <unit>mm</unit>
   <unit>in</unit>
  </Factor>
  <Factor nombre="Acumulacion hora">
   <unit>mm/h</unit>
   <unit>in/h</unit>
  </Factor>
  <Factor nombre="Concentracion ppm">
   <unit>ppm</unit>
   <unit>ppb</unit>
   <unit>ug/m3</unit>
  </Factor>
  <Factor nombre="Concentracion ppb">
   <unit>ppb</unit>
   <unit>ppm</unit>
   <unit>ug/m3</unit>
  </Factor>
 </Factores>
 <Estaciones>
  <Estacion nombre="Estacion 25">  <!-- Davis -->
   <nombre>Estacion 25</nombre>
   <sistemamt>16</sistemamt>
   <intervalo>5</intervalo>
```

```
<altura>1.50</altura>
<tablasql>weewx25</tablasql>
<url>url</url>
<ubicacion>
 <descripcion>Bomberos Anapra</descripcion>
 <domicilio>Puerto de Anapra, 32107</domicilio>
 <municipio>Juarez</municipio>
 <estado>Chihuahua</estado>
</ubicacion>
<geolocalizacion>
 <latitud>31.773333333333333</latitud>
 <longitud>-106.5611111111</longitud>
 <altitud>1193 msnm</altitud>
</geolocalizacion>
<marca>Davis</marca>
<modelo>Vantage Pro2</modelo>
<tipo>Meteorologica</tipo>
<fechains>Sep-07</fechains>
<fechaact>xfechaA</fechaact>
<datalogger>ISS</datalogger>
<sw>WaterLink</sw>
<caracteristicas>xcaracteristicas</caracteristicas>
<imagen>ximagen</imagen>
<sensores>
 <barometer>
  <unidades>milibares</unidades>
  <minimo>540</minimo>
  <maximo>1100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>barometer</sql>
  <columna>dv25baro</columna>
  <Factor>Barometrico</Factor>
 </barometer>
 <pressure>
  <unidades>milibares</unidades>
  <minimo>540</minimo>
  <maximo>1100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>pressure</sql>
  <columna>dv25pres</columna>
  <Factor>Barometrico</Factor>
 </pressure>
 <altimeter>
  <unidades>milibares</unidades>
  <minimo>540</minimo>
  <maximo>1100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>altimeter</sql>
  <columna>dv25alti</columna>
  <Factor>Barometrico</Factor>
 </altimeter>
 <inTemp>
  <unidades>centigrados</unidades>
  <minimo>-40</minimo>
  <maximo>65</maximo>
  <marca>Davis</marca>
```

```
    <modelo>Vantage Pro2</modelo>
    <sql>inTemp</sql>
    <columna>dv25inte</columna>
    <Factor>Temperatura</Factor>
   </inTemp>
   <outTemp>
    <unidades>centigrados</unidades>
    <minimo>-40</minimo>
    <maximo>65</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>outTemp</sql>
    <columna>dv25outte</columna>
    <Factor>Temperatura</Factor>
   </outTemp>
   <inHumidity>
    <unidades>porcentaje</unidades>
    <minimo>0</minimo>
    <maximo>100</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>inHumidity</sql>
    <columna>dv25inhum</columna>
  </inHumidity>
   <outHumidity>
    <unidades>porcentaje</unidades>
    <minimo>0</minimo>
    <maximo>100</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>outHumidity</sql>
    <columna>dv25outhum</columna>
   </outHumidity>
   <windSpeed>
    <unidades>km/h</unidades>
    <minimo>0</minimo>
    <maximo>320</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>windSpeed</sql>
    <columna>dv25windsp</columna>
    <Factor>Velocidad</Factor>
   </windSpeed>
   <windDir>
    <unidades>centigrados</unidades>
    <minimo>0</minimo>
    <maximo>360</maximo>
    <marca>Davis</marca>
    <modelo>6410 Anenometro</modelo>
    <sql>windDir</sql>
    <columna>dv25windir</columna>
    <Factor>Temperatura</Factor>
   </windDir>
   <windGust>
    <unidades>km/h</unidades>
    <minimo>0</minimo>
    <maximo>320</maximo>
    <marca>Davis</marca>
    <modelo>6410 Anenometro</modelo>
```

```
  <sql>windGust</sql>
  <columna>dv25windgus</columna>
  <Factor>Velocidad</Factor>
</windGust>
<windGustDir>
  <unidades>centigrados</unidades>
  <minimo>0</minimo>
  <maximo>360</maximo>
  <marca>Davis</marca>
  <modelo>6410 Anenometro</modelo>
  <sql>windGustDir</sql>
  <columna>dv25windgusd</columna>
  <Factor>Temperatura</Factor>
</windGustDir>
<rain>
  <unidades>mm</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>rain</sql>
  <columna>dv25rain</columna>
  <Factor>Acumulacion</Factor>
</rain>
<rainRate>
  <unidades>mm</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>rainRate</sql>
  <columna>dv25rainr</columna>
  <Factor>Acumulacion</Factor>
</rainRate>
<dewpoint>
  <unidades>centigrados</unidades>
  <minimo>-50</minimo>
  <maximo>60</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>dewpoint</sql>
  <columna>dv25dew</columna>
  <Factor>Temperatura</Factor>
</dewpoint>
<windchill>
  <unidades>centigrados</unidades>
  <minimo>-79</minimo>
  <maximo>57</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>windchill</sql>
  <columna>dv25windc</columna>
  <Factor>Temperatura</Factor>
</windchill>
<heatindex>
  <unidades>indice</unidades>
  <minimo>-79</minimo>
  <maximo>57</maximo>
  <marca>Davis</marca>
```

```
    <modelo>Vantage Pro2</modelo>
    <sql>heatindex</sql>
    <columna>dv25heati</columna>
   </heatindex>
   <ET>
    <unidades>mm</unidades>
    <minimo>0</minimo>
    <maximo>99.9</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>ET</sql>
    <columna>dv25et</columna>
    <Factor>Acumulacion</Factor>
   </ET>
   <radiation>
    <unidades>w/m2</unidades>
    <minimo>0</minimo>
    <maximo>1800</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>radiation</sql>
    <columna>dv25radia</columna>
    <Factor>Radiacion</Factor>
   </radiation>
   <UV>
    <unidades>indice</unidades>
    <minimo>0</minimo>
    <maximo>16</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>UV</sql>
    <columna>dv25uv</columna>
   </UV>
  </sensores>
 </Estacion>
 <Estacion nombre="Estacion 101"> <!-- Davis -->
  <nombre>Estacion 101</nombre>
  <sistemamt>16</sistemamt>
  <intervalo>5</intervalo>
  <altura>1.50</altura>
  <tablasql>weewx101</tablasql>
  <url>url</url>
  <ubicacion>
   <descripcion>IIT 01</descripcion>
   <domicilio>Av del Charro 450</domicilio>
   <municipio>Juarez</municipio>
   <estado>Chihuahua</estado>
  </ubicacion>
  <geolocalizacion>
   <latitud>31.743333333333336</latitud>
   <longitud>-106.43166666666667</longitud>
   <altitud>1127 msnm</altitud>
  </geolocalizacion>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <tipo>Meteorologica</tipo>
  <fechains>Oct-07</fechains>
  <fechaact>xfechaA</fechaact>
  <datalogger>ISS</datalogger>
```

```
<sw>WaterLink</sw>
<caracteristicas>xcaracteristicas</caracteristicas>
<imagen>ximagen</imagen>
<sensores>
 <barometer>
  <unidades>milibares</unidades>
  <minimo>540</minimo>
  <maximo>1100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>barometer</sql>
  <columna>dv101baro</columna>
  <Factor>Barometrico</Factor>
 </barometer>
 <pressure>
  <unidades>milibares</unidades>
  <minimo>540</minimo>
  <maximo>1100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>pressure</sql>
  <columna>dv101pres</columna>
  <Factor>Barometrico</Factor>
 </pressure>
 <altimeter>
  <unidades>milibares</unidades>
  <minimo>540</minimo>
  <maximo>1100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>altimeter</sql>
  <columna>dv101alti</columna>
  <Factor>Barometrico</Factor>
 </altimeter>
 <inTemp>
  <unidades>centigrados</unidades>
  <minimo>-40</minimo>
  <maximo>65</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>inTemp</sql>
  <columna>dv101inte</columna>
  <Factor>Temperatura</Factor>
 </inTemp>
 <outTemp>
  <unidades>centigrados</unidades>
  <minimo>-40</minimo>
  <maximo>65</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>outTemp</sql>
  <columna>dv101outte</columna>
  <Factor>Temperatura</Factor>
 </outTemp>
 <inHumidity>
  <unidades>porcentaje</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
```

```
  <modelo>Vantage Pro2</modelo>
  <sql>inHumidity</sql>
  <columna>dv101inhum</columna>
</inHumidity>
<outHumidity>
  <unidades>porcentaje</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>outHumidity</sql>
  <columna>dv101outhum</columna>
</outHumidity>
<windSpeed>
  <unidades>km/h</unidades>
  <minimo>0</minimo>
  <maximo>320</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>windSpeed</sql>
  <columna>dv101windsp</columna>
  <Factor>Velocidad</Factor>
</windSpeed>
<windDir>
  <unidades>centigrados</unidades>
  <minimo>0</minimo>
  <maximo>360</maximo>
  <marca>Davis</marca>
  <modelo>6410 Anenometro</modelo>
  <sql>windDir</sql>
  <columna>dv101windir</columna>
  <Factor>Temperatura</Factor>
</windDir>
<windGust>
  <unidades>km/h</unidades>
  <minimo>0</minimo>
  <maximo>320</maximo>
  <marca>Davis</marca>
  <modelo>6410 Anenometro</modelo>
  <sql>windGust</sql>
  <columna>dv101windgus</columna>
  <Factor>Velocidad</Factor>
</windGust>
<windGustDir>
  <unidades>centigrados</unidades>
  <minimo>0</minimo>
  <maximo>360</maximo>
  <marca>Davis</marca>
  <modelo>6410 Anenometro</modelo>
  <sql>windGustDir</sql>
  <columna>dv101windgusd</columna>
  <Factor>Temperatura</Factor>
</windGustDir>
<rain>
  <unidades>mm</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
```

```
  <sql>rain</sql>
  <columna>dv101rain</columna>
  <Factor>Acumulacion</Factor>
</rain>
<rainRate>
  <unidades>mm</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>rainRate</sql>
  <columna>dv101rainr</columna>
  <Factor>Acumulacion</Factor>
</rainRate>
<dewpoint>
  <unidades>centigrados</unidades>
  <minimo>-50</minimo>
  <maximo>60</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>dewpoint</sql>
  <columna>dv101dew</columna>
  <Factor>Temperatura</Factor>
</dewpoint>
<windchill>
  <unidades>centigrados</unidades>
  <minimo>-79</minimo>
  <maximo>57</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>windchill</sql>
  <columna>dv101windc</columna>
  <Factor>Temperatura</Factor>
</windchill>
<heatindex>
  <unidades>indice</unidades>
  <minimo>-79</minimo>
  <maximo>57</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>heatindex</sql>
  <columna>dv101heati</columna>
</heatindex>
<ET>
  <unidades>mm</unidades>
  <minimo>0</minimo>
  <maximo>99.9</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>ET</sql>
  <columna>dv101et</columna>
  <Factor>Acumulacion</Factor>
</ET>
<radiation>
  <unidades>w/m2</unidades>
  <minimo>0</minimo>
  <maximo>1800</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
```

```
    <sql>radiation</sql>
    <columna>dv101radia</columna>
    <Factor>Radiacion</Factor>
   </radiation>
   <UV>
    <unidades>indice</unidades>
    <minimo>0</minimo>
    <maximo>16</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>UV</sql>
    <columna>dv101uv</columna>
   </UV>
 </sensores>
</Estacion>
<Estacion nombre="Estacion 09"> <!-- Davis -->
 <nombre>Estacion 09</nombre>
 <sistemamt>16</sistemamt>
 <intervalo>5</intervalo>
 <altura>1.50</altura>
 <tablasql>weewx09</tablasql>
 <url>url</url>
 <ubicacion>
  <descripcion>Babicora</descripcion>
  <domicilio>Av Valle del Cedro</domicilio>
  <municipio>Juarez</municipio>
  <estado>Chihuahua</estado>
 </ubicacion>
 <geolocalizacion>
  <latitud>31.645833333333332</latitud>
  <longitud>-106.38444444444445</longitud>
  <altitud>1140 msnm</altitud>
 </geolocalizacion>
 <marca>Davis</marca>
 <modelo>Vantage Pro2</modelo>
 <tipo>Meteorologica</tipo>
 <fechains>Nov-07</fechains>
 <fechaact>xfechaA</fechaact>
 <datalogger>ISS</datalogger>
 <sw>WaterLink</sw>
 <caracteristicas>xcaracteristicas</caracteristicas>
 <imagen>ximagen</imagen>
 <sensores>
  <barometer>
   <unidades>milibares</unidades>
   <minimo>540</minimo>
   <maximo>1100</maximo>
   <marca>Davis</marca>
   <modelo>Vantage Pro2</modelo>
   <sql>barometer</sql>
   <columna>dv09baro</columna>
   <Factor>Barometrico</Factor>
  </barometer>
  <pressure>
   <unidades>milibares</unidades>
   <minimo>540</minimo>
   <maximo>1100</maximo>
   <marca>Davis</marca>
   <modelo>Vantage Pro2</modelo>
```

```
  <sql>pressure</sql>
  <columna>dv09pres</columna>
  <Factor>Barometrico</Factor>
 </pressure>
 <altimeter>
  <unidades>milibares</unidades>
  <minimo>540</minimo>
  <maximo>1100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>altimeter</sql>
  <columna>dv09alti</columna>
  <Factor>Barometrico</Factor>
 </altimeter>
 <inTemp>
  <unidades>centigrados</unidades>
  <minimo>-40</minimo>
  <maximo>65</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>inTemp</sql>
  <columna>dv09inte</columna>
 </inTemp>
 <outTemp>
  <unidades>centigrados</unidades>
  <minimo>-40</minimo>
  <maximo>65</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>outTemp</sql>
  <columna>dv09outte</columna>
  <Factor>Temperatura</Factor>
 </outTemp>
 <inHumidity>
  <unidades>porcentaje</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>inHumidity</sql>
  <columna>dv09inhum</columna>
 </inHumidity>
 <outHumidity>
  <unidades>porcentaje</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>outHumidity</sql>
  <columna>dv09outhum</columna>
 </outHumidity>
 <windSpeed>
  <unidades>km/h</unidades>
  <minimo>0</minimo>
  <maximo>320</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>windSpeed</sql>
  <columna>dv09windsp</columna>
```

```
  <Factor>Velocidad</Factor>
</windSpeed>
<windDir>
  <unidades>centigrados</unidades>
  <minimo>0</minimo>
  <maximo>360</maximo>
  <marca>Davis</marca>
  <modelo>6410 Anenometro</modelo>
  <sql>windDir</sql>
  <columna>dv09windir</columna>
  <Factor>Temperatura</Factor>
</windDir>
<windGust>
  <unidades>km/h</unidades>
  <minimo>0</minimo>
  <maximo>320</maximo>
  <marca>Davis</marca>
  <modelo>6410 Anenometro</modelo>
  <sql>windGust</sql>
  <columna>dv09windgus</columna>
  <Factor>Velocidad</Factor>
</windGust>
<windGustDir>
  <unidades>centigrados</unidades>
  <minimo>0</minimo>
  <maximo>360</maximo>
  <marca>Davis</marca>
  <modelo>6410 Anenometro</modelo>
  <sql>windGustDir</sql>
  <columna>dv09windgusd</columna>
  <Factor>Temperatura</Factor>
</windGustDir>
<rain>
  <unidades>mm</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>rain</sql>
  <columna>dv09rain</columna>
  <Factor>Acumulacion</Factor>
</rain>
<rainRate>
  <unidades>mm</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>rainRate</sql>
  <columna>dv09rainr</columna>
  <Factor>Acumulacion</Factor>
</rainRate>
<dewpoint>
  <unidades>centigrados</unidades>
  <minimo>-50</minimo>
  <maximo>60</maximo>
  <marca>Davis</marca>
  <modelo>Vantage Pro2</modelo>
  <sql>dewpoint</sql>
```

```
    <columna>dv09dew</columna>
    <Factor>Temperatura</Factor>
   </dewpoint>
   <windchill>
    <unidades>centigrados</unidades>
    <minimo>-79</minimo>
    <maximo>57</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>windchill</sql>
    <columna>dv09windc</columna>
    <Factor>Temperatura</Factor>
   </windchill>
   <heatindex>
    <unidades>indice</unidades>
    <minimo>-79</minimo>
    <maximo>54</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>heatindex</sql>
    <columna>dv09heati</columna>
   </heatindex>
   <ET>
    <unidades>mm</unidades>
    <minimo>0</minimo>
    <maximo>99.9</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>ET</sql>
    <columna>dv09et</columna>
    <Factor>Acumulacion</Factor>
   </ET>
   <radiation>
    <unidades>w/m2</unidades>
    <minimo>0</minimo>
    <maximo>1800</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro2</modelo>
    <sql>radiation</sql>
    <columna>dv09radia</columna>
    <Factor>Radiacion</Factor>
   </radiation>
   <UV>
    <unidades>indice</unidades>
    <minimo>0</minimo>
    <maximo>16</maximo>
    <marca>Davis</marca>
    <modelo>Vantage Pro II</modelo>
    <sql>UV</sql>
    <columna>dv09uv</columna>
   </UV>
  </sensores>
 </Estacion>
 <Estacion nombre="Estacion 26">  <!-- Campbell - Meteorologica -->
  <nombre>Estacion 26</nombre>
  <sistemamt>16</sistemamt>
  <intervalo>1</intervalo>
  <altura>1.50</altura>
  <tablasql>weewx26</tablasql>
```

```
<url>url</url>
<ubicacion>
  <descripcion>Clinica nutricion</descripcion>
  <domicilio>Calle Batalla de Zacatecas</domicilio>
  <municipio>Juarez</municipio>
  <estado>Chihuahua</estado>
</ubicacion>
<geolocalizacion>
  <latitud>31.654999999999998</latitud>
  <longitud>-106.46611111111112</longitud>
  <altitud>1217 msnm</altitud>
</geolocalizacion>
<marca>Campbell Scientific</marca>
<modelo>xmodelo</modelo>
<tipo>Meteorologica</tipo>
<fechains>Ene-09</fechains>
<fechaact>xfechaA</fechaact>
<datalogger>CR1000</datalogger>
<sw>LoggerNet</sw>
<caracteristicas>xcaracteristicas</caracteristicas>
<imagen>ximagen</imagen>
<sensores>
  <barometer>
    <unidades>milibares</unidades>
    <minimo>500</minimo>
    <maximo>1100</maximo>
    <marca>Campbell Scientific</marca>
    <modelo>CS100</modelo>
    <sql>barometer</sql>
    <columna>cb26baro</columna>
    <Factor>Barometrico</Factor>
  </barometer>
  <outTemp>
    <unidades>centigrados</unidades>
    <minimo>-40</minimo>
    <maximo>60</maximo>
    <marca>Campbell Scientific</marca>
    <modelo>HC2S3</modelo>
    <sql>outTemp</sql>
    <columna>cb26outte</columna>
    <Factor>Temperatura</Factor>
  </outTemp>
  <outHumidity>
    <unidades>porcentaje</unidades>
    <minimo>0</minimo>
    <maximo>100</maximo>
    <marca>Campbell Scientific</marca>
    <modelo>HC2S3</modelo>
    <sql>outHumidity</sql>
    <columna>cb26outhum</columna>
  </outHumidity>
  <windSpeed>
    <unidades>m/s</unidades>
    <minimo>0</minimo>
    <maximo>100</maximo>
    <marca>Campbell Scientific</marca>
    <modelo>05108</modelo>
    <sql>windSpeed</sql>
    <columna>cb26windsp</columna>
```

```
</windSpeed>
<windDir>
 <unidades>centigrados</unidades>
 <minimo>0</minimo>
 <maximo>360</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>05108</modelo>
 <sql>windDir</sql>
 <columna>cb26windir</columna>
 <Factor>Temperatura</Factor>
</windDir>
<windGust>
 <unidades>m/s</unidades>
 <minimo>0</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>05108</modelo>
 <sql>windGust</sql>
 <columna>cb26windgus</columna>
</windGust>
<windGustDir>
 <unidades>centigrados</unidades>
 <minimo>0</minimo>
 <maximo>360</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>05108</modelo>
 <sql>windGustDir</sql>
 <columna>cb26windgusd</columna>
 <Factor>Temperatura</Factor>
</windGustDir>
<rainRate>
 <unidades>mm/h</unidades>
 <minimo>0</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>TE525MM</modelo>
 <sql>rainRate</sql>
 <columna>cb26rainr</columna>
 <Factor>Acumulacion hora</Factor>
</rainRate>
<rain>  <!-- se reinicia el sensor en 24 horas (es el valor de las 14 horas)-->
 <unidades>mm</unidades>
 <minimo>0</minimo>
 <maximo>500</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>TE525MM</modelo>
 <sql>rain</sql>
 <columna>cb26rain</columna>
 <Factor>Acumulacion</Factor>
</rain>
<dewpoint>
 <unidades>centigrados</unidades>
 <minimo>-40</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>HC2S3</modelo>
 <sql>dewpoint</sql>
 <columna>cb26dew</columna>
 <Factor>Temperatura</Factor>
```

```
  </dewpoint>
  <windchill>
   <unidades>centigrados</unidades>
   <minimo>-40</minimo>
   <maximo>100</maximo>
   <marca>Campbell Scientific</marca>
   <modelo>HC2S3</modelo>
   <sql>windchill</sql>
   <columna>cb26windc</columna>
   <Factor>Temperatura</Factor>
  </windchill>
  <ET> <!-- se reinicia el sensor en 24 horas  -->
   <unidades>mm</unidades>
   <minimo>0</minimo>
   <maximo>100</maximo>
   <marca>Campbell Scientific</marca>
   <modelo>HC2S3</modelo>
   <sql>ET</sql>
   <columna>cb26et</columna>
   <Factor>Acumulacion</Factor>
  </ET>
  <radiation>
   <unidades>w/m2</unidades>
   <minimo>0</minimo>
   <maximo>4000</maximo>
   <marca>Campbell Scientific</marca>
   <modelo>CMP11</modelo>
   <sql>radiation</sql>
   <columna>cb26radia</columna>
   <Factor>Radiacion</Factor>
  </radiation>
  <soilTemp1>
   <unidades>centigrados</unidades>
   <minimo>-40</minimo>
   <maximo>70</maximo>
   <marca>Campbell Scientific</marca>
   <modelo>109SS</modelo>
   <sql>soilTemp1</sql>
   <columna>cb26soil</columna>
   <Factor>Temperatura</Factor>
  </soilTemp1>
 </sensores>
</Estacion>
<Estacion nombre="Estacion 05">  <!-- Campbell - Meteorologica -->
 <nombre>Estacion 05</nombre>
 <sistemamt>16</sistemamt>
 <intervalo>1</intervalo>
 <altura>1.50</altura>
 <tablasql>weewx05</tablasql>
 <url>url</url>
 <ubicacion>
  <descripcion>Campus NCG</descripcion>
  <domicilio>El Triunfo, heroes del Chamizal</domicilio>
  <municipio>Nuevo Casas Grandes</municipio>
  <estado>Chihuahua</estado>
 </ubicacion>
 <geolocalizacion>
  <latitud>30.413888888888888</latitud>
  <longitud>-107.91166666666668</longitud>
```

```
  <altitud>1470 msnm</altitud>
</geolocalizacion>
<marca>Campbell Scientific</marca>
<modelo>xmodelo</modelo>
<tipo>Meteorologica</tipo>
<fechains>xfechainst</fechains>
<fechaact>xfechaA</fechaact>
<datalogger>CR1000</datalogger>
<sw>LoggerNet</sw>
<caracteristicas>xcaracteristicas</caracteristicas>
<imagen>ximagen</imagen>
<sensores>
  <barometer>
    <unidades>milibares</unidades>
    <minimo>500</minimo>
    <maximo>1100</maximo>
    <marca>Campbell Scientific</marca>
    <modelo>CS100</modelo>
    <sql>barometer</sql>
    <columna>cb05baro</columna>
    <Factor>Barometrico</Factor>
  </barometer>
  <outTemp>
    <unidades>centigrados</unidades>
    <minimo>-40</minimo>
    <maximo>60</maximo>
    <marca>Campbell Scientific</marca>
    <modelo>HC2S3</modelo>
    <sql>outTemp</sql>
    <columna>cb05outte</columna>
    <Factor>Temperatura</Factor>
  </outTemp>
  <outHumidity>
    <unidades>porcentaje</unidades>
    <minimo>0</minimo>
    <maximo>100</maximo>
    <marca>Campbell Scientific</marca>
    <modelo>HC2S3</modelo>
    <sql>outHumidity</sql>
    <columna>cb05outhum</columna>
  </outHumidity>
  <windSpeed>
    <unidades>m/s</unidades>
    <minimo>0</minimo>
    <maximo>100</maximo>
    <marca>Campbell Scientific</marca>
    <modelo>05108</modelo>
    <sql>windSpeed</sql>
    <columna>cb05windsp</columna>
  </windSpeed>
  <windDir>
    <unidades>centigrados</unidades>
    <minimo>0</minimo>
    <maximo>360</maximo>
    <marca>Campbell Scientific</marca>
    <modelo>05108</modelo>
    <sql>windDir</sql>
    <columna>cb05windir</columna>
    <Factor>Temperatura</Factor>
```

```
</windDir>
<windGust>
 <unidades>m/s</unidades>
 <minimo>0</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>05108</modelo>
 <sql>windGust</sql>
 <columna>cb05windgus</columna>
</windGust>
<windGustDir>
 <unidades>centigrados</unidades>
 <minimo>0</minimo>
 <maximo>360</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>05108</modelo>
 <sql>windGustDir</sql>
 <columna>cb05windgusd</columna>
 <Factor>Temperatura</Factor>
</windGustDir>
<rainRate>
 <unidades>mm/h</unidades>
 <minimo>0</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>TE525MM</modelo>
 <sql>rainRate</sql>
 <columna>cb05rainr</columna>
 <Factor>Acumulacion hora</Factor>
</rainRate>
<rain>
 <unidades>mm</unidades>
 <minimo>0</minimo>
 <maximo>500</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>TE525MM</modelo>
 <sql>rain</sql>
 <columna>cb05rain</columna>
 <Factor>Acumulacion</Factor>
</rain>
<dewpoint>
 <unidades>centigrados</unidades>
 <minimo>-40</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>HC2S3</modelo>
 <sql>dewpoint</sql>
 <columna>cb05dew</columna>
 <Factor>Temperatura</Factor>
</dewpoint>
<windchill>
 <unidades>centigrados</unidades>
 <minimo>-40</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>HC2S3</modelo>
 <sql>windchill</sql>
 <columna>cb05windc</columna>
 <Factor>Temperatura</Factor>
```

```
  </windchill>
  <ET>
   <unidades>mm</unidades>
   <minimo>0</minimo>
   <maximo>100</maximo>
   <marca>Campbell Scientific</marca>
   <modelo>HC2S3</modelo>
   <sql>ET</sql>
   <columna>cb05et</columna>
   <Factor>Acumulacion</Factor>
  </ET>
  <radiation>
   <unidades>w/m2</unidades>
   <minimo>0</minimo>
   <maximo>4000</maximo>
   <marca>Campbell Scientific</marca>
   <modelo>CMP11</modelo>
   <sql>radiation</sql>
   <columna>cb05radia</columna>
   <Factor>Radiacion</Factor>
  </radiation>
  <soilTemp1>
   <unidades>centigrados</unidades>
   <minimo>-40</minimo>
   <maximo>70</maximo>
   <marca>Campbell Scientific</marca>
   <modelo>109SS</modelo>
   <sql>soilTemp1</sql>
   <columna>cb05soil</columna>
   <Factor>Temperatura</Factor>
  </soilTemp1>
 </sensores>
</Estacion>
<Estacion nombre="Estacion 03">  <!-- Campbell - Meteorologica -->
 <nombre>Estacion 03</nombre>
 <sistemamt>16</sistemamt>
 <intervalo>1</intervalo>
 <altura>1.50</altura>
 <tablasql>weewx03</tablasql>
 <url>url</url>
 <ubicacion>
  <descripcion>Rancho Universitario</descripcion>
  <domicilio>Praxedis G. Guerrero</domicilio>
  <municipio>Paxedis</municipio>
  <estado>Chihuahua</estado>
 </ubicacion>
 <geolocalizacion>
  <latitud>31.35666666666667</latitud>
  <longitud>-105.79916666666666</longitud>
  <altitud>1080 msnm</altitud>
 </geolocalizacion>
 <marca>Campbell Scientific</marca>
 <modelo>xmodelo</modelo>
 <tipo>Meteorologica</tipo>
 <fechains>xfechainst</fechains>
 <fechaact>xfechaA</fechaact>
 <datalogger>CR1000</datalogger>
 <sw>LoggerNet</sw>
 <caracteristicas>xcaracteristicas</caracteristicas>
```

```
<imagen>ximagen</imagen>
<sensores>
 <barometer>
  <unidades>milibares</unidades>
  <minimo>500</minimo>
  <maximo>1100</maximo>
  <marca>Campbell Scientific</marca>
  <modelo>CS100</modelo>
  <sql>barometer</sql>
  <columna>cb03baro</columna>
  <Factor>Barometrico</Factor>
 </barometer>
 <outTemp>
  <unidades>centigrados</unidades>
  <minimo>-40</minimo>
  <maximo>60</maximo>
  <marca>Campbell Scientific</marca>
  <modelo>HC2S3</modelo>
  <sql>outTemp</sql>
  <columna>cb03outte</columna>
  <Factor>Temperatura</Factor>
 </outTemp>
 <outHumidity>
  <unidades>porcentaje</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Campbell Scientific</marca>
  <modelo>HC2S3</modelo>
  <sql>outHumidity</sql>
  <columna>cb03outhum</columna>
 </outHumidity>
 <windSpeed>
  <unidades>m/s</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Campbell Scientific</marca>
  <modelo>05108</modelo>
  <sql>windSpeed</sql>
  <columna>cb03windsp</columna>
 </windSpeed>
 <windDir>
  <unidades>centigrados</unidades>
  <minimo>0</minimo>
  <maximo>360</maximo>
  <marca>Campbell Scientific</marca>
  <modelo>05108</modelo>
  <sql>windDir</sql>
  <columna>cb03windir</columna>
  <Factor>Temperatura</Factor>
 </windDir>
 <windGust>
  <unidades>m/s</unidades>
  <minimo>0</minimo>
  <maximo>100</maximo>
  <marca>Campbell Scientific</marca>
  <modelo>05108</modelo>
  <sql>windGust</sql>
  <columna>cb03windgus</columna>
 </windGust>
```

```
<windGustDir>
 <unidades>centigrados</unidades>
 <minimo>0</minimo>
 <maximo>360</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>05108</modelo>
 <sql>windGustDir</sql>
 <columna>cb03windgusd</columna>
 <Factor>Temperatura</Factor>
</windGustDir>
<rainRate>
 <unidades>mm/h</unidades>
 <minimo>0</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>TE525MM</modelo>
 <sql>rainRate</sql>
 <columna>cb03rainr</columna>
 <Factor>Acumulacion hora</Factor>
</rainRate>
<rain>
 <unidades>mm</unidades>
 <minimo>0</minimo>
 <maximo>500</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>TE525MM</modelo>
 <sql>rain</sql>
 <columna>cb03rain</columna>
 <Factor>Acumulacion</Factor>
</rain>
<dewpoint>
 <unidades>centigrados</unidades>
 <minimo>-40</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>HC2S3</modelo>
 <sql>dewpoint</sql>
 <columna>cb03dew</columna>
 <Factor>Temperatura</Factor>
</dewpoint>
<windchill>
 <unidades>centigrados</unidades>
 <minimo>-40</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>HC2S3</modelo>
 <sql>windchill</sql>
 <columna>cb03windc</columna>
 <Factor>Temperatura</Factor>
</windchill>
<ET>
 <unidades>mm</unidades>
 <minimo>0</minimo>
 <maximo>100</maximo>
 <marca>Campbell Scientific</marca>
 <modelo>HC2S3</modelo>
 <sql>ET</sql>
 <columna>cb03et</columna>
 <Factor>Acumulacion</Factor>
```

```
  </ET>
  <radiation>
   <unidades>w/m2</unidades>
   <minimo>0</minimo>
   <maximo>4000</maximo>
   <marca>Campbell Scientific</marca>
   <modelo>CMP11</modelo>
   <sql>radiation</sql>
   <columna>cb03radia</columna>
   <Factor>Radiacion</Factor>
  </radiation>
  <soilTemp1>
   <unidades>centigrados</unidades>
   <minimo>-40</minimo>
   <maximo>70</maximo>
   <marca>Campbell Scientific</marca>
   <modelo>109SS</modelo>
   <sql>soilTemp1</sql>
   <columna>cb03soil</columna>
   <Factor>Temperatura</Factor>
  </soilTemp1>
 </sensores>
</Estacion>
<Estacion nombre="Teledyne">  <!--Gases01 -->
 <nombre>Teledyne</nombre>
 <sistemamt>16</sistemamt>
 <intervalo>1</intervalo>
 <tablasql>GasesP01/Teledyne</tablasql>
 <url>url</url>
 <ubicacion>
  <descripcion>IIT</descripcion>
  <domicilio>Av. del Charro 450 Nte, Partido Romero</domicilio>
  <municipio>Juarez</municipio>
  <estado>Chihuahua</estado>
 </ubicacion>
 <geolocalizacion>
  <latitud>31.743333333333336</latitud>
  <longitud>106.43166666666667</longitud>
  <altitud>1227 msnm</altitud>
 </geolocalizacion>
 <marca>Teledyne Api</marca>
 <modelo>xmodelo</modelo>
 <tipo>Gases y Particulas suspendidas</tipo>
 <fechains>Agosto - 2018</fechains>
 <fechaact>xfechaA</fechaact>
 <datalogger>Adam</datalogger>
 <sw>EMC Station Manager</sw>
 <caracteristicas>xcaracteristicas</caracteristicas>
 <imagen>ximagen</imagen>
 <sensores>
  <PM10>
   <unidades>ug/m3</unidades>
   <minimo>0.1</minimo>
   <maximo>10000</maximo>
   <marca>Teledyne Api</marca>
   <modelo>T640</modelo>
   <sql>PM10</sql>
   <columna>typm10</columna>
  </PM10>
```

```
<PM2_5>
 <unidades>ug/m3</unidades>
 <minimo>0.1</minimo>
 <maximo>10000</maximo>
 <marca>Teledyne Api</marca>
 <modelo>T640</modelo>
 <sql>PM2_5</sql>
 <columna>typm25</columna>
</PM2_5>
<PM10_2_5>
 <unidades>ug/m3</unidades>
 <minimo>0.1</minimo>
 <maximo>10000</maximo>
 <marca>Teledyne Api</marca>
 <modelo>T640</modelo>
 <sql>PM10_2_5</sql>
 <columna>typm1025</columna>
</PM10_2_5>
<PM10_1hr>
 <unidades>ug/m3</unidades>
 <minimo>0.1</minimo>
 <maximo>10000</maximo>
 <marca>Teledyne Api</marca>
 <modelo>T640</modelo>
 <sql>PM10_1hr</sql>
 <columna>typm101</columna>
</PM10_1hr>
<PM2_5_1hr>
 <unidades>ug/m3</unidades>
 <minimo>0.1</minimo>
 <maximo>10000</maximo>
 <marca>Teledyne Api</marca>
 <modelo>T640</modelo>
 <sql>PM2_5_1hr</sql>
 <columna>typm251</columna>
</PM2_5_1hr>
<PM10_2_5_1hr>
 <unidades>ug/m3</unidades>
 <minimo>0.1</minimo>
 <maximo>10000</maximo>
 <marca>Teledyne Api</marca>
 <modelo>T640</modelo>
 <sql>PM10_2_5_1hr</sql>
 <columna>typm10251</columna>
</PM10_2_5_1hr>
<PM10_12hr>
 <unidades>ug/m3</unidades>
 <minimo>0.1</minimo>
 <maximo>10000</maximo>
 <marca>Teledyne Api</marca>
 <modelo>T640</modelo>
 <sql>PM10_12hr</sql>
 <columna>typm1012</columna>
</PM10_12hr>
<PM2_5_12hr>
 <unidades>ug/m3</unidades>
 <minimo>0.1</minimo>
 <maximo>10000</maximo>
 <marca>Teledyne Api</marca>
```

```
  <modelo>T640</modelo>
  <sql>PM2_5_12hr</sql>
  <columna>typm2512</columna>
</PM2_5_12hr>
<PM10_2_5_12hr>
  <unidades>ug/m3</unidades>
  <minimo>0.1</minimo>
  <maximo>10000</maximo>
  <marca>Teledyne Api</marca>
  <modelo>T640</modelo>
  <sql>PM10_2_5_12hr</sql>
  <columna>typm102512</columna>
</PM10_2_5_12hr>
<PM10_24hr>
  <unidades>ug/m3</unidades>
  <minimo>0.1</minimo>
  <maximo>10000</maximo>
  <marca>Teledyne Api</marca>
  <modelo>T640</modelo>
  <sql>PM10_24hr</sql>
  <columna>typm1024</columna>
</PM10_24hr>
<PM2_5_24hr>
  <unidades>ug/m3</unidades>
  <minimo>0.1</minimo>
  <maximo>10000</maximo>
  <marca>Teledyne Api</marca>
  <modelo>T640</modelo>
  <sql>PM2_5_24hr</sql>
  <columna>typm2524</columna>
</PM2_5_24hr>
<PM10_2_5_24hr>
  <unidades>ug/m3</unidades>
  <minimo>0.1</minimo>
  <maximo>10000</maximo>
  <marca>Teledyne Api</marca>
  <modelo>T640</modelo>
  <sql>PM10_2_5_24hr</sql>
  <columna>typm102524</columna>
</PM10_2_5_24hr>
<SO2Ppm>
  <unidades>PPB</unidades>
  <minimo>0</minimo>
  <maximo>20000</maximo>
  <marca>Teledyne Api</marca>
  <modelo>T100</modelo>
  <sql>SO2Ppm</sql>
  <columna>tyso2</columna>
  <Factor>Concentracion ppm</Factor>
  <pm>64.0638</pm>
</SO2Ppm>
<NOxPpm>
  <unidades>PPB</unidades>
  <minimo>0</minimo>
  <maximo>20000</maximo>
  <marca>Teledyne Api</marca>
  <modelo>T200</modelo>
  <sql>NOxPpm</sql>
  <columna>tynox</columna>
```

```
      <Factor>Concentracion ppm</Factor>
      <pm>46.0055</pm>
     </NOxPpm>
     <NO2Ppm>
      <unidades>PPB</unidades>
      <minimo>0</minimo>
      <maximo>20000</maximo>
      <marca>Teledyne Api</marca>
      <modelo>T200</modelo>
      <sql>NO2Ppm</sql>
      <columna>tyno2</columna>
      <Factor>Concentracion ppm</Factor>
      <pm>46.0055</pm>
     </NO2Ppm>
     <NOPpm>
      <unidades>PPB</unidades>
      <minimo>0</minimo>
      <maximo>20000</maximo>
      <marca>Teledyne Api</marca>
      <modelo>T200</modelo>
      <sql>NOPpm</sql>
      <columna>tyno</columna>
      <Factor>Concentracion ppm</Factor>
      <pm>46.0055</pm>
     </NOPpm>
     <O3Ppm>
      <unidades>PPB</unidades>
      <minimo>0</minimo>
      <maximo>100</maximo>
      <marca>Teledyne Api</marca>
      <modelo>T400</modelo>
      <sql>O3Ppm</sql>
      <columna>tyo3</columna>
      <Factor>Concentracion ppm</Factor>
      <pm>47.998</pm>
     </O3Ppm>
     <COPpm>
      <unidades>PPM</unidades>
      <minimo>0</minimo>
      <maximo>1000</maximo>
      <marca>Teledyne Api</marca>
      <modelo>T300</modelo>
      <sql>COPpm</sql>
      <columna>tyco</columna>
      <Factor>Concentracion ppb</Factor>
      <pm>28.01</pm>
     </COPpm>
    </sensores>
   </Estacion>
  </Estaciones>
</Configuraciones>
```

Anexo III – Librerías del *framework*.

Archivos en Python que contienen las librerías del *framework.*

homoframe.py

El archivo *homoframe.py*, se encuentra formado por clases, una con las librerias de *list_stations, list_sensors, list_factors, range_dates, quality_data, save_session, homogeneizar*, y otra clase con las librerias para las salidas del *dataset*, como, *result_txt, result_json, result_xml* y *result_csv.*

```
# -*- coding: utf-8 -*-
# Framework formed by functions set
# homoframe has functions to process homogeization
__author__ = 'Alicia Jimenez'

from engine import DB, Utils
from conversor import Conversor

import os # save txt
from bson.json_util import dumps # save json
import csv # csv
import numpy as np # csv
import pandas as pd # csv
# XML
from lxml import objectify
import requests
import xml.etree.ElementTree as ET
import urllib.request # url
import time # Sleep and epoch
from functools import reduce  # dict
import operator # dict

def getFromDict(dataDict, mapList):
  return reduce(operator.getitem, mapList, dataDict)

iDB = DB()
iUtils = Utils()
iConversor = Conversor()

class Dataset:
  def __init__(self):
    self.iDB = DB()

  def switch_data(self,wparameters='',wsession='',opcion=''):
    pos1=wsession.find("session")
    num=wsession[pos1+7:]
    file='README'+num+'.txt'
    self.result_txt(file,wparameters)
    if opcion == '1':
      self.result_json(wsession)
    elif opcion == '2':
      self.result_csv(wparameters,wsession)
    elif opcion == '3':
      self.result_xml(wsession)

  # create readme.txt
  def result_txt(self,filename='',parameters=''):
    dicstations={}
    dicsensors={}
    dicfactors={}
    dicstate={}
```

```
    dicdates={}
    dicheader={}
    collectionArchive = self.iDB.obtener_datos(parameters)
    for xS in collectionArchive:
        if 'Estaciones' in xS:
            dicstations = xS['Estaciones']
        if 'Valores' in xS:
            dicsensors = xS['Valores']
        if 'Calidad' in xS:
            dicstate = xS['Calidad']
        if 'Fechas' in xS:
            dicdates = xS['Fechas']
        if 'Factores' in xS:
            dicfactors = xS['Factores']
        if 'Encabezados' in xS:
            dicheader = xS['Encabezados']
    with open(filename,"w") as file:
        file.write("Este archivo contiene los valores con los que se genero el dataset: " + os.linesep)
        file.write("\n  Estaciones: {} ".format(dicstations) + os.linesep)
        file.write("\n  Sensores por cada estación, factor de conversión utilizado, peso molecular (en los casos que aplique para la
conversión - pm) y la unidad original del sensor: \n  {} ".format(dicsensors) + os.linesep)
        file.write("\n  Calidad de datos: {} ".format(dicstate) + os.linesep)
        file.write("  Para los datos de calidad, se incluyeron: " + os.linesep)
        file.write("  Calidad 1 (originales): " )
        file.write("Datos estado 0 (Recien insertados) y 1 (Pasaron por proceso de calidad y son datos correctos)"+os.linesep)
        file.write("  Calidad 2 (sospechosos): " )
        file.write("Datos estado 2 (Pasaron por proceso de calidad y son datos sospechosos)"+os.linesep)
        file.write("  Calidad 3 (corregidos): " )
        file.write("Datos estado 1 (Pasaron por proceso de calidad y son datos correctos) y 3 (Pasaron por proceso de calidad y
fueron corregidos)"+os.linesep)
        file.write("  Los datos con estados NO incluidos en la calidad, se les asigno NULL para su manejo" + os.linesep)
        file.write("\n  El rango de fechas que utiliza se muestra en tiempo universal coordinado (UTC/GMT) ")
        file.write("\n  Rango de fechas: {} ".format(dicdates) + os.linesep)
        file.write("\n  Unidades de salida de acuerdo con la homogeneización: {} ".format(dicfactors) + os.linesep)
        file.write("\n  Encabezados (para el archivo csv):\n {} ".format(dicheader) + os.linesep)
        file.close()

  def jsonDefault(self, object):
    return object.__dict__

  # create json
  def result_json(self, wsession):
    collectionArchive = list(self.iDB.obtener_datos_sort(wsession,'','dateTime'))
    filename = str(wsession)+'.json'
    file = open(filename, "w")
    file.write(dumps(collectionArchive))
    file.close()

  # create xml
  def result_xml(self, wsession=''):
    collectionArchive =  list(self.iDB.obtener_datos_sort(wsession,'','dateTime'))
    filename = str(wsession)+'.xml'
    root = ET.Element("root")
    for document in collectionArchive:
        estaciones = document['Estaciones']
        # Create new Element
        doc_tag=ET.Element("dateTime",{'timestamp':str(document['dateTime']),'fecha':str(document['fecha'])})
        for k,v in estaciones.items():
            station_tag = ET.Element("Estacion",name=k)
            for sen in v['sensores']:
                ET.SubElement(station_tag, sen).text = str(v['sensores'][sen])
            doc_tag.append(station_tag)
```

```
        root.append(doc_tag)
    arbol = ET.ElementTree(root)
    arbol.write(filename)

  # create csv
  def result_csv(self,parameters='',wsession=''):
    collectionP = self.iDB.obtener_datos(parameters)
    ListHeader = []
    for xS in collectionP:
       if 'Encabezados' in xS:
          ListHeader = xS['Encabezados']
    collectionArchive = list(self.iDB.obtener_datos_sort(wsession,'','dateTime'))
    columnas = len(ListHeader)
    filas = self.iDB.cuantos(wsession)
    my_array = [ [ np.nan for i in range(columnas) ] for j in range(filas) ]
    fil=0
    for document in collectionArchive:
       estaciones = document['Estaciones']
       for k,v in estaciones.items():
          my_array[fil][0]=document['dateTime']
          my_array[fil][1]=document['fecha']
          pos=ListHeader.index(k)
          my_array[fil][pos]=k
          for sen in v['sensores']:
             pos=ListHeader.index(sen)
             my_array[fil][pos]=v['sensores'][sen]
       fil=fil+1
    df = pd.DataFrame(my_array, columns=ListHeader)
    filename = str(wsession)+'.csv'
    df.to_csv(filename, index=False)

class Frame():
  def __init__(self):
    self.iDB = DB()
    self.iConversor = Conversor()

  # Show list stations the XML
  def list_stations(self,url='http://cecatev.uacj.mx/EstacionX3.xml'):
      elements = {}
      rurl = requests.get(url)
      objeto = objectify.fromstring(rurl.content)
      for item in objeto.Estaciones.Estacion:
        elements[item.get('nombre')]=item.find('tipo').text
      return {'elements': elements}

  # Show list sensors the XML
  def list_sensors(self,url='http://cecatev.uacj.mx/EstacionX3.xml',parameters=''):
    collectionArchive = self.iDB.obtener_datos(parameters)
    Stations = {}
    for xS1 in collectionArchive:
       if 'Estaciones' in xS1:
          Stations = xS1['Estaciones']
    req1=urllib.request.Request(url)
    resp1=urllib.request.urlopen(req1)
    tree=ET.parse (resp1)
    dSensors = {}
    #lista =[]
    elements = {}
    for xS in Stations:
       Station = tree.find('./Estaciones/Estacion[@nombre="'+xS+'"]')
       dicSensors = Station.find("sensores")
       dSensors={}
```

```
      #lista=[]
      for key in dicSensors:
         dSensors[key.tag]=({'unidad':key.find('unidades').text})
         #lista.append(key.tag)
      elements[xS]= dSensors #lista
   return {'elements': elements}

 # Save valores en la coleccion
 def saveValores(self,url='http://cecatev.uacj.mx/EstacionX3.xml',parameters='',elements={}):
   collectionArchive = self.iDB.obtener_datos(parameters)
   Sensors = {}
   for xS1 in collectionArchive:
      if 'Sensores' in xS1:
         Sensors = xS1['Sensores']
   req1=urllib.request.Request(url)
   resp1=urllib.request.urlopen(req1)
   tree = ET.parse (resp1)
   dicSensors={}
   allSensors = {}
   dicFactores={}
   dicHeader={}
   listaFactor=[]
   LHeader=[]
   LHeader.append('dateTime')
   LHeader.append('fecha')
   for xS in Sensors:
      dSensors = {}
      Station = tree.find('./Estaciones/Estacion[@nombre="'+xS+'"]')
      ListaSensores=Sensors[xS]
      LHeader.append(xS)
      for x in ListaSensores:
        sensor = Station.find("sensores/"+x)
        xfactor = sensor.findtext('Factor',default=None)
        xpm = sensor.findtext('pm',default=None)
        dSensors[x]=({'unidadE':sensor.find('unidades').text,'factor':xfactor,'pm':xpm})
        listaFactor.append(xfactor)
        LHeader.append(x)
      allSensors[xS]=dSensors
   dicSensors['Valores']=allSensors
   self.iDB.actualizar('Valores',parameters,dicSensors)
   LFactor = list(set((filter(None, listaFactor))))
   dicFactores['Factores']=sorted(LFactor)
   dicHeader['Encabezados']=LHeader
   self.iDB.actualizar('Factores',parameters,dicFactores)
   self.iDB.actualizar('Encabezados',parameters,dicHeader)

 # Seleccion de factores de conversion para salida
 def list_factors(self, url='http://cecatev.uacj.mx/EstacionX3.xml',parameters=''):
   collectionArchive = self.iDB.obtener_datos(parameters)
   Factors = {}
   for xS1 in collectionArchive:
      if 'Factores' in xS1:
         Factors = xS1['Factores']
   req1=urllib.request.Request(url)
   resp1=urllib.request.urlopen(req1)
   tree=ET.parse (resp1)
   elements = {}
   LSensors=[]
   for xS in Factors:
    xFactor = tree.find('./Factores/Factor[@nombre="'+xS+'"]')
    LSensors=[]
    i=0
```

```
    for key in xFactor:
      i+=1
      LSensors.append(key.text)
    elements[xS]=LSensors
  return {'elements': elements}

 # Selection dates
 def range_dates(self, parameters='', d1='', d2=''):
  dicDates={}
  dict1={}
  dict2={}
  dict1['date1'] = d1
  dict2['date2'] = d2
  dicDates['Fechas']=dict(dict1, **dict2)
  self.iDB.actualizar('Fechas',parameters,dicDates)

 def quality_data(self,parameters='',estado=''):
  dicState={}
  dicState['Calidad']=estado
  self.iDB.actualizar('Calidad',parameters,dicState)

 def saveSession(self, tabla='archive', parameters='', wsession=''):
  start = time.time()
  self.iDB.borrar(wsession)
  dicstations={}
  dicsensors={}
  dicFactor={}
  dicstate={}
  dicdates={}
  collectionParameters = self.iDB.obtener_datos(parameters)
  for key in collectionParameters:
     if 'Estaciones' in key:
        dicstations = key['Estaciones']
     if 'Valores' in key:
        dicsensors = key['Valores']
     if 'Calidad' in key:
        dicstate = key['Calidad']
     if 'Fechas' in key:
        dicdates = dict(key['Fechas'])
     if 'Factores' in key:
        dicFactor = key['Factores']
  # Construyo el query para la fecha
  queryFec={}
  if 'date1' in dicdates:
     d1 = dicdates['date1']
  if 'date2' in dicdates:
     d2 = dicdates['date2']
  date1 = self.iConversor.utc_to_epoch(d1)
  date2 = self.iConversor.utc_to_epoch(d2)
  queryFecList = []
  queryFecList.append({"$gte":date1})   # 2020-01-15 00:05
  queryFecList.append({"$lte":date2})   # 2020-01-15 23:55
  queryFec["dateTime"] = {"$gte":date1, "$lte":date2}
  print('queryfecha: {}'.format(queryFec))

  # Construyo query de las estaciones
  queryList=[]
  queryStation={}
  for i in dicstations:
     queryList.append({"station":i})
  queryStation["$or"] = queryList
  print('queryStation: {}'.format(queryStation))
```

```
# Calidad de los datos
querySList=[]
queryState={}
for i in dicstate:
   if i=='1':
      querySList.append({"data.state":0})
      querySList.append({"data.state":1})
      xestado=1
   if i=='2':
      querySList.append({"data.state":2})
      querySList.append({"data.state":2})
      xestado=2
   if i=='3':
      querySList.append({"data.state":1})
      querySList.append({"data.state":3})
      xestado=3
queryState["$or"] = querySList
print('queryState: {}'.format(queryState))

cadena = {}
cadena={'$and':[queryStation,queryState,queryFec]}
print('\nConsulta: {}'.format(cadena))
collectionArchive = list(self.iDB.obtener_datos_sort(tabla,cadena,'dateTime'))

varFecha=''
docto = {}
docto2={} # esta variable si no se vacia concatena las estaciones
doctoN1 = {}

for xS in collectionArchive:
 dic = xS['data']
 estado = dic['state']
 sensorsdata = dic['sensor']
 # para empotrar por fecha
 if varFecha!=xS['dateTime'] and docto!={} :
  docto2={} # esta variable si no se vacia concatena las estaciones
  self.iDB.insertar(wsession,doctoN1)
 varFecha=xS['dateTime']
 docto={}
 for sensorname in sensorsdata.keys():
    try:
     xUE = getFromDict(dicsensors, [xS['station'],sensorname,'unidadE'])
     xfactor = getFromDict(dicsensors, [xS['station'],sensorname,'factor'])
     if xfactor==None :
      xUS = xUE
     else:
      xUS=getFromDict(dicFactor, [xfactor])
     xpm = getFromDict(dicsensors, [xS['station'],sensorname,'pm'])
     if xpm==None :
      xpm = 0
     sensorvalue = sensorsdata[sensorname]['value']
     # si es nulo lo debe dejar nulo
     xvalue=sensorvalue
     if estado==3:
      if 'newvalue' in sensorsdata[sensorname].keys():
       xvalue = sensorsdata[sensorname]['newvalue']
     # esta parte es para eliminar los demas valores
     if xestado==1:
        if estado==2 or estado==3:
           xvalue=None
     if xestado==2:
```

```
            if estado==0 or estado==1 or estado==3:
              xvalue=None
          if xestado==3:
            if estado==0 or estado==2:
              xvalue=None
          # termino eliminar valores diferente estado
          if xvalue != None and xvalue != 0:
           # homogeneizar el valor de acuerdo con las unidades de entrada y salida
           xvalor=self.homogeneizar(xvalue,xUE,xUS,xpm,xfactor)
           docto.update({sensorname : xvalor})
          else :
           docto.update({sensorname : xvalue})
        except KeyError:
         pass

    doctoN1 = {}
    docto1={}
    fecha = self.iConversor.epoch_to_utc(xS['dateTime'])
    doctoN1 = {'dateTime':xS['dateTime'], 'fecha':str(fecha)}
    docto1['sensores'] = docto
    docto2[xS['station']]=docto1
    doctoN1['Estaciones']=docto2

  self.iDB.insertar(wsession,doctoN1)
  end = time.time()
  xtiempo= end-start
  print('\nTiempo total grabar session: {}'.format(xtiempo))
  print('Total documentos insertados: {}'.format(self.iDB.cuantos(wsession)))
  self.iDB.crea_indice(wsession,'dateTime')

# homogeniza la salida
def homogeneizar(self,xquantity,unitE,unitS,Pm,Factor):
  if unitE==unitS:
    nquantity=xquantity
    return nquantity
  if Factor=='Barometrico':
    if unitE=='milibares':
      nquantity=self.iConversor.mb(xquantity,unitS)
    else:
      valor1=self.iConversor.to_mb(xquantity,unitE)
      nquantity=self.iConversor.mb(valor1,unitS)
    return nquantity
  if Factor=='Temperatura':
    if unitE=='centigrados':
      nquantity=self.iConversor.celsius(xquantity,unitS)
    else:
      valor1=self.iConversor.to_celsius(xquantity,unitE)
      nquantity=self.iConversor.celsius(valor1,unitS)
    return nquantity
  if Factor=='Velocidad':
    if unitE=='km/h':
      nquantity=self.iConversor.km_h(xquantity,unitS)
    else:
      valor1=self.iConversor.to_km_h(xquantity,unitE)
      nquantity=self.iConversor.km_h(valor1,unitS)
    return nquantity
  if Factor=='Radiacion':
    if unitE=='w/m2':
      nquantity=self.iConversor.w_m2(xquantity,unitS)
    else:
      valor1=self.iConversor.to_w_m2(xquantity,unitE)
      nquantity=self.iConversor.w_m2(valor1,unitS)
```

```
  return nquantity
if Factor=='Acumulacion' or Factor=='Acumulacion hora':
  if unitE=='mm' or unitE=='mm/h':
    nquantity=self.iConversor.mm(xquantity,unitS)
  else:
    valor1=self.iConversor.to_mm(xquantity,unitE)
    nquantity=self.iConversor.mm(valor1,unitS)
  return nquantity
if Factor=='Concentracion ppm' or Factor=='Concentracion ppb': # Factor=='Concentracion microgramos' or
  nquantity=self.iConversor.pollutants(xquantity,unitE,unitS,Pm)
  return nquantity
```

conversor.py

Contiene las librerias usadas para las diferentes conversiones.

```
# -*- coding: utf-8 -*-
# Framework formed by functions set
# conversor has functions to change units
__author__ = 'Alicia Jimenez'

import time # Sleep and epoch
from datetime import datetime, date #epoch
import calendar

class Conversor:
  def utc_to_epoch(self,timestamp_string):
    '''Use this function to convert utc to epoch'''
    timestamp = datetime.strptime(timestamp_string, '%Y-%m-%d %H:%M')
    epoch = int(calendar.timegm(timestamp.utctimetuple()))
    return epoch

  def epoch_to_utc(self,timestamp):
    fecha = datetime.fromtimestamp(timestamp)
    return fecha

  def mb(self,quantity,unit): # Factor 1
    # funcion que convierte de mb a otras unidades
    if unit == 'Pa': #print('Unidad en mb a Pa')
      newquantity = quantity * 100
      return newquantity
    elif unit == 'Atm': #print('Unidad en mb a Atm')
      newquantity = quantity * 0.00098692
      return newquantity
    elif unit == 'inHg': #print('Unidad en mb a inHg')
      newquantity = quantity * 0.02953007
      return newquantity
    elif unit == 'hPa': #print('Unidad en mb a hPa')
      newquantity = quantity
      return newquantity
    elif unit == 'mmHg': #print('Unidad en mb a mmHg')
      newquantity = quantity * 0.7500638
      return newquantity
    elif unit == 'Psi': #print('Unidad en mb a Psi')
      newquantity = quantity * 0.01450377
      return newquantity
    elif unit == 'Torr': #print('Unidad en mb a Torr')
      newquantity = quantity * 0.7500638
      return newquantity

  def to_mb(self,quantity,unit): # Factor 1
    # funcion que convierte de otras unidades a mb
    if unit == 'Pa': #print('Unidad de Pa a mb')
      newquantity = quantity * 0.01
      return newquantity
    elif unit == 'Atm': #print('Unidad de Atm a mb')
      newquantity = quantity * 1013.25
      return newquantity
    elif unit == 'inHg': #print('Unidad de inHg a mb')
      newquantity = quantity * 33.86
      return newquantity
    elif unit == 'hPa': #print('Unidad de hPa a mb')
      newquantity = quantity
      return newquantity
    elif unit == 'mmHg': #print('Unidad de mmHg a mb')
```

```
    newquantity = quantity * 1.33
    return newquantity
  elif unit == 'Psi': #print('Unidad de Psi a mb')
    newquantity = quantity * 68.95
    return newquantity
  elif unit == 'Torr': #print('Unidad de Torr a mb')
    newquantity = quantity * 1.33
    return newquantity

 def celsius(self,quantity,unit): # Factor 2
  # funcion que convierte de celsius a otras unidades
  if unit == 'fahrenheit': #print('Unidad en Celsius a Fahrenheit')
    newquantity = (quantity * 9/5)+32
    return newquantity
  elif unit == 'kelvin': #print('Unidad en Celsius a Kelvin')
    newquantity = quantity + 273.15
    return newquantity

 def to_celsius(self,quantity,unit): # Factor 2
  # funcion que convierte de otras unidades a celsius
  if unit == 'fahrenheit': #print('Unidad en Fahrenheit a Celsius')
    newquantity = (quantity -32) * 5/9
    return newquantity
  elif unit == 'kelvin': #print('Unidad en Kelvin a Celsius')
    newquantity = quantity - 273.15
    return newquantity

 def km_h(self,quantity,unit): # Factor 3
  # funcion que convierte de km/h a otras unidades
  if unit == 'mps': #print('Unidad en km/h a m/s')
    newquantity = quantity / 3.6
    return newquantity
  elif unit == 'mph': #print('Unidad en km/h a mph')
    newquantity = quantity / 1.609
    return newquantity
  elif unit == 'nudo': #print('Unidad en km/h a nudo')
    newquantity = quantity / 1.852
    return newquantity

 def to_km_h(self,quantity,unit): # Factor 3
  # funcion que convierte de otras unidades a km/h
  if unit == 'mps': #print('Unidad en m/s a km/h')
    newquantity = quantity * 3.6
    return newquantity
  elif unit == 'mph': #print('Unidad en mph a km/h')
    newquantity = quantity * 1.609
    return newquantity
  elif unit == 'nudo': #print('Unidad en nudo a km/h')
    newquantity = quantity * 1.852
    return newquantity

 def w_m2(self,quantity,unit): # Factor 4
  # funcion que convierte de wh/m2 a otras unidades
  if unit == 'langley': #print('Unidad en w/m2 a langley')
    newquantity = quantity * 0.085985
    return newquantity
  elif unit == 'btu/ft2': #print('Unidad en w/m2 a btu/ft2')
    newquantity = quantity * 0.3170
    return newquantity
  elif unit == 'joules/m2': #print('Unidad en w/m2 a joules/m2')
    newquantity = quantity * 3600
    return newquantity
```

```
def to_w_m2(self,quantity,unit): # Factor 4
  # funcion que convierte de otras unidades a w/m2
  if unit == 'langley': #print('Unidad en langley a w/m2')
    newquantity = quantity / 0.085985
    return newquantity
  elif unit == 'btu/ft2': #print('Unidad en btu/ft2 a w/m2')
    newquantity = quantity / 0.3170
    return newquantity
  elif unit == 'joules/m2': #print('Unidad en joules/m2 a w/m2')
    newquantity = quantity / 3600
    return newquantity

def mm(self,quantity,unit): # Factor 5 y 6
  # funcion que convierte de mm a otras unidades
  if unit == 'in' or unit == 'in/h' : #print('Unidad en mm a in')
    newquantity = quantity * 0.0394
    return newquantity

def to_mm(self,quantity,unit): # Factor 5 y 6
  # funcion que convierte de otras unidades a mm
  if unit == 'in' or unit == 'in/h': #print('Unidad en in a mm')
    newquantity = quantity / 0.0394
    return newquantity

def pollutants(self,quantity,unit_1,unit_2,pm):
  # funcion conversora de contaminantes de acuerdo con la unidad y gas (para el peso molecular)
  if unit_1 == 'ug/m3': # de ug/m3 a varios
    if unit_2 == 'ug/m3':   #print('Unidad en ug/m3 a ug/m3')
      newquantity = quantity
    elif unit_2 == 'ppm': #print('Unidad en ug/m3 a ppm')
      newquantity = quantity / (float(pm) * (pow(10,3)/24.5))
    elif unit_2 == 'ppb': #print('Unidad en ug/m3 a ppb')
      newquantity = (quantity / (float(pm) * (pow(10,3)/24.5))) * 1000
    return newquantity
  elif unit_1 == 'PPM': # de ppm a varios
    if unit_2 == 'ppm':   # a ppm
      newquantity = quantity
    elif unit_2 == 'ug/m3': # a ug/m3
      newquantity = ((quantity * float(pm))/24.5) * pow(10,3)
    elif unit_2 == 'ppb': # a ppb
      newquantity = quantity * 1000
    return newquantity
  elif unit_1 == 'PPB': # de ppb a varios
    if unit_2 == 'ppb':   # a ppb
      newquantity = quantity
    elif unit_2 == 'ug/m3': # a ug/m3
      newquantity = (((quantity/1000) * float(pm))/24.5) * pow(10,3)
    elif unit_2 == 'ppm': # a ppm
      newquantity = quantity / 1000
    return newquantity
```

engine.py

Contiene las librerias de conexión a la base de datos, operaciones de actualización en MongoDB y utilerias.

```
# -*- coding: utf-8 -*-
# Framework formed by functions set
# engine has functions connection db and utils
__author__ = 'Alicia Jimenez'
# Mongo
import pymongo # MongoClient
from pymongo.errors import ConnectionFailure, AutoReconnect # Errors
# XML
from lxml import objectify
import requests
import time # Sleep and epoch
import sys #exception errores

class DB:
  # Function to Connect to the MongoDB
  # MongoDB IP is in file XML
  def connect(self, host = "localhost", port = "27017", bd = "Clima"):
    self.mongoClient = pymongo.MongoClient("mongodb://{}:{}".format(host, port))
    self.db = self.mongoClient[bd]
    return self.db

  def insertar(self,coleccion,datos):
    try:
      bd = self.connect()
      var = bd[coleccion].insert_one(datos).inserted_id
      return var
    except AutoReconnect as ex:
      print(' Fallo insertar documento ... {} '.format(ex))
      time.sleep(1)
      self.insertar(coleccion, datos)
    except Exception as ex:
      print('Error al insertar documento ... : {}'.format(sys.exc_info()[0]))
      print('documento: {}'.format(datos))

  def obtener_datos(self,coleccion):
    try:
      bd = self.connect()
      return bd[coleccion].find()
    except:
      print('Error al obtener datos ... ')

  def obtener_datos_sort(self,coleccion,cadena,orden):
    try:
      bd = self.connect()
      if cadena=='':
        return bd[coleccion].find({},{'_id':0}).sort(orden, pymongo.ASCENDING)
      else :
        return bd[coleccion].find(cadena).sort(orden, pymongo.ASCENDING)
    except:
      print('Error al obtener datos ... ')

  def eliminar(self,coleccion,llave):
    try:
      bd = self.connect()
      collectionArchive = self.obtener_datos(coleccion)
      for xS in collectionArchive:
        if llave in xS:
```

```
            bd[coleccion].delete_one({'_id':xS['_id']})
      except:
        print('Error al borrar la llave de la coleccion ... ')

    def borrar(self, coleccion):
      try:
        bd = self.connect()
        bd[coleccion].drop()
      except:
        print('Error al borrar la coleccion ... ')

    def actualizar(self, llave, coleccion, datos):
      try:
        if llave=='Estaciones':
          self.eliminar(coleccion,'Sensores')
          self.eliminar(coleccion,'Factores')
          self.eliminar(coleccion,'Valores')
          self.eliminar(coleccion,'Encabezados')
        self.eliminar(coleccion,llave)
        self.insertar(coleccion,datos)
      except:
        print('Error al actualizar la informacion ... ')

    def cuantos(self, coleccion):
      bd = self.connect()
      return bd[coleccion].count_documents({})

    def crea_indice(self,coleccion,indice):
      bd = self.connect()
      var = bd[coleccion].create_index(indice)
      return var

class Utils:
    def serverxml(self, url='http://cecatev.uacj.mx/EstacionX3.xml'):
      r = requests.get(url)
      objeto = objectify.fromstring(r.content)
      return objeto.Servidor.text
```

frame_clima.py

Ejemplo de implementación del *framework* en línea de comando.

```
# -*- coding: utf-8 -*-
# Framework formed by functions set
__author__ = 'Alicia Jimenez'

import random
from functools import reduce  # dict
import operator # dict

from engine import DB, Utils
from conversor import Conversor
from homoframe import Frame, Dataset

def getFromDict(dataDict, mapList):
 return reduce(operator.getitem, mapList, dataDict)

class frame_clima:
 # First constructor
 def __init__(self, bdOpt = {}):
  self.conversor = Conversor() # Instanciar conversor
  self.DB = DB() # Instanciar motor de BD
  self.frame = Frame() # Instanciar motor de Frame
  self.dataset = Dataset() # Instanciar motor de Dataset
  if bdOpt:
   self.db = self.DB.connect(bdOpt['host'], bdOpt['port'], bdOpt['bd'])
  else:
   self.db = self.DB.connect()
  self.create_session()
  print('Sesion: ', self.session)


 # Function to connect MongoDB and create collections session and parameters (random) for user session
 def create_session(self):
  # list collections to db Clima
  collist = self.db.list_collection_names()
  while True:
   # Create user session number (random)
   x = str(random.randrange(1000))
   xsession =  "session"+x
   xparameters = "parameters"+x
   self.session = xsession # eliminar es para pruebas
   self.parameters = xparameters # eliminar es para pruebas
   if xsession not in collist:
    self.session = xsession
    self.parameters = xparameters
    self.DB.insertar(self.parameters, {})
    self.DB.insertar(self.session, {})
    break

 # Function to delete collections session and parameters and Close the connection to MongoDB
 def close(self):
  #self.db[self.parameter].drop() # delete collection parameters
  #self.db[self.session].drop()  # delete collection session
  #self.mongoClient.close()   # close the connection to MongoDB
  print('Cierra sesion')

 def Estaciones(self,url,xparameters=''):
  print("\nEstaciones disponibles \n")
  elements={}
```

```
    # trae las estaciones disponibles de engine
    elements=self.frame.list_stations(url)['elements']

    #print('elements : {}'.format(elements))
    dicStations = {}
    allStations = []
    i=0
    for element in elements:
       i+=1
       print(str(i),'-',element)
       dicStations.update({i:element})
       allStations.append(i)
    print ("\nElija las estaciones de la lista, separando por coma cada estación (1,2,3,..) o (0) para todas las estaciones\nPresione
enter para finalizar")
    # create the stations list
    xStations=[]
    stationSelect = input ('>')
    if stationSelect =='0':
     stationSelect = allStations
    else:
     stationSelect = stationSelect.split(',')
    for x in stationSelect:
     x1 = int(x)
     if x1 in dicStations:
       xStations.append(dicStations.get(x1))
    # Graba estaciones en parametros
    dicStations = {}
    dicStations['Estaciones']=xStations
    #print('estaciones: {}'.format(dicStations))
    self.DB.actualizar('Estaciones',xparameters,dicStations)

  def Sensores(self,url,xparameters=''):
    print("\nSensores disponibles \n")
    elements={}
    # trae las estaciones disponibles de engine
    elements=self.frame.list_sensors(url,xparameters)['elements']

    dicSensores={}
    dicSensor={}
    for key in elements:
     i=0
     allSensors=[]
     dlistaSensors = {}
     print('\nEstación: {}'.format(key))
     for item in elements[key]:
        i+=1
        print(str(i),'-',item,'=',getFromDict(elements[key],[item,'unidad']))
        allSensors.append(i)
        dlistaSensors.update({i:item})
     print ("\nElija los sensores a mostrar (1,2,3,..) o (0) para todos los sensores ")
     print("\nPresione enter para finalizar")
     sensorSelect = input ('>')
     if sensorSelect =='0':
      sensorSelect = allSensors
     else:
      sensorSelect = sensorSelect.split(',')
     # genero la lista de los sensores que seleccionaron
     LSensors=[]
     for x in sensorSelect:
      x1 = int(x)
      if x1 in dlistaSensors:
       LSensors.append(dlistaSensors.get(x1))
```

```
    # Lleno un diccionario con los sensores seleccionados por estacion
    diccionario={}
    i=0
    for xx in LSensors:
      diccionario[xx]=LSensors[i]
      i+=1
    dicSensor[key]=LSensors
  dicSensores['Sensores']=dicSensor
  self.DB.actualizar('Sensores',xparameters,dicSensores)
  self.frame.saveValores(url,xparameters,dicSensores)

def Factores(self,url,xparameters=''):
  print("\nFactores de conversión \n")
  dicFactor={}
  dicFactores={}
  elements={}
  # trae los factores grabados en el engine
  elements=self.frame.list_factors(url,xparameters)['elements']

  for key in elements:
    i=0
    dlistaFactors={}
    print('\nFactor de conversión: {}'.format(key))
    for item in elements[key]:
       i+=1
       print(str(i),'-',item)
       dlistaFactors.update({i:item})
    print ("\nElija la unidad de salida (1...) o (0) para la unidad base del sensor ")
    factorSelect = input ('>')
    x1 = int(factorSelect)
    if x1 == 0:
      x1 = 1
    if x1 in dlistaFactors:
      dicFactor.update({key:dlistaFactors.get(x1)})
  dicFactores['Factores']=dicFactor
  self.DB.actualizar('Factores',xparameters,dicFactores)
  print('\n{}'.format(dicFactor))

# Selection dates
def Fechas(self,xparameters):
    try:
      d1 = input('\nIngrese la fecha inicial en formato (yyyy-mm-dd hh:mm): ')
      d2 = input('\nIngrese la fecha final en formato (yyyy-mm-dd hh:mm): ')
      self.frame.range_dates(xparameters,d1,d2)
    except Exception as e:
      print('-> Input date is not valid..: yyyy-mm-dd hh:mm  Try again!: {}'.format(e))

def Calidad(self,xparameters):
  print("\nSeleccione la calidad de los datos a utilizar (1 - Dato original, 2- Dato sospechoso, 3 - Dato procesado)")
  while True:
    try:
      estado=input('Calidad : ')
    except:
      print ('You have entered an invalid value.')
    else:
      if estado not in '1,2,3':
        print('Dato invalido, vuelve a intentar.')
      else:
        self.frame.quality_data(xparameters,estado)
        #print('Se va a grabar calidad.')
        break
```

```
  def GrabaSession(self,xparameters,xsession):
    self.frame.saveSession('archive',xparameters,xsession)

  def SalidaDataset(self,xparameters,xsession):
    print("\nElija el formato para el dataset (1-Json, 2-csv, 3-xml)")
    while True:
      try:
        xdataset=input('Dataset : ')
      except:
        print ('You have entered an invalid value.')
      else:
        if xdataset not in '1,2,3':
          print('Dato invalido, vuelve a intentar.')
        else:
          self.frame.switch_data(xparameters,xsession,xdataset)
          break

  def main(self):

    # XML source
    url = 'http://cecatev.uacj.mx/EstacionX3.xml'
    self.Estaciones(url,self.parameters)

    #print('\nLista de sensores')
    self.Sensores(url,self.parameters)

    #print('\nLista de factores')
    self.Factores(url,self.parameters)

    #print('\nRango de fechas')
    self.Fechas(self.parameters)

    #print('\nCalidad de datos')
    self.Calidad(self.parameters)

    #print('\nGraba coleccion\n')
    self.GrabaSession(self.parameters,self.session)

    #print('\nSelecciona salida dataset')
    self.SalidaDataset(self.parameters,self.session)


# principal process
if __name__ == '__main__':
  xhost = Utils.serverxml('http://cecatev.uacj.mx/EstacionX3.xml') # trae la ip del servidor
  frame = frame_clima( { 'host': xhost, 'port': "27017", 'bd': "Clima" } ) # servidor CECATEV
  frame.main()
```

views.py

Ejemplo de implementación del *framework* en Django en el archivo views.py

```
# -*- coding: utf-8 -*-
# Framework formed by functions set Django
__author__ = 'Alicia Jimenez'

from apps.frame import engine
from apps.frame import conversor
from apps.frame import homoframe

from django.shortcuts import render, redirect
from django.http import HttpResponse
import random
from django.contrib import messages
import crypt

iDB = engine.DB()
iUtils = engine.Utils()
iConversor = conversor.Conversor()
iFrame = homoframe.Frame()
iDataset = homoframe.Dataset()

def homepage(request):
    return render(request = request,
                  template_name='home.html',
                  context = {"usuario":request.session.get('email')})

def register(request):
    if request.method == 'POST':
        diccionario={}
        email = request.POST.get('email')
        password1 = request.POST.get('password')
        password2 = request.POST.get('password_confirm')
        if password1!=password2:
            messages.success(request, f"Las contraseñas deben ser iguales, intente de nuevo .. ")
            return render(request = request,template_name='register.html')
        else:
            if existe_email(email):
                messages.success(request, f"El correo ya existe, intente de nuevo .. ")
                return render(request = request, template_name='register.html')
            else:
                val=create_session()
                crypass = crypt.crypt(password2, "22")
                diccionario = {'email' : email, 'password' : crypass, 'session' : 'session'+val, 'parameters': 'parameters'+val}
                iDB.insertar('users',diccionario)
                iDB.insertar('parameters'+val, {'usuario':email})
                iDB.insertar('session'+val, {})
                messages.success(request, f"New account created: {email}")
                request.session['email'] = email
                return render(request = request, template_name='home.html') #, context={'valores':diccionario})

    return render(request = request, template_name='register.html')

def login(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        password = request.POST.get('password')
        crypass = crypt.crypt(password, "22")
        collectionArchive = iDB.obtener_datos_sort('users',{'$and':[{'email':email},{'password':crypass}]},'email')
        for xS in collectionArchive:
            if 'email' in xS:
```

```
            messages.success(request, f"Login successfully: {email}")
            request.session['email'] = email
            return render(request, 'home.html')
         else:
            messages.success(request, f"Email o password incorrecto .. Intenta de nuevo ..")
            return redirect("login")

   return render(request = request,
              template_name='login.html')

def existe_email(email):
   cadena={'email':email}
   collectionArchive = iDB.obtener_datos_sort('users',cadena,'email')
   for xS in collectionArchive:
      if 'email' in xS:
         return True
      else:
         return False

def valida_user(email,password):
   cadena={'$and':[{'email':email},{'password':password}]}
   collectionArchive = iDB.obtener_datos_sort('users',cadena,'email')
   for xS in collectionArchive:
      if 'email' in xS:
         return True
      else:
         return False

def cole_session(email):
   cadena={'email':email}
   collectionArchive = iDB.obtener_datos_sort('users',cadena,'email')
   for xS in collectionArchive:
      if 'session' in xS:
         return xS['session']
      else:
         return ''

def cole_parameter(email):
   cadena={'email':email}
   collectionArchive = iDB.obtener_datos_sort('users',cadena,'email')
   for xS in collectionArchive:
      if 'parameters' in xS:
         return xS['parameters']
      else:
         return ''

def logout(request):
   try:
      del request.session['email']
   except KeyError:
      pass
   messages.info(request, "You logged out")
   return redirect('homepage')

# Function to connect MongoDB and create collections session and parameters (random) for user sesion
def create_session():
   xsession=''
   bd = iDB.connect()
   # list collections to db Clima
   collist = bd.list_collection_names()
   while True:
      # Create user session number (random)
```

```
        x = str(random.randrange(1000))
        xsession = 'session'+x
        if xsession not in collist:
            return x

def index(request):
    server=iUtils.serverxml('http://cecatev.uacj.mx/EstacionX3.xml')
    parameters=cole_parameter(request.session.get('email'))
    dicstations={}
    dicsensors={}
    dicfactors={}
    dicstate={}
    dicdates={}
    collectionArchive = iDB.obtener_datos(parameters)
    for xS in collectionArchive:
        if 'Estaciones' in xS:
            dicstations = xS['Estaciones']
        if 'Sensores' in xS:
            dicsensors = xS['Sensores']
        if 'Calidad' in xS:
            dicstate = xS['Calidad']
        if 'Fechas' in xS:
            dicdates = xS['Fechas']
        if 'Factores' in xS:
            dicfactors = xS['Factores']
    num_visits=request.session.get('num_visits', 0)
    request.session['num_visits'] = num_visits+1
    return render(request, 'index.html',
        context={'x_stations':dicstations,'x_sensors':dicsensors,'x_factors':dicfactors,
              'x_state':dicstate, 'x_dates':dicdates, 'servidor':server, 'session':cole_session(request.session.get('email')),
              'parameters':parameters, 'email':request.session.get('email'), 'visitas':num_visits},)

# Selection list stations the XML
def Estaciones(request, template_name='stations.html', url='http://cecatev.uacj.mx/EstacionX3.xml'):
    if request.method == 'POST': # opcion 2 grabar
        parameters=cole_parameter(request.session.get('email'))
        lista = request.POST.getlist('station')
        dicStations = {}
        dicStations['Estaciones']=lista
        iDB.actualizar('Estaciones',parameters,dicStations)
        return redirect('index')

    # opcion 1 mostrar
    elements = {}
    elements = iFrame.list_stations(url)['elements']
    return render(request, template_name , {'elements': elements})

# Selection list sensors the XML
def Sensores(request, url='http://cecatev.uacj.mx/EstacionX3.xml', template_name='sensors.html'):
    if request.method == 'POST':
        parameters=cole_parameter(request.session.get('email'))
        LSensors = request.POST.getlist('options')
        print('Lista: {}'.format(LSensors))
        listSensor=[]
        dicSensores={}
        dicEstaciones={}
        xestacion=''
        for cadena in LSensors:
            pos1=cadena.find("st:")
            pos2=cadena.find("se:")
            sta=cadena[pos1+3:pos2]
            sen=cadena[pos2+3:-1]
```

```
            if xestacion=='' : # es la primera vez
                xestacion=sta
                listSensor.append(sen)
            else:
                if sta != xestacion: # cambio de estacion
                    dicEstaciones[xestacion]=listSensor #dicSensores
                    listSensor=[]
                    xestacion=sta
                    listSensor.append(sen)
                else:
                    listSensor.append(sen)
            dicEstaciones[sta]=listSensor #dicSensores
        dicSensores['Sensores']=dicEstaciones
        iDB.actualizar('Sensores',parameters,dicSensores)
        iFrame.saveValores(url,parameters,dicSensores)
        return redirect('index')

    parameters=cole_parameter(request.session.get('email'))
    elements = {}
    elements = iFrame.list_sensor(url,parameters)['elements']
    print('sensores: {}'.format(elements))
    return render(request, template_name , {'elements': elements})

# Seleccion de factores de conversion para salida
def Factores(request, template_name='factors.html', url='http://cecatev.uacj.mx/EstacionX3.xml'):
    if request.method == 'POST':
        parameters=cole_parameter(request.session.get('email'))
        LFactors = request.POST.getlist('options')
        print('LFactors: {}'.format(LFactors))
        dicFactor={}
        dicFactores={}
        for cadena in LFactors:
            pos1=cadena.find("fa:")
            pos2=cadena.find("us:")
            fac=cadena[pos1+3:pos2]
            uns=cadena[pos2+3:]
            dicFactor[fac]=uns
        dicFactores['Factores']=dicFactor
        iDB.actualizar('Factores',parameters,dicFactores)
        return redirect('index')

    parameters=cole_parameter(request.session.get('email'))
    elements={}
    elements = iFrame.list_factor(url,parameters)['elements']
    return render(request, template_name, {'elements': elements})

# Selection quality data
def Calidad(request, template_name='quality.html'):
    if request.method == 'POST':
        parameters=cole_parameter(request.session.get('email'))
        estado = request.POST.get('options')
        iFrame.quality_data(parameters,estado)
        return redirect('index')
    return render(request, template_name)

# Selection dates
def Fechas(request, template_name='dates.html'):
    if request.method == 'POST':
        parameters=cole_parameter(request.session.get('email'))
        iFrame.range_date(parameters,request.POST.get('date11'),request.POST.get('date12'))
        return redirect('index')
    return render(request, template_name)
```

```
# Selection output dataset
def SalidaDataset(request, template_name='output.html'):
    if request.method == 'POST':
        session=cole_session(request.session.get('email'))
        parameters=cole_parameter(request.session.get('email'))
        xdataset = request.POST.get('options')
        iDataset.switch_data(parameters,session,xdataset)
        return redirect('index')
    return render(request, template_name )

def GrabaSession(request):
    parameters=cole_parameter(request.session.get('email'))
    session=cole_session(request.session.get('email'))
    iFrame.saveSession('archive',parameters,session)
    messages.success(request, f"Se grabo la colección .. ")
    return redirect('index')
```