Facultad de Ciencias

Desarrollo de un visor de datos espaciales para un nuevo índice de confort térmico.

Development of a spatial data viewer for a new index of thermal comfort.

Trabajo de fin de grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Lucía Cobo López
Codirector: Pablo Fernández de Arroyabe
Codirector: Domingo Gómez Pérez

junio de 2022

Agradecimientos

A mis padres.

Desarrollo de un visor de datos espaciales para un nuevo índice de confort térmico.

Resumen

palabras clave: visor, web, temperatura, salud, confort térmico, clima, biometeorología, Copernicus, netCDF, Python, Django, HTML, javaScript, Leaflet

Los sistemas de información geográfica (SIG) permiten reutilizar la información generada por diferentes sistemas como satélite, información meteorológica, etc. El análisis de estos datos permite ofrecer nuevos servicios mediante técnicas de modelado y visualización de datos basados en la Infraestructura de Datos Biometeorológicos (IDB) del grupo GeoBioMet que recoge datos meteorológicos y predice los cambios en la cantidad de oxígeno en la atmósfera para una localización geográfica concreta.

Este proyecto proporciona información mediante un visor de datos espaciales a través del manejo de datos científicos multidimensionales con librerías de python.

Development of a spatial data viewer for a new index of thermal comfort.

Abstract

keywords: viewer, web, temperature, health, thermal comfort, climate, biometeorology, Copernicus, netCDF, Python, Django, HTML, javaScript, Leaflet

Geographic information systems (GIS) allow the reuse of information generated by different systems such as satellite, meteorological information, etc. The analysis of these data makes it possible to offer new services through data modeling and data visualization techniques based on the GeoBioMet group's Biometeorological Data Infrastructure (BID) that collects meteorological data and predicts changes in the amount of oxygen in the atmosphere for a specific geographic location.

This project provides information with a spatial data viewer through the management of multidimensional scientific data with python libraries.

Índice

Índice de ilustraciones

Índice de cuadros

1 Introducción

El procesamiento de grandes volúmenes de datos tiene una gran importancia en el mundo moderno para poner en perspectiva hechos o tendencias que aparentemente pueden parecer poco relacionadas entre sí. Normalmente se asocia este concepto a grandes corporaciones y empresas privadas que cuentan con los recursos para realizar este tipo de estudios y generar un rendimiento económico a estos datos. Pero no solo las empresas privadas pueden beneficiarse del «Big Data». La extracción de información sobre grandes volúmenes de datos puede extrapolarse a múltiples áreas, siendo una de las más importantes el ámbito científico, más concretamente, el estudio sobre el clima y su relación con la salud.

Debido a la heterogeneidad tanto de los datos disponibles, como de los campos de aplicación de la información, es necesario crear procesos personalizados para su estudio y análisis. En el caso de los datos asociados a variables geográficas se deben tener en cuenta no solo cada uno de los datos captados, sino el contexto que les aporta sentido a cada uno, siendo necesario almacenar y valorar tanto el área de la superficie terrestre que se pretende abarcar en el estudio, como la resolución de los datos en dicho área, las unidades en que se representan cada una de las variables de datos incluidas en el estudio e incluso la altitud a la que se captan dichos datos. Estos datos componen un contexto genérico que puede aplicarse tanto a todo el estudio como a cada variable de datos analizada, pero existen más datos que forman parte del contexto y son necesarios para poder operar con la muestra, lo que aumenta considerablemente la cantidad de datos a analizar, como por ejemplo relacionar cada dato captado con su posición geográfica y gracias a esta etiqueta relacionar las diferentes variables medidas en el mismo punto físico.

A esta complejidad propia de los datos geográficos ha de sumarse el carácter temporal de algunos estudios que tienen que tratar con ventanas temporales muy amplias para poder observar tendencias, lo que aumenta considerablemente la cantidad de datos a analizar. Es decir, pasamos de operar con un dato numérico, a operar con un dato numérico asociado a un punto geográfico y una fecha concreta dentro de un contexto asociado aún más amplio. Por ello, un problema de máximo interés en este área de investigación es cómo procesar automáticamente estos datos.

En este caso, la solución ideal pasaría por implementar un sistema de información geográfica (SIG) orientado a un problema o elemento geográfico específico para el que no existe solución específica, que tras procesar los datos, muestre los resultados en perspectiva, tanto temporal como geográfica, mediante mapas interactivos. Estos mapas deberían mostrar a su vez las diferentes variables de información que recoja el estudio o las soluciones encontradas dentro del contexto para ayudar a su estudio e interpretación por parte de usuarios expertos en la materia. Limitando mediante la distribución libre de un visor web de consulta las necesidades computacionales y técnicas del usuario final de estos datos, que no tiene por qué conocer cómo procesar la información mediante lenguajes de alto nivel.

Este proyecto se ha centrado en la implementación de un visor web de un nuevo índice biometeorológico, diseñado para representar el estrés que genera la persistencia de altas temperaturas para analizar cómo estas afectan a la salud de las personas. Para calcular dichos datos es necesario aplicar este índice de manera programática mediante lenguajes de alto nivel sobre series de datos meteorológicas amplias (espacial y geográficamente) para ser utilizado en proyectos globales siguiendo la tendencia actual del tratamiento y análisis de datos. Para ello se cuenta con los datos del Proyecto Copernicus, la mayor iniciativa europea a nivel mundial en la recolección y distribución de información meteorológica de calidad. El objetivo final del proyecto es facilitar el acceso a estos datos generados de una manera

sencilla, visual e intuitiva que permita a usuarios especializados en la biometeorología su acceso sin la barrera que supone la necesidad de descargar o aprender nuevas herramientas de visualización o manipulación de datos. Nuestra aplicación es de código libre y las visualizaciones pueden ser utilizadas libremente.

2 Marco teórico

Si ponemos el foco en el ámbito científico, más concretamente en la meteorología y sus aplicaciones. Nos encontramos en una situación cuya gran cantidad de datos a procesar supone un reto para los usuarios que más podrían beneficiarse de estos recursos. Estos usuarios ya especializados en los datos, necesitarían volverse expertos en tecnologías y/o herramientas para procesar dichos datos, con el objetivo de analizarlos u operar con ellos a nivel global, para obtener información más amplia en sus estudios.

Esto hace que muchos estudios se limiten a muestras más pequeñas y manejables. Que pueden acabar implicando un mayor esfuerzo por parte del usuario al tener que duplicar procesos manualmente para comparar muestras de diversa procedencia, que validen los resultados para evitar sesgos y reducir los índices de error. Pero sin eliminar realmente ese sesgo dado por el tamaño de la muestra. Si bien es cierto que este tipo de estudios se benefician de controlar mejor los parámetros específicos que intervienen en la muestra seleccionada, pierden la perspectiva global y la información que podría inferirse con una muestra de alcance mayor, tanto temporal como geográfica.

Por otra parte, el alcance de estos estudios puede estar limitado en otros aspectos, como la manipulación de datos. Ya que, si bien existen programas preparados para procesar ciertos formatos de datos y conversores para lograr que los repositorios disponibles tengan el formato adecuado para hacer uso de esas herramientas de consulta y visualización, las operaciones de manipulación o cálculo que se pueden realizar sobre dichos datos y los resultados que se pueden calcular con estos programas predeterminados no siempre se ajustan a la premisa o el enfoque con el que se desearían realizar dichos estudios. Esto supone en muchos casos una necesidad de aprendizaje por parte del usuario al que principalmente le interesa poder interpretar los resultados obtenidos, no conocer el funcionamiento interno de la herramienta. Lo que genera una falta de especialización en los estudios y una falta de control sobre los resultados calculados (tratamiento de situaciones especiales, errores, precisión) ya sea por desconocimiento de la herramienta y complejidad de uso o porque la propia herramienta no está preparada para permitir al usuario tratar dichas excepciones.

Dados los avances tecnológicos actuales y los estudios realizados hasta el momento, es importante poner el foco en perspectivas y estudios con un enfoque global como es el caso Brimicombe et al. [2022], que ayuden a complementar los datos y las bases de conocimiento sobre el medio ambiente ya existentes, aunque para ello sea imprescindible la colaboración directa entre expertos informáticos y expertos en biometeorología que amplíen las opciones existentes.

2.1. Escenario

Este trabajo parte de la idea del profesor Pablo Fernández de Arroyabe, un “cliente” cuyo perfil se ajusta a ese experto en datos, pero no en tecnologías. Cuyo interés, partiendo de estudios tanto propios como ajenos, se ha centrado finalmente en la perspectiva global de los repositorios de datos actuales y el uso práctico que podría hacerse de ellos.

Para ello en los siguientes puntos se van a describir los recursos actuales en los que el cliente sustenta la idea para este trabajo y los recursos de los que desea hacer uso.

2.2. Estudios sobre confort térmico

Comenzando con un repaso a trabajos publicados sobre la relación existente entre la temperatura aparente y el impacto sobre la salud. Encontramos la base para relacionar ciertos parámetros meteorológicos, principalmente las altas temperaturas, con problemas de salud que a priori no parecen directamente relacionados.

Estas variables meteorológicas están definidas en estos estudios mediante fórmulas matemáticas denominadas índices biometeorológicos. Algunos estudios declaran o establecen estos índices mediante experimentos o mediciones en condiciones controladas. Estos índices posteriormente se ponen a prueba para demostrar su validez y precisión, e incluso para replantear y adecuar la fórmula de manera que se ajuste mejor para representar la realidad.

Por ejemplo, en el artículo de Santurtún et al. [2020a], encontramos un estudio que en base a 3 índices de confort térmico en situaciones de baja temperatura, analiza la relación respecto a los ingresos hospitalarios en 3 ciudades españolas y 2 portuguesas, en un periodo de 9 años. Encontrando una tendencia significativa en las muestras de 3 de estas ciudades.

En el caso de Santurtún et al. [2020b], tras ser identificada una variación geográfica en el ratio de suicidios por algunos autores, este estudio evaluó la relación estadística entre el suicidio en Madrid y Lisboa y el índice biometeorológico de la Temperatura Aparente (AT, Apparent Temperature), un índice de confort térmico. El resultado de este trabajo, a falta de evaluar otros factores demográficos que podían influir en el resultado, según indican los autores, mostraba una relación entre el suicidio y el índice de confort térmico en Lisboa. Bajo temperaturas altas, entre otras variables, el riesgo de suicidio incrementaba. Este trabajo tomaba como referencia estudios previos que analizaban la temperatura y la humedad como factores relacionados con las muertes por suicidio.

Es decir, ambos estudios aplicaban funciones matemáticas predefinidas con las que representar el confort térmico y medir su relación con diferentes patologías. Pero los ámbitos de aplicación eran limitados, mientras que la complejidad de los índices dependían de multitud de factores.

Por ello se revisaron los diversos índices biometeorológicos registrados en de Freitas and Grigorieva [2015]. Observando tanto los parámetros utilizados en cada estudio, como la metodología empleada en su validación. Haciendo especial esfuerzo en aquellos que contaban entre sus variables con la temperatura. Estos estudios se basan principalmente en experimentos y mediciones sobre pequeños grupos de voluntarios cuyas variables ambientales estaban estrictamente documentadas y controladas.

Este trabajo, por el contrario, busca un enfoque global que estudie tendencias gracias al gran volumen de datos, por encima de la precisión obtenida de un control férreo sobre las variables ambientales que puedan intervenir en una zona puntual. Siguiendo un enfoque similar al definido en el trabajo Brimicombe et al. [2022] que desarrolla una librería Python para calcular y visualizar algunos de los índices biometeorológicos ya definidos en de Freitas and Grigorieva [2015] a una escala global.

Brimicombe et al. [2022] se limita a proporcionar una herramienta para estas funciones previamente definidas, facilitando su uso en estudios con perspectivas globales. Pero sigue implicando un nivel medio de manejo de Python para su uso, lo que limita el acceso a la herramienta y solo simplifica y agiliza el cálculo y la visualización que un usuario de ese nivel podría hacer por sí mismo sobre los datos mediante lenguaje Python. Sin proporcionar una solución útil a aquellos usuarios que quieren evitar el uso de lenguajes de alto nivel. Además los estudios que requieren de una manipulación de datos, como por ejemplo aquellas que quieran definir un nuevo índice biometeorológico, como es este caso, siguen necesitando de un proceso personalizado implementado mediante dichos lenguajes de alto nivel.

Para ello se propone un nuevo índice biometeorológico de confort térmico que dependa únicamente de datos de temperatura que, siguiendo los requerimientos del cliente, represente el estrés que las altas temperaturas pueden llegar a generar en las personas, sobre todo tras periodos largos de calor. Que posteriormente pueda ser accedido desde un visor web facilitando su acceso y uso a nuevos usuarios sin la necesidad de utilizar herramientas de visualización externas como ArcGIS, que suponen una inversión, económica y de aprendizaje, importante. Ni precisar del conocimiento sobre lenguajes de alto nivel.

Tras revisar de Freitas and Grigorieva [2015] no se encontró un índice reconocido en el que solo

intervinieran las variables de temperatura, ni que reflejara ese estrés procedente de los periodos largos de calor extremo que sobrepasan la capacidad humana (y animal) de adaptación y generan un estrés reflejado mediante posibles problemas de salud como las enfermedades cardiacas estudiadas en Santurtún et al. [2020a]. El cliente desea que se diseñe y genere un nuevo índice biometeorológico que represente ese incremento en el tiempo para poder comparar con series de datos médicas, mediante registros de ingresos hospitalarios o morbilidad diaria. Para comprobar la relación entre el estrés sufrido durante las olas de calor y la salud de las personas.

2.3. Proyecto Copernicus

El cliente mostró su interés en el uso del Proyecto Copernicus para la elaboración de este trabajo.

El proyecto Copernicus, antes conocido como Vigilancia Mundial del Medio Ambiente y la Seguridad (GMES), es la mayor iniciativa mundial en la adquisición de datos meteorológicos. Está coordinada por la Comisión Europea, y en ella participan diversos organismos como la Agencia Espacial Europea (ESA) y la Organización Europea para la Explotación de Satélites Meteorológicos (Eumetsat), además de las diversas agencias meteorológicas de los Estados miembros.

Esta iniciativa tiene el objetivo de recopilar datos terrestres de calidad para entender los cambios ambientales y su impacto, con el objetivo de poder contribuir a la protección del medio ambiente y la salud de los ciudadanos. Mediante la elaboración de un repositorio unificado que permita a la comunidad científica y empresas realizar sus labores con datos accesibles y consistentes para generar un valor añadido a esta información.

Los datos son tomados a través de varios satélites (7 por el momento, llamados “sentinels”), que miden tanto temperaturas como composición del aire, biodiversidad, presencia de aerosoles, movimientos marítimos y otros parámetros ambientales. Estos datos se complementan con los datos recogidos por estaciones en tierra de agencias meteorológicas.

El proyecto Copernicus almacena datos y proporciona información confiable y actualizada sobre el estado del planeta. Estos datos se reúnen en diferentes conjuntos de “servicios” dependiendo de su naturaleza, en este trabajo se ha utilizado el “servicio Climático” (CDS), centrado en repositorios de datos de naturaleza meteorológica. Se trata del servicio más nuevo del proyecto pero cuenta con varios años de desarrollo.

Este servicio es de acceso libre a todo el público y gratuito, esto es importante, ya que algunos servicios están restringidos por cuestiones de seguridad debido a la sensibilidad de los datos. Desde el Proyecto Copernicus se anima tanto al sector privado como a la comunidad científica a utilizar estos datos para generar valor a la gran inversión que suponen estos servicios, pero el acceso a los datos queda en ocasiones fuera del alcance de la comunidad científica, ya que el acceso a estos datos ha de realizarse de manera programática, aunque la plataforma proporciona un asistente para generar el código necesario e instrucciones para la descarga, es necesario registrarse en el servicio previamente.

Además, esta gran cantidad de datos es recopilada en archivos de texto en formato netCDF, un estándar para el almacenamiento y distribución de datos científicos en internet.

2.4. NetCDF

NetCDF o Network Common Data Form es un conjunto de librerías software y formatos de archivo (con extensión “.nc”), cuyo propósito es la creación, el acceso y el intercambio de datos científicos multidimensionales de manera eficiente. Almacenando los datos (variables) en una estructura matricial mutidimensional, cuyos índices se corresponden con las dimensiones que definen dichas variables.

El «Unidata Program Center» mantiene interfaces de programación para la mayoría de lenguajes de alto nivel, entre ellos Python, que es el que se va a utilizar en este trabajo. Como se recoge en la documentación oficial, los archivos de datos en formato netCDF son:

- Auto-descriptivos: incluyen información sobre los datos almacenados.

- Portables: accesibles desde equipos con diferentes formas de almacenamiento de datos (integers, caracteres y números en coma flotante).
- Escalables: Se puede acceder de manera eficiente a subconjuntos de largos volúmenes de datos.
- Anexables: los datos se pueden agregar a un archivo netCDF bien estructurado sin necesidad de generarlo desde cero.
- Compartibles: Lecturas de fichero simultáneas, pero solo un escritor cada vez.
- Archivables: el acceso a todas las versiones anteriores de netCDF será compatible con las versiones posteriores.

Estas características han hecho del formato netCDF un estándar comunitario para compartir datos científicos.

2.5. Problemática actual

Los estudios multidisciplinares son imprescindibles para entender la complejidad del mundo actual, lo que implica utilizar datos de fuentes muy diversas y estudiar sus relaciones. Como se comentaba anteriormente, a pesar de existir un perfil de profesionales expertos en el ámbito de aplicación de los datos capaces de diseñar y dirigir estudios orientados a la biometeorología. Cuando se trata de enfoques globales, la existencia de repositorios de datos solo accesibles y manipulables mediante programación en lenguajes de alto nivel, supone una barrera casi infranqueable a la explotación de los recursos de datos con los que contamos actualmente.

Las aplicaciones existentes para interpretar archivos en formato netCDF como ArcGIS o incluso herramientas para calcular ciertos índices como Brimicombe et al. [2022], no resuelven el problema de calcular nuevos índices biometeorológicos de forma automática. La comunidad científica demanda herramientas que aportan mayor manejabilidad sobre los datos, pero esto supone una inversión económica elevada. Incluso mayor si se quiere aportar un desarrollo personalizado que implique fuentes de datos heterogéneas (clima, salud, meteorología) para representar o medir una realidad concreta.

Es decir, modificar o calcular datos necesariamente debe realizarse de manera programática. La implementación del nuevo índice biometeorológico dependiente de las temperaturas que el cliente desea implementar debe realizarse mediante un lenguaje de alto nivel para el alcance físico y temporal (al menos 30 o 35 años) sobre el que el cliente desea que se calcule. A su vez, es importante facilitar los resultados calculados en un formato sencillo y accesible para ser visualizado posteriormente, eliminando esa limitación provocada por los lenguajes de alto nivel, las herramientas actuales y los formatos de datos no legibles.

2.6. Solución propuesta

Siguiendo los requerimientos del profesor encargado Pablo Fernández de Arroyabe como cliente, se propone desarrollar un visor web que presente los datos de un nuevo índice biometeorológico calculado con datos de temperatura del proyecto Copernicus. Con el objetivo de proporcionar un portal web de consulta para el posterior estudio de los datos por parte de personas especializadas de manera rápida, libre y sencilla.

Este trabajo describe una solución personalizada desde la descarga de los datos desde el proyecto Copernicus en formato netCDF, pasando por la personalización y procesamiento del dataset original en lenguaje Python, para generar nuevos datos dentro del periodo de interés, unos 35 años, según el índice biometeorológico diseñado; hasta la creación de un visor web como interfaz de usuario para el estudio y distribución de los datos a un público no especializado en programación pero sí en datos meteorológicos.

3 Objetivos

3.1. Objetivo general

Mostrar la relación existente entre las olas de calor y la salud, mediante la implementación de un nuevo índice meteorológico, denominado ICA, basado en las temperaturas máximas y mínimas para reflejar el estrés al que se somete el cuerpo humano durante periodos largos de calor. Representar los datos obtenidos de manera sencilla y visual para el usuario final.

3.2. Objetivos específicos

- Investigar sobre el proyecto Copernicus, el formato netCDF y sobre los estudios actuales que indican una relación entre las altas temperaturas y diversos problemas de salud, validando así la utilidad de plantear un nuevo índice sobre este tema.
- Revisar los índices biometeorológicos actuales para comprobar que la solución propuesta no está recogida oficialmente, por lo que aún no se ha puesto a prueba su utilidad o corrección.
- Desarrollo de los scripts de manipulación de datos.
- Generación de los nuevos datasets en función del índice biometeorológico diseñado.
- Creación de un proyecto Django para la implementación de un visor web para la distribución de los datos calculados.

3.3. Estructura del trabajo

Este trabajo se va a componer de tres partes clave:

- Captación y análisis de requisitos del cliente.
- Desarrollo de la aplicación siguiendo el proceso de ingeniería de software.
- Demostración de la aplicación y conclusiones.

3.4. Tareas

1. Captación y análisis de requisitos del cliente: Pablo Fernández de Arroyabe, uno de los profesores encargados de este TFG y experto en Geografía Física, ejerce el rol de cliente en este proyecto. Los requisitos son captados y discutidos mediante reuniones telemáticas realizadas de manera continua a lo largo de la realización de este trabajo.
    - Análisis y estudio sobre el área de aplicación de este trabajo.
    - Análisis y estudio sobre los recursos disponibles y su utilidad: Proyecto Copernicus y el dataset a utilizar.
    - Diseño y desarrollo del nuevo índice biometeorológico.

2. Desarrollo de la aplicación siguiendo el proceso de ingeniería de software: diseño de algoritmos y diagramas para la organización del trabajo a realizar.
   - Selección del dataset y análisis del formato de los datos: búsqueda de un dataset que incluya al menos datos de temperatura máxima y mínima diaria por punto geográfico con un periodo de tiempo disponible de varios años.
   - Descarga de los datos y selección del periodo de tiempo deseado para generar el índice biometeorológico, almacenamiento de las variables modificadas en un archivo netCDF para su posterior procesamiento.
   - Cálculo del nuevo dataset en función del índice biometeorológico definido para medir la acumulación de calor.
3. Demostración de la aplicación y conclusiones.
   - Desarrollo de un portal web como interfaz de usuario.
   - Mapeo de temperaturas en función del índice biometeorológico mediante la librería Leaflet.
   - Habilitar la descarga desde la web de los datos calculados.

3.5. Herramientas

Software:

- Repositorio de datos: Bitbucket.
- Sistema operativo: Windows 10. Se decidió utilizar este sistema operativo pensando en un administrador no especializado que pudiera generar y procesar los datos en un equipo doméstico sin necesidad de máquinas virtuales o especializarse en la lógica del sistema.
- Lenguaje gestión de datos: Python versión 3.6.5. Este lenguaje suponía un requisito inicial para la elaboración de este trabajo.
- Librerías Python:
  - «netCDF4»: para el formato de datos.
  - «cftime»: para la gestión de datos temporales, recomendado por la librería «netCDF4».
  - «numpy»: librería utilizada para operaciones con matrices seleccionada por su amplia utilización tanto en otras librerías como en la librería «netCDF4».
  - «geojson»: utilizada para la creación de ficheros compatibles con leaflet.
- Framework web: Django versión 3.2.10 para la implementación del sitio web.
- Lenguajes visor web: Html5 y JavaScript.
- Librerías JavaScript: «Leaflet» y «leaflet-ajax» para la organización del visor web y la lectura de datos.

Hardware:

- Memoria virtual ampliada a 20GB para evitar problemas de memoria al alojar datos extensos en variables python.
- Intel(R) Core(TM) i7-6700 CPU con 8GB de RAM, 4 núcleos físicos y 8 lógicos.

4 Requerimientos y requisitos

El cliente tiene un perfil científico y desea un visor web cuyo contenido principal es la visualización de series de datos modificadas que no puede encontrar en otro lugar. Para ello desea que la autora de este trabajo cree por sí misma el índice a partir de unas ideas básicas con las que ha estado trabajando anteriormente y consultado en varios estudios, pero no ha podido poner en práctica debido a la necesidad de utilizar lenguajes de alto nivel, con los que no se siente familiarizado.

A partir de esta premisa y algunas sesiones de estudio para la familiarización con el ámbito de trabajo en el que nos encontramos se han definido unos requisitos básicos (1 y 2) mediante reuniones telemáticas en las que se puso en común las ideas desarrolladas a partir de las directrices iniciales del cliente.

4.1. Requisitos funcionales

REQUISITOS FUNCIONALES	**CÓDIGO**
Calcular el índice biometeorológico y los datos necesarios a partir de la temperatura.	RF01
Crear el visor web de un índice biometeorológico.	RF02
Permitir descargar datos en base a diferentes parámetros desde la web.	RF03

Cuadro 1: Resumen requisitos funcionales

- RF01: Se busca implementar un índice de calor que solo precise el uso de las temperaturas máximas (tx) y mínimas (tn). Para ello presenta la idea del estrés que generan los periodos largos de tiempo con temperaturas elevadas en la salud de las personas y la idea de cuantificar matemáticamente este incremento respecto a un punto de referencia. El punto de referencia se sitúa en los percentiles superiores de las series de datos de temperaturas máximas y mínimas. La horquilla recomendada por el cliente está entre el percentil 95 y el 98. Además del índice de calor acumulado (ICA) calculado, el fichero final almacenará también el percentil y algunos datos derivados de temperatura para proporcionar información global sobre los datos con el fin de ampliar su contexto.

- RF02: Se define el resultado final como un mapa web interactivo que muestre los datos del índice día a día. De esta manera se evita el uso a priori de lenguajes de alto nivel y aplicaciones de terceros para visualizar los datos.

- RF03: Dado que se desea trabajar con un gran volumen de datos, se desea poder delimitar unos parámetros de descarga para poder descargar subconjuntos de datos más manejables, de manera automática.

REQUISITOS NO FUNCIONALES	CÓDIGO
Utilizar datos del proyecto Copernicus a través de la librería CDS del dataset E-OBS.	RNF01
Incluir los derechos sobre los datos originales (legal).	RNF02
Utilizar el lenguaje Python para procesar los datos.	RNF03
Utilizar la librería Leaflet para crear el visor.	RNF04
Automatizar el proceso lo máximo posible para el mantenimiento del visor.	RNF05

Cuadro 2: Resumen requisitos no funcionales

4.2. Requisitos no funcionales

- RNF01: El cliente desea utilizar los datos del proyecto Copernicus, concretamente los datos del repositorio E-OBS, pertenecientes al servicio de cambio climático CDS.
- RNF02: Los datos son propiedad del proyecto Copernicus y pueden estar sujetos a limitaciones de uso que deben ser cumplidas.
- RNF03: Los datos de este repositorio están almacenados en formato netCDF. Cuenta con soporte en los principales lenguajes de alto nivel, siendo requisito para este trabajo el uso concreto del lenguaje Python.
- RNF04: La librería de javascript Leaflet es de uso libre, ampliamente documentada y desarrollada. No cuenta con limitaciones de uso, como sería el caso de la libería de Google maps, por lo que ha sido recomendada por los profesores para su uso.
- RNF05: Desde la descarga de los datos, pasando por su cálculo y actualización, se desea que el sistema sea lo más adaptable posible a cambios en el futuro, debido a que el cliente final no cuenta con experiencia en programación y el futuro administrador web no tiene por qué tener conocimiento de la lógica interna del proyecto.

5 Análisis y diseño

Como se ha podido observar en los requisitos funcionales, la premisa es sencilla: A partir de una serie de datos, se desea aplicar una fórmula para generar un dataset más detallado, al que después referenciar desde una aplicación web para su visualización y descarga.

La complejidad en el cálculo de datos deriva del volumen de datos, el formato y el diseño del índice en sí mismo. Mientras que la complejidad de la implementación del visor web radica en el uso de la librería «leaflet» y la conexión con los datos disponibles.

5.1. Arquitectura

La arquitectura de este proyecto está basada en el modelo de tres capas, organizando la estructura del proyecto en tres niveles lógicos: el nivel de presentación mediante el visor web; el nivel de aplicación, con el procesamiento de datos en python, y el nivel de datos, almacenados tanto en ficheros netCDF como geoJson.

5.2. Casos de uso

El usuario final no va a interactuar con el cálculo de los datos, consultará el visor web y se moverá por las dimensiones del mapa: tiempo, latitud y longitud, para visualizar los datos calculados, además de ser capaz de descargar los datos para su uso personal. Como se representa en el siguiente caso de uso de usuario (1):

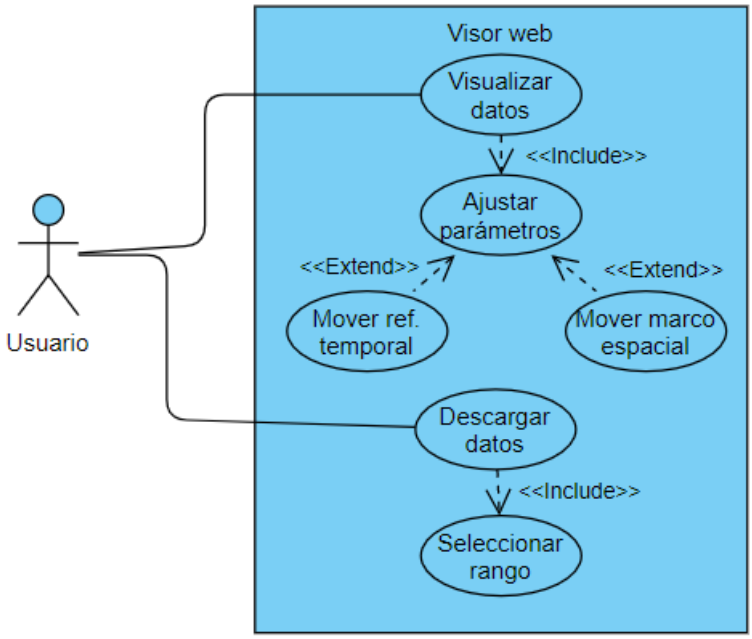

Ilustración 1: Casos de uso de usuario

- Escenario: Visor Web del índice biometeorológico calculado.
- Visualizar datos (RF02): El usuario puede ajustar los parámetros de visualización del índice respecto al tiempo (mover referencia temporal), la latitud y longitud (mover referencia espacial).
- Descargar datos (RF03): el usuario puede seleccionar un rango temporal y espacial para la descarga de datos.

5.3. Desarrollo del proyecto

La metodología de desarrollo que se ha elegido para este proyecto es el desarrollo en cascada, por varios motivos. Al estar el equipo formado por un único desarrollador y cliente, era el más sencillo de implementar. Al tener acceso al cliente de forma continua se han podido corregir errores en diseños de forma efectiva en cada una de las reuniones semanales. Además, el cliente está familiarizado con el desarrollo de otros productos similares, ya que dirige actualmente un doctorado con un enfoque similar, lo cual ha permitido un desarrollo lineal en este trabajo.

En la siguiente ilustración se puede apreciar el diagrama de Gantt que describe la carga temporal de este proyecto (2).

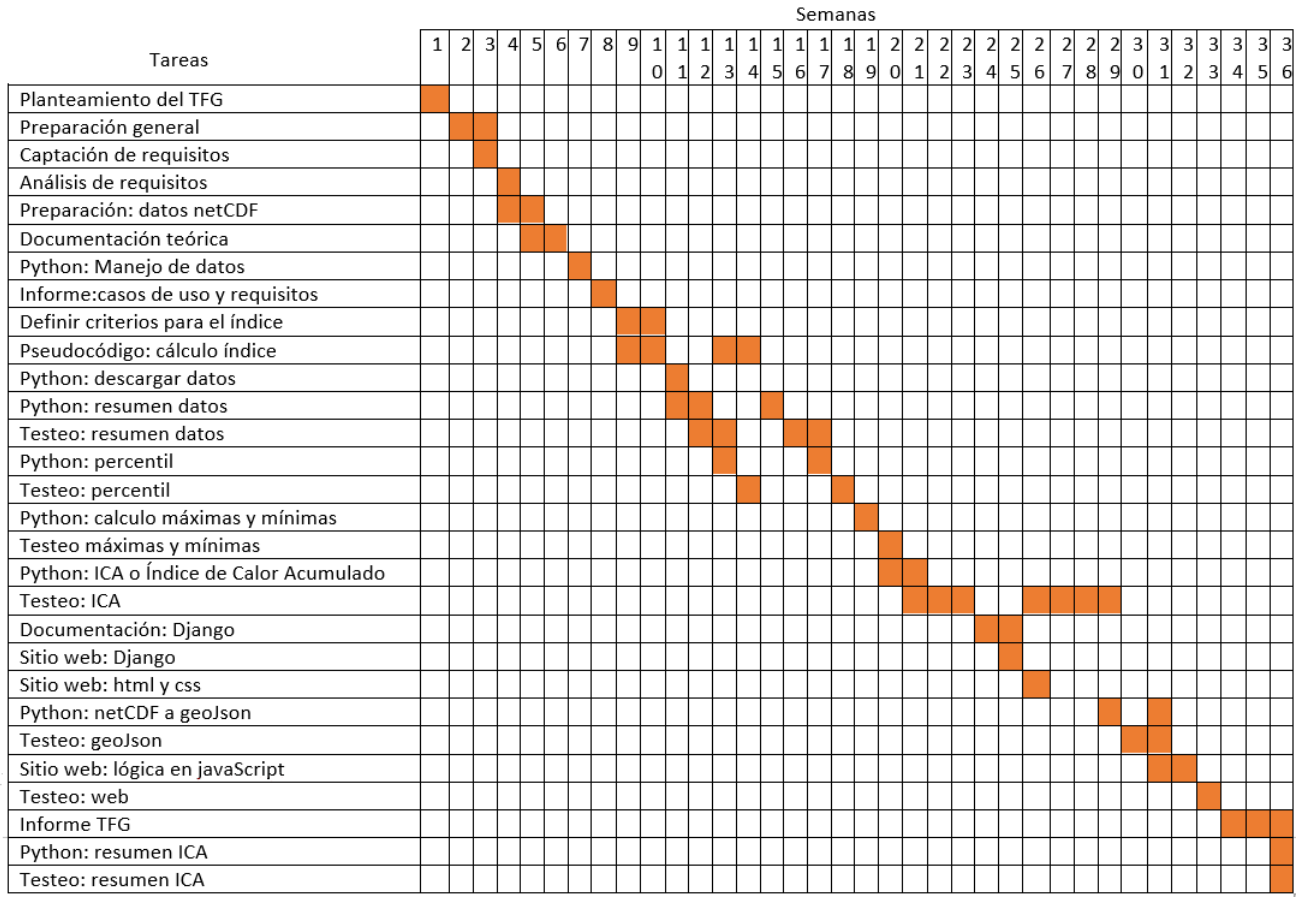

Ilustración 2: Diagrama de Gantt

5.4. Mockups

Ilustración 3: Caso de uso. Visualizar datos.

Ilustración 4: Caso de uso. Descargar datos.

En las ilustraciones 3 y 4 se representa la vista de usuario definida en el apartado anterior, 5.2.

Siguiendo una estructura básica, la vista de usuario se centra en la página de inicio. Esta página de inicio incluye una representación visual de los datos mediante un mapa interactivo con un pequeño formulario para cargar el mapa diario que se desee. Seguido a continuación por un formulario para la descarga de los datos del índice biometeorológico en formato netCDF.

Los botones que referencian a vistas como “Acerca de” o “Legal” no se han representado mediante un mockup, ya que incluirán información básica sobre el proyecto en texto plano, pero manteniendo la estética y distribución de la página principal.

Se han representado las vistas con estos colores para crear un contraste entre la paleta de colores del mapa de calor (amarillos a rojos) sobre fondo blanco respecto a azules violáceos y negros. Además de seguir la tendencia de las webs actuales de utilizar un fondo oscuro (modo nocturno) para mejorar la experiencia de usuario.

5.5. Pseudocódigo

Siguiendo los requisitos del cliente se define y diseña el nuevo índice biometeorológico de confort térmico como un “Índice de Calor Acumulado” o “ICA”. Este índice representa la acumulación de las temperaturas por encima del percentil 95 para las temperaturas máximas y mínimas durante períodos continuos de calor. Como se define en la tabla 5.1.

Nomenclatura empleada:

- «tx»: temperatura máxima.
- «tn»: temperatura mínima.
- «txPercentil95» o «tnPercentil95»: el prefijo se refiere al elemento (temperatura máxima o mínima) al que pertenece el percentil calculado y el sufijo (dato numérico) al percentil concreto calculado.
- «ICA»: índice de calor acumulado, el dato numérico calculado a partir de las temperaturas.

Listing 5.1: Fórmula del índice de calor acumulado.

```
#día actual = n+1
ICA[n+1]=
    suma(máximo(tx[n+1]-txPercentil95,0),máximo(tn[n+1]-tnPercentil95,0))
if ICA[n+1]> 0:
    ICA[n+1]=ICA[n+1]+ICA[n]
```

Estas variables de datos dependen a su vez de tres variables: tiempo, latitud y longitud, con el doble papel de ejercer como variables de datos por sí mismas con sus valores almacenados en matrices cuyos índices actúan además dimensiones para las variables de datos. En el caso del percentil, este dato se calcula sobre toda la serie temporal, midiéndose cada valor sobre dos dimensiones: latitud y longitud. Por ello se ha definido el siguiente pseudocódigo que desglosa la fórmula anterior aplicándola a toda la serie de datos:

Listing 5.2: Cálculo del índice de calor acumulado.

```
def indiceCalorAcumulado2(tx, tn):
    txPercentil95[i,j]=percentile(tx[:,:,:], 95, axis=0)
    tnPercentil95[i,j]=percentile(tn[:,:,:], 95, axis=0)
    for n in range(len(tx[:,0,0])): #tiempo
        for i in range(len(tx[0,:,0])): #latitud
            for j in range(len(tx[0,0,:])): #longitud
                calorAcumulado[n,i,j]=0.0
                diaCalor=False
```

```
                if tx[n,i,j]>=txPercentil95[i,j]:
                    calorAcumulado[n,i,j]+=(tx[n,i,j]-txPercentil95[i,j])
                    diaCalor=True
                if tn[n,i,j]>=txPercentil95[i,j]:
                    calorAcumulado[n,i,j]+=(tn[n,i,j]-tnPercentil95[i,j])
                    diaCalor=True
                if diaCalor==True:
                    calorAcumulado[n,i,j]+=calorAcumulado[n-1,i,j]
    return calorAcumulado
```

Se calculan los percentiles para las temperaturas máximas (tx) y mínimas (tn), en cada punto geográfico, el axis 0 indica que lo calcula respecto a la primera dimensión, el tiempo.

Se calcula el índice de calor para cada día por coordenada (dimensiones: tiempo, latitud y longitud). Si la temperatura máxima supera el percentil 95, se calculan los grados por encima de dicho percentil, se realiza la misma operación para la temperatura mínima. Esto dará un resultado negativo si están por debajo del percentil y cero o positivo si están por encima, si el resultado es negativo se sustituye por cero (máximo entre cero y el valor calculado).

Una vez calculados estos resultados normalizados, se suman ambos resultados, los grados por encima del percentil para la temperatura mínima y máxima. Si esta suma es mayor a cero, se agrega la temperatura acumulada del índice del día anterior, valor comprendido entre 0 y un número positivo.

En el caso de que la temperatura máxima no supere el percentil, pero la mínima sí, se considera que existe una noche tropical que puede mantener el calor acumulado de un día previo, por lo que cuenta como una prolongación del estrés acumulado por las temperaturas altas. Nuestro enfoque puede prolongar la racha de calor, aunque la temperatura máxima no supere el percentil que sería el requisito usual que haría que se considerara un día de calor extremo en otras circunstancias.

Se ha decidido utilizar el percentil 95 para el cálculo debido circunstancias surgidas durante el desarrollo que se detallarán más adelante, la idea inicial dada por el cliente es utilizar un porcentaje alto, en la horquilla entre el percentil 95 y el 98, que es lo que se considera generalmente como un día de calor extremo respecto a las temperaturas máximas. El ICA ha sido diseñado con el objetivo de analizar posteriormente su idoneidad, por lo que no hay una solución "correcta".

Para poner en perspectiva la decisión de definir así el índice, en la siguiente tabla podemos ver una simulación sobre como se calcularían los datos aplicando el ICA propuesto en un ejemplo concreto 3.

PercentilTx	27
PercentilTn	14

Día	**tx**	**tn**	**ICAn, ICAn+1: tx-ptx , tn-ptn**
1	27	12	...+0+0=0
2	26	16	0+0+2=2
3	28	14	2+1+0=3
4	29	15	3+2+1=6
5	28	13	6+1+0=7
6	26	16	7+0+2=9
7	25	13	0+0=0 (no se suma el ICAn)
8	28	14	0+1+0=1
9	26	16	1+0+2=3

Cuadro 3: Ejemplo cálculo índice de calor acumulado (ICA)

5.6. Características del dataset E-OBS

Estas son algunas de las carácteristicas del dataset E-OBS seleccionado por el cliente (RNF01) dentro del servicio climático (CDS) del Proyecto Copernicus 2.3:

- Cobertura geográfica de los datos: Europea.
- Altura a la que se recogen los datos por defecto: Cerca de la superficie.
- Datos disponibles desde: Enero de 1950 al presente, aunque el programa diseñado seleccionará los últimos 35 años registrados.
- Frecuencia de los datos: diaria.
- Formato de los datos: NETCDF-4
- Convenciones en el formato de los datos: Climate and Forecast Metadata Convention v1.4 (CF-v1.4)
- Actualizaciones: versiones nuevas cada 6 meses.

Las opciones utilizadas para la descarga del dataset en este trabajo:

- Tipo de producto seleccionado: «Ensemble mean».
- Variables de interés seleccionadas: «Maximum temperature» y «Minimum temperature».
- Resolución seleccionada: grid de «0.1 deg».
- Periodo seleccionado: «Full period».
- Versión seleccionada: La última disponible.
- Formato seleccionado: ejemplo implementado con formato «Zip» en el repositorio de este trabajo, pero no es obligatorio.

Para acceder a estos datos es necesario tener una cuenta de usuario gratuita en el servicio Climático del Proyecto Copernicus.

El proceso de descarga y actualización de los datos de manera automatizada no es posible debido a la necesidad de acceder de manera activa a una cuenta personal en el servicio climático antes de iniciar la descarga. Al iniciar sesión se genera una clave necesaria para que se inicie el proceso de descarga.

La descarga debe realizarse mediante código python, el asistente web genera el código necesario para la descarga de los datos según las opciones seleccionadas.

Siguiendo la ruta "./Codigo/calorAcumulado/calorAcumulado/codigos_python_ICA/datos_originales" respecto al directorio raíz de este trabajo, se puede encontrar un script llamado "descargarDataset.py". Este script contiene a modo de ejemplo el código que genera el asistente para descargar los datos del repositorio. Este script también contiene una función para descomprimir dichos datos, aunque pueden ser descomprimidos a mano.

Para que la solución desarrollada funcione correctamente es necesario descargar el dataset para todo el periodo temporal, aunque solo queramos actualizar los datos. Y actualizar el nombre de los dos archivos de datos en el script de tratamiento de datos "resumenDatos.py" ubicado en "./Codigo/calorAcumulado/calorAcumulado/codigos_python_ICA', no se pasan como parámetro de consola de comandos para evitar que el administrador no sepa qué versión se ha utilizado para calcular el índice, al cambiarlo sobre el script queda el registro de la última versión de datos calculada, sin la necesidad de acceder a los propios datos.

5.7. Leaflet

Como se comentaba en la introducción, una de las mejores maneras de representar este tipo de datos geográficos es mediante mapas interactivos. Partiendo de la idea de una distribución libre de los resultados de este proyecto mediante un visor web, se buscó la información sobre librerías JavaScript de mapas interactivos, encontrando finalmente Leaflet.

Leaflet es una librería JavaScript de mapas interactivos, open-source, de uso gratuito y compatible con la mayoría de plataformas y motores de búsqueda.

Ya que no cuenta con limitaciones de uso gratuito, a diferencia de otras librerías como «Google maps», es una opción bastante utilizada y la seleccionada para este trabajo.

Dispone de múltiples plugins desarrollados por terceros que aumentan sus posibles usos. Cuenta también gran cantidad de opciones de personalización, desde la plantilla del mapa hasta la modificación de punteros. Además de contar con documentación útil, sencilla y descriptiva para iniciarse en la utilización de los diferentes elementos de la librería.

6 Descripción de la solución propuesta

Las solución implementada en este proyecto se recoge en este repositorio git y está disponible para su descarga.

6.1. Primeros pasos

A continuación voy a describir paso a paso las acciones realizadas, íntimamente ligadas a los requisitos funcionales y no funcionales definidos anteriormente (4.1). Implementados temporalmente siguiendo la cronología mostrada en el diagrama de Gantt (5.3).

Antes de empezar se comenzó con una documentación exhaustiva sobre el repositorio de datos a utilizar (5.6) y la importancia del proyecto Copernicus en el panorama Europeo. La elección del repositorio fue tomada por el cliente (RNF01), experto en la materia. Este indicó la utilidad de asistir a un webinar oficial sobre el Proyecto Copernicus y su uso, al cual se asistió de manera telemática y se complementó con otros recursos recogidos en la web de la iniciativa.

Simultáneamente, el cliente proporcionó algunos trabajos como Santurtún et al. [2020a] o Santurtún et al. [2020b] para proporcionar una visión más concreta sobre su idea para el cálculo del índice biometeorológico basado en las temperaturas altas extremas y su utilidad, a partir del cual se diseñó la idea del índice biometeorológico de calor acumulado o ICA descrita en la sección 5.5.

6.2. Cálculo del ICA

Se continuó por el desarrollo de los scripts Python que para el cálculo de los datos en función del ICA diseñado a partir del repositorio original en formato netCDF (5.6).

Los archivos finales generados en esta sección se ubican en el directorio "./calorAcumulado/calorAcumulado/codigos_python_ICA" representado como directorio raíz "./" en esta sección para agilizar la lectura. Se ha separado este punto en subsecciones para describir el desarrollo y la lógica de cada script.

6.2.1. Resumen de datos

El directorio "./datos_originales" contiene (o debe contener, en el caso de actualizaciones) los ficheros de datos originales. El archivo "descargarDataset.py" ubicado en "./datos_originales" es un script de ejemplo para la descarga de datos, contiene las directrices para llevar a cabo la descarga y el código generado por el asistente con los parámetros utilizados para este trabajo a modo de ejemplo para futuras actualizaciones. Este no es un proceso automatizable debido a que es necesario una cuenta de usuario para la autentificación e instalar algunos recursos en la máquina que desee realizar la descarga.

Una vez descargado el dataset original, se procede a manipular los datos mediante código Python y la librería netCDF4 .

Al principio, se decidió utilizar un periodo de estudio de 35 años para el trabajo. Por lo que se creó el script "./resumenDatos.py" que resume la serie de datos original almacenando en un nuevo netCDF solo los datos dentro del periodo de interés.

A continuación se procedió al cálculo de los percentiles para el posterior cálculo del ICA. Pero debido a la gran cantidad de datos se produjeron diversos fallos al ejecutar el código Python, para subsanarlos se optó por ampliar la memoria virtual a 20GB (tamaño aproximado del dataset) y se decidió limitar la cantidad de datos que se iban a procesar aplicando una sencilla idea: Si el índice biometeorológico solo va a analizar los datos más elevados de temperaturas para representar el confort térmico solo en ese aspecto, podemos bajar el valor del percentil a calcular al más bajo de la horquilla (percentil 95) y calcularlo sobre la serie de datos del periodo junio-septiembre de la muestra de 35 años.

Para ello se actualizó "./resumenDatos.py" con los nuevos umbrales temporales. Este resumen es posible dado que el trabajo se centra en la zona geográfica europea, por lo que el periodo de calor está delimitado a unos meses concretos. Por lo que aunque el código funcionaría con datos de otras zonas geográficas, es necesario tener en cuenta que los cálculos de otras zonas con los parámetros temporales que maneja este trabajo no tendrían sentido, deberían ser modificados o utilizar la serie temporal completa en el caso de estudios globales. En el caso de emplear la serie temporal completa dentro del periodo habría que ajustar los cálculos para que la memoria virtual no se sobrecargue generando latencias.

Otra solución podría ser bajar la resolución de los datos (grid a 0.2 grados), para trabajar con la mitad de puntos geográficos, pero no se ha realizado en este trabajo ya que el cliente deseaba que se empleara el grid de 0.1.

El script "./resumenDatos.py" lee los datos del directorio "./datos_originales" que contiene el fichero de temperatura máxima (tx) y el fichero de temperatura mínima (tn) en formato netCDF, recorriendo los datos temporales a partir de la fecha inicial de referencia para almacenar las fechas útiles en el nuevo dataset. Para cada punto temporal válido se almacena toda la sección geográfica, logrando así una escritura en disco balanceada con la memoria virtual utilizada para evitar un incremento exponencial debido a la paginación excesiva.

Antes de ejecutar este script es necesario actualizar el nombre de los ficheros de datos de lectura originales, ya que al no poder automatizar la descarga tampoco se puede automatizar el nombre del fichero de datos que se genera. No se incluyen como parámetro a introducir por consola para evitar errores del administrador, al hacerlo sustituyendo el código quedará registrado en el script cuál ha sido el último archivo utilizado para generar el dataset.

Los problemas detectados durante la ejecución del script "./resumenDatos.py" respecto al uso de memoria y eficiencia fueron revisados mediante «breakpoints» en zonas de interés y mediciones de tiempos de ejecución respecto a muestras más pequeñas y controlables. A su vez, los errores observados debidos al formato de datos y sus peculiaridades fueron revisados mediante la lectura del fichero de datos netCDF generado y el contenido almacenado en sus variables desde consola Python.

Tras la resolución de los errores detectados se genera el primer fichero de datos netCDF intermedio "./datos_temporales/dataJunioSeptiembreUnmasked.nc". Los ficheros netCDF son anexables pero ya que por temas de eficiencia el código se debía dividir en diferentes scripts modulares y para simplificar la detección de errores y ahorrar tiempos de ejecución y pruebas, se optó por ir generando ficheros temporales en cada script del cálculo de datos.

Durante las pruebas con este primer fichero de datos calculado respecto al original se detectó entre otras cosas, que los datos originales estaban almacenados de la manera más eficiente posible, como enteros, pero la representación los mostraba como números decimales y a los valores nulos como símbolos "-". Esto implicaba que el fichero guardaba las temperaturas como enteros con una escala 0.01. Lo cual generó problemas hasta que se detectó la diferencia entre los valores almacenados y los representados. Por defecto, netCDF aplica una máscara a los datos para camuflar los valores no válidos que contiene cuando se consulta y realiza el paso del entero al dato en coma flotante solo para representarlo, es decir, el valor visualizado y el almacenado no es el mismo. En ese sentido, netCDF está más enfocado a la consulta que a la escritura, por lo que se pueden generar errores al manipular los datos y crear copias de dichas subsecciones si no se conocen las peculiaridades de este formato al manipular los datos como se ha realizado en este trabajo .

Para evitar errores de formato, tiempo de procesamiento extra y mantener la eficiencia del fichero,

se decidió eliminar el offset original de los datos, almacenando las unidades como centésimas de grado e indicándolo en los metadatos. Además de deshabilitar la máscara al principio de cada script para trabajar directamente con los valores y no con su representación, para tener mayor control sobre los datos.

Para cumplir con el RNF02 en el nuevo fichero de datos generado "./datos_temporales/ dataJunioSeptiembreUnmasked.nc" se crea una copia de todos los metadatos originales y los que se han modificado al eliminar la escala. Se añade un parámetro extra a los metadatos llamado "ModifiedBy" con mis datos.

Este script tiene un tiempo de ejecución aproximado en mi equipo de 70s.

6.2.2. Percentiles, máximas y mínimas

Una vez calculada la versión final del resumen de datos, se desarrolló el script "./percentilMinMax.py", que ha de ser ejecutado a continuación. En este script se hace uso de la librería "numPy" para calcular nuevas variables netCDF respecto a las temperaturas resumidas en el anterior script y almacenadas en un nuevo fichero de datos. Con esta librería se calcula el percentil 95 para las temperaturas máximas (perTx95) y mínimas (perTn95), necesario para el cálculo del ICA. También se calculan la temperatura máxima para las temperaturas máximas (maxTx) y mínimas (maxTn), y las temperaturas mínimas también para las series de datos máximas (minTx) y mínimas (minTn).

Para calcular estos datos es necesario limpiar las variables de temperatura almacenadas en el dataset "./datos_temporales/dataJunioSeptiembreUnmasked.nc" de valores de relleno (-9999 generalmente, establecido por defecto) para evitar falsas temperaturas mínimas y máximas, para ello se sustituye en la serie de datos el valor por "None" mediante un "bucle for comprimido", ya que al intentar hacer uso de la librería "numPy" con la función "where()" se generaba un problema de memoria por el gran tamaño de los datos que se proporcionaban como parámetros. Por lo demás, al usar funciones predeterminadas, solo fue necesario comprobar que las matrices de datos estuvieran bien definidas en la función y observar los resultados para descartar posibles errores.

Por último, es necesario crear de nuevo una copia de los metadatos en el nuevo fichero de datos netCDF "./datos_temporales/dataPercMaxMin.nc". Las nuevas variables de datos no tienen un original de quien obtener las descripciones desde sus metadatos, se deben definir un ".standard_name", ".long_name" y ".units" para cada una de las variables, además de crear una copia para las variables ya incluidas en el anterior dataset como se realiza en el anterior script.

Este script tiene un tiempo de ejecución aproximado en mi equipo de 5 min.

6.2.3. Índice de calor acumulado. Primer paso

A continuación ha de ejecutarse el script "./IndCATxTn.py". Este script calcula la primera parte del cálculo del índice de calor acumulado (ICA o "indCA" como se identifica en la variable netCDF por convención) y deja pendiente el último "if" definido en el pseudocódigo 5.1 para el siguiente script. Es decir, calcula la diferencia entre la temperatura diaria y el percentil correspondiente para los datos de temperatura máxima (tx) y mínima (tn). Para estos dos valores, calcula el máximo entre el valor calculado y cero, quedándose con el mayor positivo de ambos datos. Finalmente suma estos dos valores, almacenando un valor positivo o cero.

Como se ha comentado, aún faltaría evaluar si han existido temperaturas altas anteriores que sumar a este valor para registrar la acumulación de temperaturas, es decir, el "if" definido en 5.1, que queda pendiente de ser calculado en el siguiente script por motivos de eficiencia. Además de esta condición, derivado del cambio en "./resumenDatos.py" que resume las fechas válidas a 4 meses por año de la serie de datos, lo que provoca que las fechas no sean continuas, queda pendiente evaluar que dicha condición solo se aplique si el día anterior almacenado en la serie de datos es consecutivo al actual.

La división en el cálculo del ICA se debe a que los datos son muy pesados en memoria, lo que provoca una paginación excesiva. Se ha conseguido bajar el tiempo de ejecución gracias a técnicas que empeoran la legibilidad del código, como la reutilización de variables, que minimizan la cantidad de

datos alojados en memoria. Se ha buscado a su vez un equilibrio entre el uso de la memoria virtual y el cuello de botella que genera la lectura/escritura en disco.

Aun así, existe un límite a partir del cual la paginación hace que el código no sea eficiente y el tiempo de ejecución aumente de manera exponencial. Las soluciones que proporciona python para realizar una gestión de memoria virtual por parte del usuario para este tipo de casos es el uso de la librería "gc", pero esto no solucionó el problema de eficiencia, posiblemente debido al sistema operativo utilizado. Por lo que se ha optado por esta solución de dividir el proceso en secciones aparentemente eficientes para el volumen de datos trabajado, estable gracias a "./resumenDatos.py".

Es posible que para una cantidad de datos más extensa esta solución entre en conflicto con el umbral de datos que es capaz de procesar el equipo antes de entrar de nuevo en un proceso de paginación poco eficiente, que haga necesario dividir el proceso por bloques de datos (o puntos geográficos) y no la lógica del programa. Esta solución implicará invertir un mayor esfuerzo en el control de datos, y en la recombinación de todos los ficheros generados con el índice calculado. Una solución tenida en cuenta y puesta a prueba durante la implementación de este trabajo, en la búsqueda de la opción más eficiente y sencilla. Finalmente se ha optado por esta versión del cálculo en dos partes, ya que era la opción más eficiente y simplificaba el cálculo final en vez de añadir complejidad.

Es importante saber que el fichero generado con este script solo contiene los datos necesarios para cumplir con el formato netCDF y ser "correcto", pero no contiene todos los datos calculados en este trabajo, ya que por eficiencia no era útil. Por ello en el cálculo final han de leerse dos ficheros de datos para obtener todos los elementos necesarios para el fichero netCDF final. "./datos_temporales/dataInd.nc" es un fichero auxiliar sin utilidad por sí mismo ya que recoge una solución parcial, aunque cuenta con todos los requisitos de metadatos para ser interpretado correctamente como fichero netCDF.

Este script tiene un tiempo de ejecución aproximado de 2min 30s.

6.2.4. Índice de calor acumulado. Segundo paso

Por último ha de ejecutarse el script "./indCA.py". Este script comprueba si el índice para el día actual es mayor que cero y consecutivo en la serie temporal y si es así suma el índice anterior a este. Esto funciona ya que en este caso, los datos intermedios calculados no tienen el valor de relleno "-9999" sino valor "0", por lo que no hay necesidad de programar un control de datos sobre cada posibilidad.

El fichero de datos netCDF generado "ICA.nc" contiene todos los datos calculados hasta ahora, además del ICA. Los datos son almacenados en el directorio de datos estáticos del proyecto Django "./calorAcumulado/calorAcumulado/static" o "../static", para poder ser referenciados desde el visor web. Los datos del ICA son almacenados en un formato de datos enteros superior al de las variables con datos simples de temperaturas, ya que al realizar un sumatorio se producía un «overflow» en algunas de las rachas calculadas.

Este script tiene un tiempo de ejecución aproximado de 5min. Con este script se da por generado el fichero de datos netCDF que contiene el índice calculado a partir de temperaturas, es decir, el índice de calor acumulado ICA, que satisface el RF01.

6.2.5. Informe

Una vez calculados todos los datos, el cliente solicitó un resumen en formato legible para los humanos que recogiera los datos de las rachas de calor detectadas. Por ello, se ha creado un script "informe.py" que escribe en el archivo "../static/informeICA.txt" para cada punto geográfico válido, una tupla con la fecha de inicio de cada racha detectada, la duración de esta y el índice de calor acumulado final de dicha racha.

Este fichero es útil para realizar búsquedas específicas en el visor. Es necesario tener en cuenta, que al igual que en los ficheros de datos netCDF, las temperaturas están almacenadas por eficiencia como enteros que representan centésimas de grados, formato que se mantiene en este informe, aunque no en el visor web.

Este script tiene un tiempo de ejecución de 18min. La gran cantidad de escrituras en disco hace que el proceso sea mucho más lento que la escritura en los ficheros netCDF, por eso se mantiene el

formato lo más eficiente posible, evitando el cambio de formato a punto flotante con el sobre coste de caracteres.

6.2.6. Ficheros geoJson

Una vez creado el sitio web y examinado en profundidad el funcionamiento de la librería Leaflet, se observa que aún no existe una API reconocida por Leaflet que conecte los datos en formato netCDF con la librería JavaScript. Por lo que una vez generados los datos es necesario transformarlos de manera programática en algún formato válido para la librería.

Observando los recursos disponibles se observa que uno de los formatos de datos que cuenta con mayor soporte y documentación es el geoJson, además de ser uno de los recomendados por los profesores de este proyecto.

Por ello se desarrollaron dos scripts "./json2D.py" y "./json3D.py" ambos siguen un enfoque similar, el primero guarda en "../static/jsons/ICA2D.geojson" un fichero con todos los datos de 2 dimensiones del fichero "../static/ICA.nc" para cargar en un solo mapa. Este script tiene un tiempo de ejecución de 25s.

El segundo, "./json3D.py [arg0] [arg1]", guarda en el mismo directorio un fichero geoJson por cada día del periodo de tiempo almacenado que contiene las variables de 3 dimensiones con formato "ICA_AÑO_MES_DIA.geojson", estos son los datos que representarán el mapa de calor en el visor web. Cada fichero tarda en ejecutarse entre 20s y 2 minutos aproximadamente por cada mes del período de 4 meses por año, para el período de 35 años (se coge un año extra, 36 años, pero este puede estar incompleto) lo que hace un tiempo de ejecución aproximado de 2 horas.

Este segundo script necesita dos parámetros para su ejecución: "python ./json3D.py 0 1" ejecuta el script "json3D.py" para el año 0 del período, es decir, del primer año del periodo de 35+1 años (0-35) almacenado y el mes 1 del periodo (0-3, junio-septiembre), es decir, julio.

Este manejo de parámetros puede ser confuso, con el objetivo de simplificar al máximo la gestión de este proyecto en el futuro, se ha creado un tercer script python "./exec_json.py" que ejecuta en orden todos los scripts para la generación de ficheros geojson dentro del periodo temporal calculado en el proyecto.

6.2.7. Ejecución de usuario

Como se indica en el apartado anterior, se ha implementado un script que contiene la lógica de ejecución de los scripts que generan los ficheros geojson. Pero este no es el único script que automatiza una ejecución.

Existe un script denominado "./exec.py" que ejecuta en orden y de manera secuencial todos los scripts correspondientes a la lógica de datos descritos anteriormente, incluyendo el script "./exec_json.py". Siendo, pues, el único script python que el futuro gestor deberá ejecutar para generar u actualizar los datos, sin necesidad de introducir parámetros. Solo es necesario actualizar previamente los datos como se indica en "./datos_originales/descargarDatos.py" y con los nuevos ficheros netCDF en la carpeta sustituir en "./resumenDatos.py" los nombres de los archivos de lectura por los nuevos ficheros de datos, probablemente solo sea necesario actualizar la versión del nombre 5.

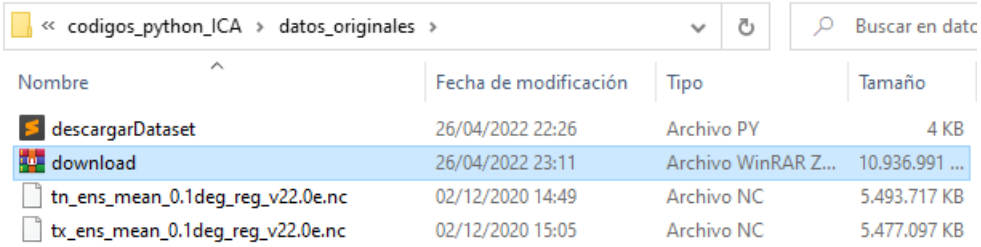

Ilustración 5: Actualización paso 1: descarga

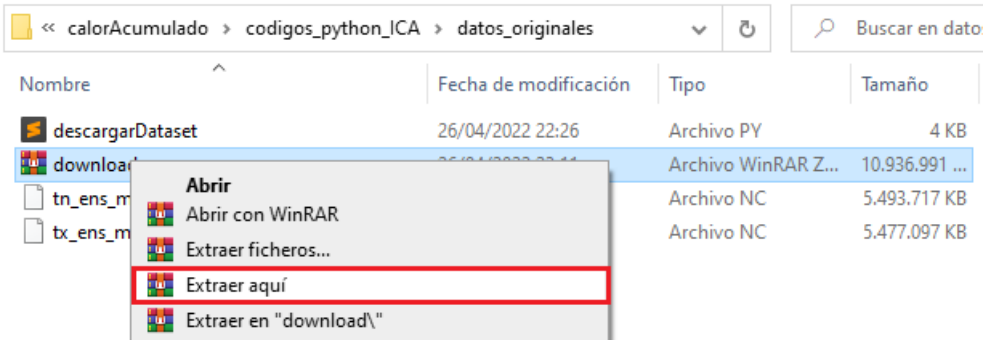

Ilustración 6: Actualización paso 2: descomprimir datos

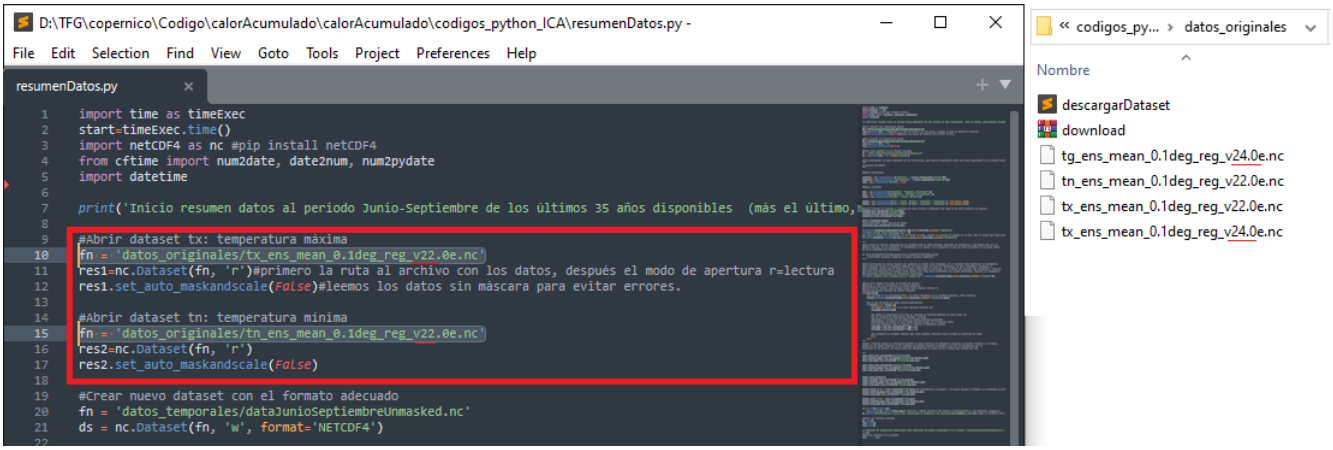

Ilustración 7: Actualización paso 3: actualizar versión del nombre

Este script tiene un tiempo de ejecución total de 2h 40min, a ejecutar 1 vez al año. Los datos se actualizan cada 6 meses, pero solo incluyen el periodo completo de verano 1 vez al año, por lo que no es necesaria más de una actualización anual, ya que solo junio del último año será actualizado la mitad de las veces.

En cada ejecución se generan nuevos ficheros de datos pero no se eliminan todos los anteriores para que, si por algún motivo un fichero se corrompe o pierde, no exista la necesidad de generar todo lo anterior. Esto ha sido muy útil para depurar errores durante las pruebas de ejecución o para comprobar los datos almacenados en cada fase del desarrollo de cálculo de datos. Esto no supone un problema en la mayoría de archivos, ya que en cada ejecución se sobrescribirán los ficheros, pero el contenido de la carpeta "../static/jsons/" podría eliminarse con cada nueva ejecución para evitar que los archivos más antiguos que no se sobrescribirán y quedarán desactualizados sigan siendo accesibles y estén ocupando recursos. El administrador sabrá qué archivos puede eliminar mediante los mensajes por pantalla que le informarán del periodo temporal que se va a generar u observando posterior a la descarga aquellos ficheros que no se hayan actualizado. Puede borrarse el contenido de la carpeta para automatizar la actualización, pero eso dejará fuera de servicio el visor durante unas 2 horas.

Con esto podemos dar el RNF03 por satisfecho. El RNF02 lo cumplimos al generar desde el fichero original una copia completa de los metadatos. Con esta automatización del cálculo de datos cumplimos con el RNF05.

6.3. Proyecto Django

Se ha creado un proyecto Django llamado "./Codigo/calorAcumulado" como gestor del sitio web que hará las veces de interfaz de usuario para los datos. Los scripts python referentes a la gestión del proyecto Django relativo al visor web se encuentran alojados en "./Codigo/calorAcumulado/ca-

lorAcumulado", en esta sección utilizaremos este directorio como directorio raíz "./" para mejorar la legibilidad.

Debido a la falta de familiaridad con este framework, se ha creado un sitio web sencillo cuya necesidad de gestión es mínima. El único problema que puede generarse debido a la migración del proyecto es en el archivo "./settings.py" que contiene las rutas a los directorios "./static" y "./templates", que referencian a la ubicación del directorio estático Django, el cual contiene los datos para que la librería pueda acceder a ellos desde el cliente y al directorio "./templates" que contiene el código html desarrollado que da forma al sitio web. No debería generar problemas, ya que las rutas declaradas son relativas al proyecto, por lo que no es necesario modificarlas al mover el proyecto siempre que se mantenga la estructura organizativa establecida.

Se ha modificado el script "./urls.py" para referenciar las direcciones a las vistas correspondientes. Una vez iniciado el servidor con el comando "python ../manage.py runserver" como se muestra en la ilustración 8, se puede acceder a todos los elementos desde la página principal de la web desarrollada direccionando en el navegador la dirección generada por el servidor Django y la ruta declarada en el fichero, por ejemplo: "127.0.0.1:8000/Home"

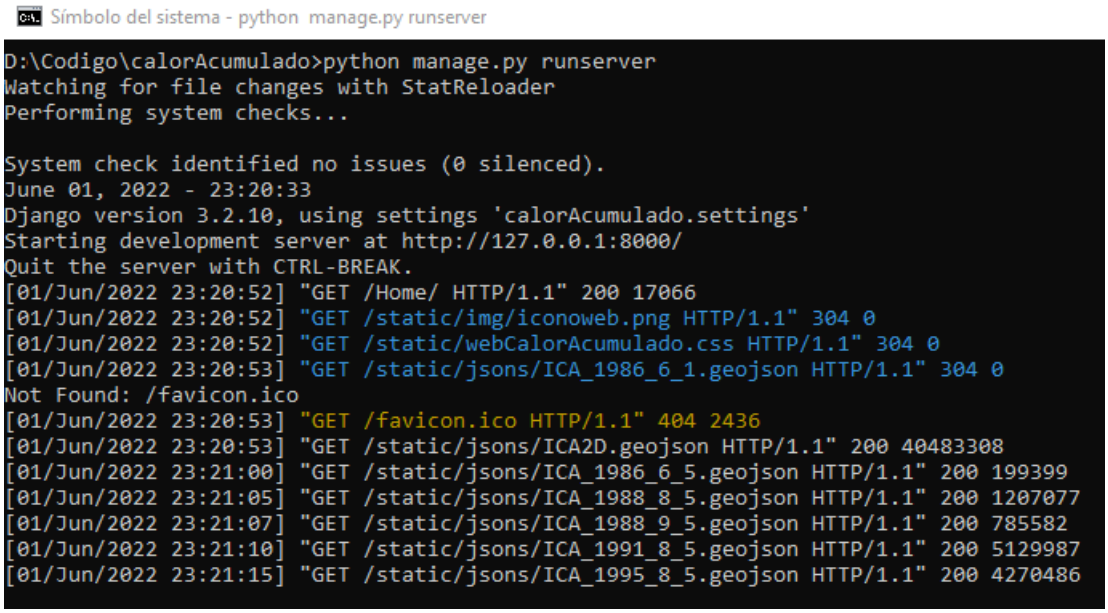

Ilustración 8: Iniciar servidor Django

Por último, se ha modificado el script "./views.py" para devolver las vistas de los html creados al hacer una petición a la dirección url correspondiente, no se pasa ningún contexto, ni parámetro, ya que se trata de un sitio web simplificado al máximo, enfocado en desarrollar la funcionalidad del visor.

6.4. Visor web

Una vez visto cómo se han procesado y almacenado los datos y cómo se ha estructurado el sitio web desde Django, se va a presentar la lógica del sitio web.

Apenas un caso de uso 5.2 definía la funcionalidad del visor web, esta funcionalidad está desarrollada en el archivo "maps.html" en el directorio "./Codigo/CalorAcumulado/CalorAcumulado/templates" que actuará en esta sección como directorio raíz "./". Aunque no es la única página html creada para este sitio web, junto a ella se pueden encontrar los archivos "acerca.html", "legal.html" y "webCalorAcumulado.html".

- "maps.html: contiene dos mapas, uno para visualizar los datos del ICA válidos junto con la temperatura máxima y mínima relativa a ese día y punto geográfico, es decir, se trata de un

mapa en 3 dimensiones: tiempo, latitud y longitud. Y un segundo mapa para visualizar los datos globales: percentiles, mínimos y máximos. Es decir, 2 dimensiones: latitud y longitud. Los mapas se han creado siguiendo el ejemplo publicado por Leaflet en su web y adaptándolo a los datos y necesidades propias. Se puede acceder a esta página desde el navegador con la dirección "/Home/" como se percibe en las ilustraciones 9 y 10

- "acerca.html": hace las veces de carta de presentación personal para poner en contexto el trabajo y pseudocódigo del cálculo de datos, de esta manera, el usuario no necesita descargar el dataset netCDF para entender qué está representando el ICA y cómo. Una vez iniciado el servidor Django se puede acceder a esta página desde el navegador con la dirección generada seguida por "/acerca/" como se percibe en la ilustración 11

- "legal.html": esta vista sirve de apoyo a los metadatos incluidos en el fichero netCDF y es necesaria para cumplir el RNF02, reconociendo la autoría original de los datos e indicando que han sido modificados para este trabajo. Se puede acceder a esta página desde el navegador con la dirección "/legal/" como se percibe en la ilustración 12

- "webCalorAcumulado.html": es una plantilla común a los 3 archivos anteriores que sirve para unificar la estructura y simplificar la escritura de elementos comunes.

Debido a que javaScript no procesa datos que no sean de tipo javaScript, se necesitaba un plugin para utilizar la librería Leaflet con los datos necesarios, pero el formato de datos netCDF no aparecía soportado por ningún plugin reconocido por Leaflet, por lo que se optó por el formato de datos más extendido y recomendado previamente por los profesores encargados, el geoJson, como se indica en 6.2.6.

Para ello se buscó una librería en Python que permitiera convertir fácilmente los datos de formato netCDF a geoJson. Eligiendo como formato de datos geoJson el "Polygon", costoso al generar tantos puntos geográficos, pero el mejor para representar fielmente en el mapa los datos geográficos originales, una cuadrícula con grid 0.1. Al tratarse de ficheros creados exclusivamente para la implementación del visor, se sacrificó la eficiencia en pos de la calidad de imagen.

El fichero de 2 dimensiones geoJson se carga de manera automática en el mapa, mientras que con los ficheros de datos diarios, se carga el primer script de la serie, el más antiguo, y después dispone de un pequeño formulario de datos que actualiza el mapa en función del día introducido. Este formulario definido en "./maps.html" debe actualizar el periodo anual al que se puede hacer referencia con cada actualización, para una mejor experiencia de usuario.

Debido a que geoJson es mucho menos eficiente que netCDF al almacenar estructuras de datos, se decidió cambiar algunos detalles en su escritura Python 6.2.6, como el nombre de las variables, acortándolo todo lo posible pero manteniendo identificables las variables almacenadas en el fichero, siendo legible para un futuro gestor de cara a futuras modificaciones que necesiten interpretar la información almacenada en el fichero. Desde el código javaScript se han definido unos elementos de "información" que muestran en el mapa, en la zona superior derecha, los datos del fichero en un formato más adecuando y legible para el usuario del visor web, grados centígrados en vez de centésimas de grado y cada variable con su nombre identificativo completo.

Los ficheros geoJson de 3 dimensiones solo devuelven para cada día aquellos puntos cuyo ICA es superior a cero, ya que al representar este mapa de temperatura en función de este índice, no tenía sentido el sobre coste de representar datos nulos que no aportarían información visual útil.

Los datos descargables desde los enlaces accesibles desde cada una de las páginas del sitio web son los datos completos en netCDF "../static/ICA.nc" y el resumen de rachas "../static/informeICA.txt". Se distribuye el fichero netCDF por eficiencia y mantener el estándar de datos utilizado para la distribución de datos científicos en red, a diferencia del interpretado por la librería Leaflet para visualizar los datos. Pero no se ha logrado encontrar una forma con la que cumplir el RF03 y generar un marco de descarga personalizable en netCDF, ya que aún no existe una API reconocida para la manipulación de ficheros netCDF desde javaScript, por lo que no se puede programar la lógica de manipulación y escritura del fichero en local.

6.4.1. Estilo del sitio web

Referente al estilo del sitio web, al no recibir ninguna directiva específica, se decidió utilizar colores que contrastaran con el mapa que, para visualizar correctamente las temperaturas, tiene una apariencia brillante en tonos blancos y grises, y se visualizan datos desde el amarillo hasta el granate. Siendo el fondo azul y negro el mejor para generar un contraste con colores complementarios como se describe en 5.4.

Los mapas están alineados a la izquierda para permitir una zona amplia a la derecha en la que poder hacer «scroll», ya sea con el dedo (Leaflet es compatible con dispositivos móviles) o con el ratón sin interferir con los controles propios del mapa de Leaflet.

Los archivos ".html" de la carpeta "./templates" hacen referencia a la hoja de estilos "webCalorAcumulado.css" en el directorio estático del proyecto Django "../static" creada para dar este formato al sitio web, y a la imagen del logotipo de la página web almacenado en el directorio "../static/img/iconoweb.png".

Para modularizar el código se separó el contenido común de la estructura utilizando la herencia con la plantilla "./webCalorAcumulado.html", de la que heredan el resto de ficheros ".htmls". Haciendo que la imagen del sitio web esté unificada y sea flexible a modificaciones.

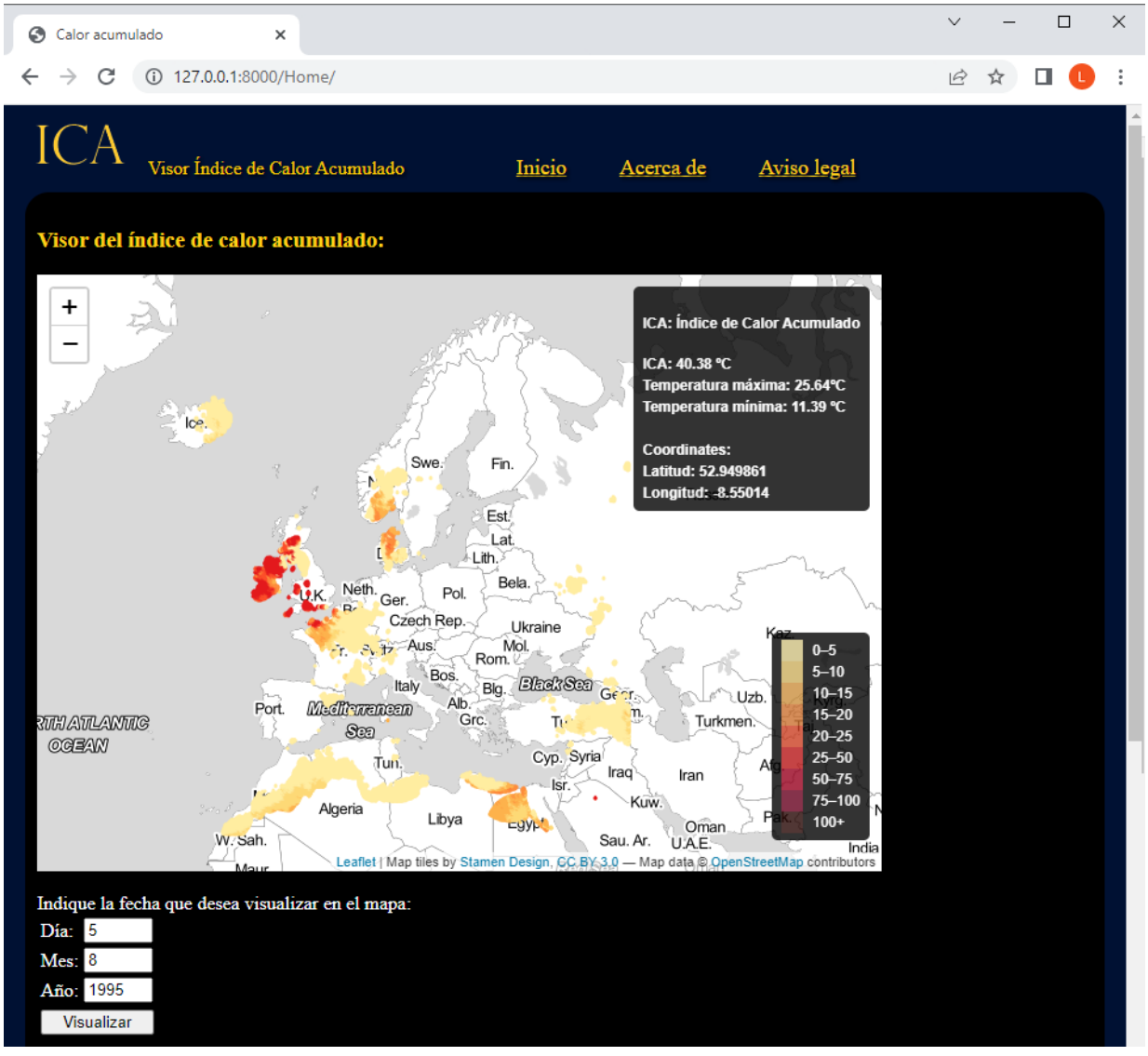

Ilustración 9: Página Home equivalente a maps.html, mapa 3D.

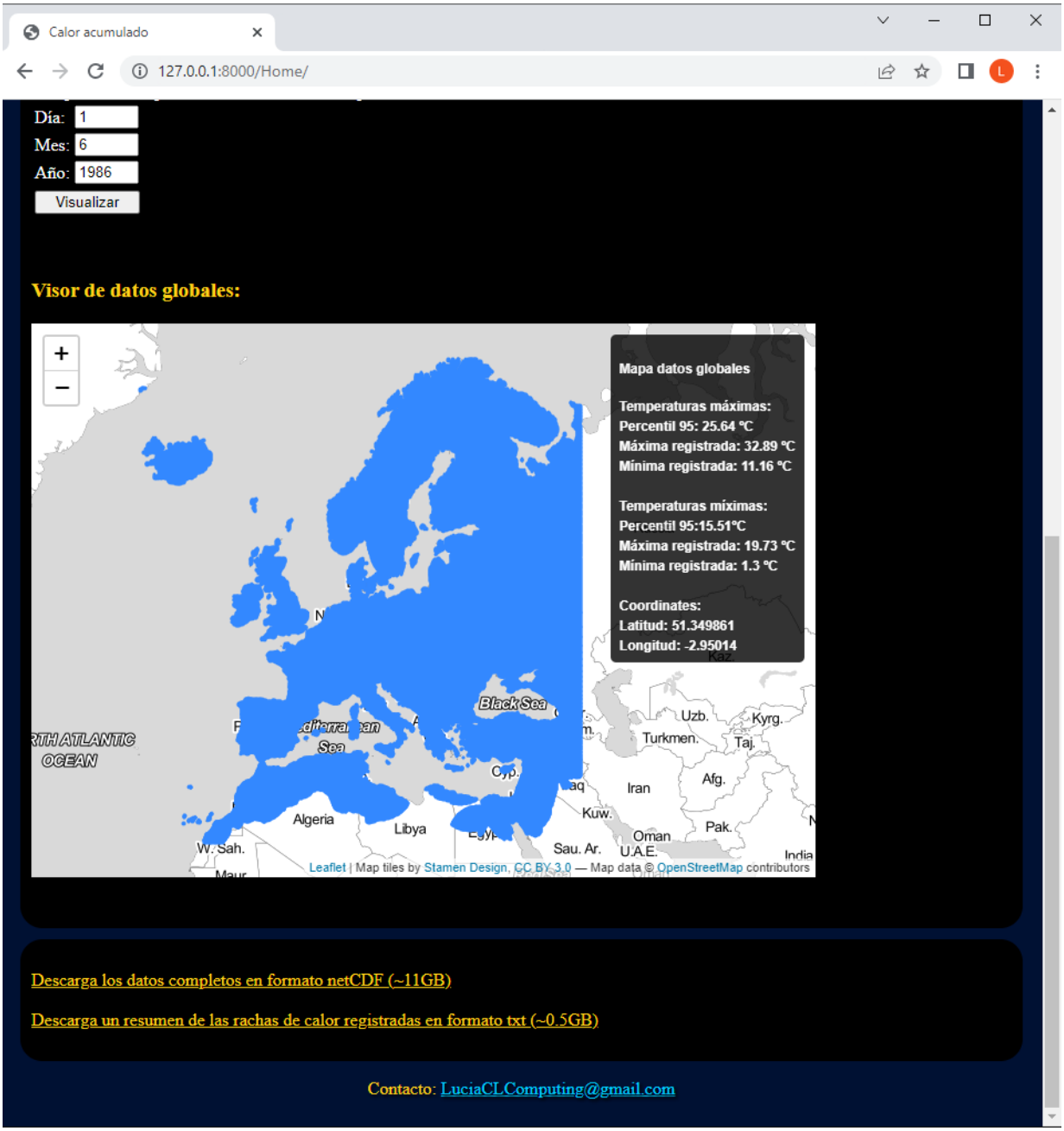

Ilustración 10: Página Home/Inicio equivalente a maps.html, mapa 2D.

6.4.2. Dependencias y resultados

El sitio web tiene dependencias externas con la librería "Leaflet", como es lógico, y con el plugin "Ajax-geojson" necesario para la comunicación entre los datos en formato geoJson y la librería del mapa, por lo que depende de ambas para su correcto funcionamiento como visor.

Es importante comentar que los datos en 2D o globales cargan un archivo muy pesado que provoca que el mapa genere algo de latencia e interfiera en el correcto funcionamiento de la página al renderizar la imagen al desplazar el mapa o cambiar el zoom. Es lo que pasaría si se desearan mapear, por ejemplo, los datos de temperatura diarios, incluyendo los ICAs no válidos (valor igual a 0). Por eso se ha tomado la decisión de hacer que el mapa en función del tiempo, que es el más atractivo, útil e interactivo, sea lo más ágil posible solo cargando los datos por encima del valor cero. Sirviendo el segundo mapa de apoyo para dar contexto a los datos.

La librería genera de manera automática los controles para desplazarnos por el mapa y renderizar

Ilustración 11: Página Acerca de equivalente a acerca.html

la imagen, cumpliendo parcialmente con la vista de casos de uso del usuario, que indica la posibilidad de modificar un marco espacial por parte del usuario en el visor (5.2). En el mapa del ICA, al hacer click sobre un punto geográfico, se producirá un zoom de manera automática sobre ese punto, para poder trabajar más fácilmente sobre un punto en concreto. Además al tratarse de un mapa de color para los datos válidos, este mapa cuenta con una leyenda para interpretar los valores visualmente.

Se ha implementado para ambos mapas la función que genera información en un menú flotante al pasar el cursor sobre uno de los puntos de datos, justificando la utilidad del mapa global y aumentando la utilidad del mapa del ICA que al pasar el ratón nos mostrará los valores de temperatura referentes a ese día además del ICA válido y la información visual.

De esta manera se cumplen los RNF04 y RNF05 parcialmente, ya que en este caso no hay ningún proceso de datos que no esté automatizado, aunque se recomienda actualizar el formulario para la visualización del ICA, actualizando el periodo accesible de datos para mejorar la experiencia de usuario,

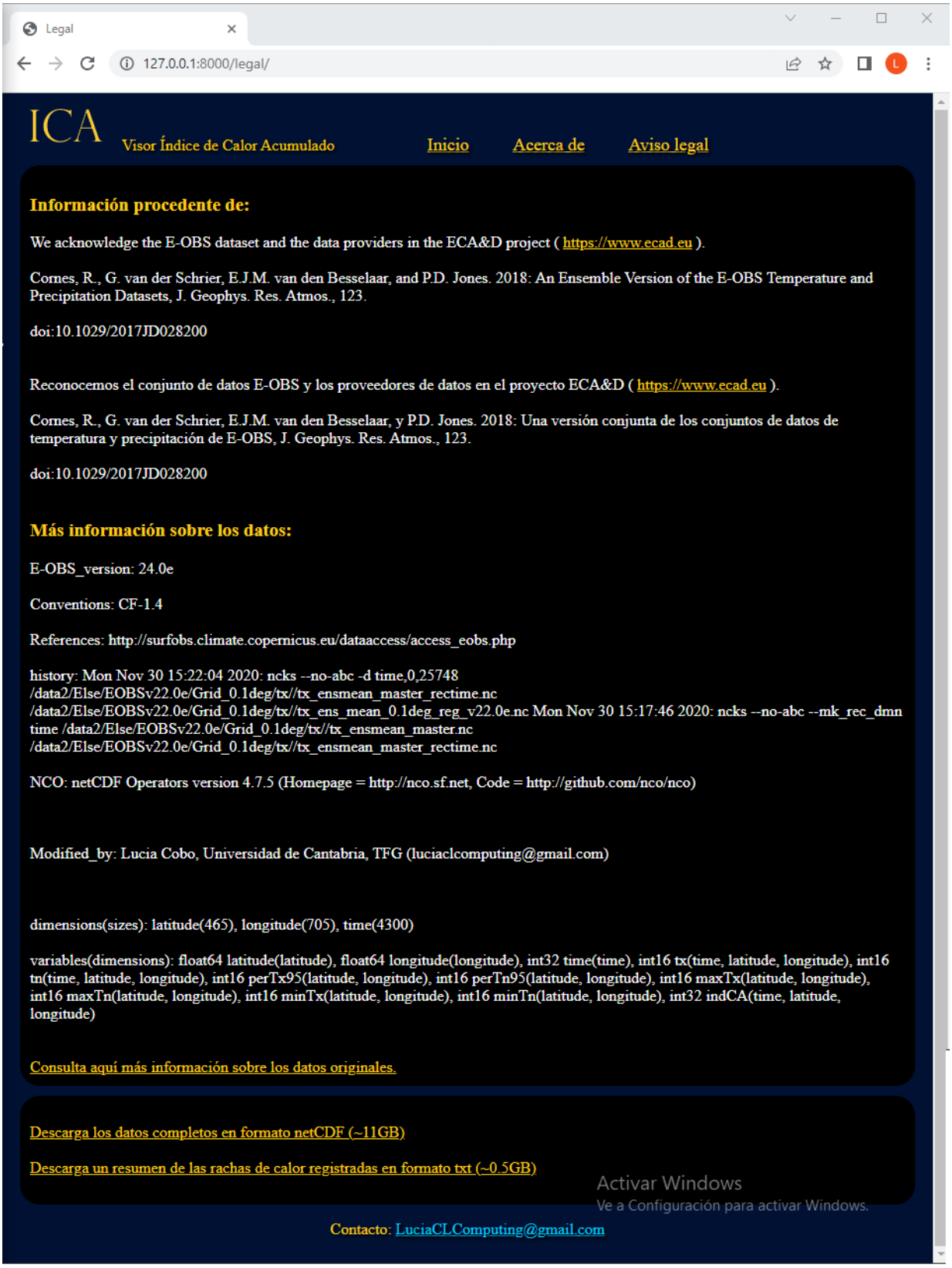

Ilustración 12: Página Aviso legal equivalente a legal.html

es decir, modificando el periodo de año inicial y final del formulario del visor del ICA para que el usuario no intente cargar datos no contemplados. Aunque como se ha comentado, no se ha logrado cumplir con el RF03, por lo que en lugar de una descarga personalizada del fichero netCDF, se ha proporcionado el fichero completo para su descarga junto con el resumen de las rachas de calor solicitado por el cliente.

7 Conclusiones

Aunque aparentemente sencillo, este trabajo ha supuesto más retos de los que pudiera parecer a simple vista. El uso de tantas tecnologías y formatos de datos con los que no estoy familiarizada han aumentado el tiempo del ya considerable proceso de documentación que conlleva este trabajo y aún más, el tiempo invertido en la depuración para generar unos datos de calidad en el formato adecuado.

Desde los webinars online del Proyecto Copernicus y la NASA para el uso del formato netCDF, pasando por los trabajos de referencia consultados para estudiar el ámbito de aplicación de este trabajo, hasta la consulta uno a uno de los índices biometeorológicos registrados en de Freitas and Grigorieva [2015], implicaron un esfuerzo y tiempo que puede pasar desapercibido.

El diseño del índice puede parecer simple, pero se barajaron más ideas que se desecharon en pos de implementar un caso base más eficiente y sencillo conceptualmente, sobre el que cimentar posibles soluciones más complejas a futuro, una vez puesta a prueba su idoneidad. La implementación del cálculo del propio índice no se quedó atrás en cuanto a complejidad. No se trataban de operaciones complejas, pero el costo de mantener los datos en memoria sin contar con un recolector de basura útil limitaba mucho la capacidad y decisiones a tomar sobre los datos. Aunque se recomendaba el uso de la librería Python "gc" para el problema que yo manejaba, esta no funcionaba correctamente sobre Windows. Se pasó de benchmarks de varias semanas e incluso meses a la solución actual de 5 minutos para el segundo paso del cálculo completo del ICA. Todo ello gracias a encontrar un equilibrio entre las costosas operaciones de lectura/escritura de los datos y el uso de la memoria virtual, siendo lo más eficiente posible a la hora de utilizar las variables, los tipos de datos y detectar los límites de ejecución mediante pruebas de benchmark para detectar fugas de tiempo. Dividiendo el código según los límites de la carga en memoria que acababa implicando una paginación que convertía el tiempo de ejecución en exponencial, detectados en ejecuciones previas.

Acostumbrada a la programación orientada a objetos, la implementación de este código secuencial ha supuesto un cambio en mi manera de ver los grandes volúmenes de datos y las estructuras en memoria y disco. La búsqueda extrema de la eficiencia me ha llevado a utilizar cálculos con matrices y a analizar los tamaños de hasta el dato más simple de manera activa.

NetCDF es un formato de datos fantástico y eficiente. Aunque aún no se encuentra una comunidad amplia experta en el tema, por lo que muchas dudas puntuales y simples acaban implicando horas buscando entre la documentación oficial sin encontrar el matiz deseado. Según algunas fuentes consultadas durante la preparación previa para este trabajo, al parecer muchos profesionales están migrando el uso de otros tipos de datos a formato netCDF (sobre todo Estadounidenses), por lo que quizás en los próximos años esto ya no suponga un problema.

Es posible, por tanto, que en un futuro cercano se desarrolle un plugin netCDF para que Leaflet gestione directamente los datos en este formato como hace con geoJson. Leaflet es una herramienta compleja por la gran cantidad de posibilidades de las que dispone y ofrece de manera gratuita, y en muchos casos es difícil depurar los errores, sobre todo para principiantes. Esta librería sí cuenta con una comunidad grande y activa que ayuda a superar esos problemas iniciales, ya sea una dependencia, una hoja de estilo o un pluging al que pertenece la función que necesitas. Adicionalmente, la librería ofrece una documentación amplia y accesible que ayuda a atraer a nuevos usuarios.

8 Futuras líneas de actuación

Este trabajo está ampliamente documentado mediante este informe, el fichero "./Codigo/calorAcumulado/calorAcumulado/README.txt" y los múltiples comentarios en el interior de cada script de este proyecto. De esta manera, el siguiente alumno interesado en retomar el tema no tendrá que invertir tanto tiempo en la documentación, desarrollo y depuración del código, al tener disponible la información necesaria para saber cuáles son los puntos clave y problemáticos de la manipulación de datos netCDF y trabajar con el índice ya calculado.

Se deja pendiente para futuros trabajos de fin de grado en colaboración con alguna institución que pueda proporcionar acceso a datos médicos diarios, una segunda parte de este TFG que analice la utilidad del índice calculado y muestre su relación con algún indicador de la salud como la morbilidad diaria, mediante algún proceso de inferencia para analizar su relación con datos médicos y poner a prueba su validez. Como se barajó en las primeras fases de desarrollo de este proyecto para el último paso de la definición del índice. Debido a la sensibilidad de los datos, no fue posible encontrar un dataset lo suficientemente específico y detallado con el que comparar los datos del ICA calculado. Por lo que se replanteó este trabajo como una herramienta de consulta que sentara las bases para futuros desarrollos en torno al ICA diseñado.

Otra de las futuras líneas de actuación sería diseñar un sistema de alerta temprana basado en datos de pronósticos meteorológicos cuya implementación permitiría reducir el impacto de las olas de calor en la mortalidad y la morbilidad y facilitaría la gestión de los recursos sanitarios de nuestra sociedad con el beneficio múltiple que ello conlleva de cara al bienestar de las personas. El código proporcionado es fácilmente adaptable a datos de pronóstico y a la implementación de nuevos modelos, por ello la propuesta metodológica es extrapolable a otros estudios que investiguen la relación entre los procesos atmosféricos y la salud de las personas, mucho más en el contexto actual de cambio climático.

Referencias

C. Brimicombe, C. Di Napoli, T. Quintino, F. Pappenberger, R. Cornforth, and H. L. Cloke. Thermofeel: A python thermal comfort indices library. *SoftwareX*, 18:101005, 2022.

C. R. de Freitas and E. A. Grigorieva. A comprehensive catalogue and classification of human thermal climate indices. *International Journal of Biometeorology*, 59(1):109–120, 2015.

A. Santurtún, R. Almendra, P. Fdez-Arroyabe, A. Sanchez-Lorenzo, D. Royé, M. T. Zarrabeitia, and P. Santana. Predictive value of three thermal comfort indices in low temperatures on cardiovascular morbidity in the iberian peninsula. *Science of The Total Environment*, 729:138969, 2020a.

A. Santurtún, R. Almendra, G. L. Silva, P. Fdez-Arroyabe, M. Santurtún, and P. Santana. Suicide and apparent temperature in the two capitals cities in the iberian peninsula. *Social Science & Medicine*, 265:113411, 2020b.