UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Geodésica, Cartográfica y Topográfica

Herramienta para la integración de la información de marea obtenida de Puertos del Estado para la mejora de los procesos de extracción de la línea de costa a partir de imágenes satelitales

Trabajo Fin de Máster

Máster Universitario en Ingeniería Geomática y Geoinformación

AUTOR/A: Soriano Dolz, Noelia

Tutor/a: Palomar Vázquez, Jesús Manuel

CURSO ACADÉMICO: 2024/2025

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA GEODÉSICA CARTOGRÁFICA Y TOPOGRÁFICA

Herramienta para la integración de la información de marea obtenida de Puertos del Estado para la mejora de los procesos de extracción de la línea de costa a partir de imágenes satelitales

Trabajo Final de Máster

Máster en Ingeniería Geomática y Geoinformación

Autora: Noelia Soriano Dolz
Tutor: Jesús Manuel Palomar Vázquez

Curso académico: 2024 - 2025

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que han contribuido de manera significativa en la realización de este proyecto.

En primer lugar, quiero agradecer a mi familia y amigos, por su amor, paciencia y por creer en mí en todo momento, apoyándome y motivándome para conseguir todo lo que me he propuesto.

En especial a Mario, que has sido mi apoyo incondicional, mi inspiración y mi fuerza en los momentos difíciles. Gracias por estar siempre a mi lado, por creer en mí cuando yo dudaba y por compartir cada paso de este camino. Este logro también es tuyo.

A Jesús por brindarme la oportunidad y su apoyo en la realización de este Trabajo Final de Máster.

Compromiso

"El presente documento ha sido realizado completamente por el firmante; no ha sido entregado como otro trabajo académico previo y todo el material tomado de otras fuentes ha sido convenientemente entrecomillado y citado su origen en el texto, así como referenciado en la bibliografía"

Valencia, 10/07/2025 Noelia Soriano Dolz

Resumen

Este Trabajo Final de Máster se centra en el desarrollo de una herramienta de apoyo orientada a mejorar la precisión en la detección de líneas de costa obtenidas a partir de imágenes satelitales que el grupo de investigación de Cartografía GeoAmbiental y Teledetección (CGAT) ha desarrollado, incorporando información de la Red de Mareógrafos de Puertos del Estado. Una línea de costa es la superficie dinámica que define el límite entre el mar y la tierra, cuya posición presenta variaciones temporales y espaciales que están ligadas, entre otros factores, al estado de la marea.

Para ello, se recopilan datos con una resolución temporal de un minuto provenientes de los mareógrafos disponibles distribuidos por el litoral español. Estos datos se relacionan espacial y temporalmente con imágenes satelitales de las series Landsat 8, 9 y Sentinel-2, previamente armonizadas. La integración de los datos permite ajustar la posición de la línea en función de las series temporales del nivel del mar en el momento de adquisición de la imagen.

La metodología se desarrolla en un entorno de programación en Python, utilizando bibliotecas específicas para la descarga, tratamiento y análisis de los datos geoespaciales y oceanográficos.

Palabras clave: Mareógrafos, línea de costa, imágenes satelitales, Puertos del Estado, Python.

Resum

Este Treball Final de Màster se centra en el desenrotllament d'una ferramenta de suport orientada a millorar la precisió en la detecció de línies de costa obtingudes a partir d'imatges satel·litàries que el grup d'investigació de Cartografia Geoambiental i Teledetecció (CGAT) ha desenrotllat, incorporant informació de la Xarxa de Mareògrafs de Ports de l'Estat. Una línia de costa és la superfície dinàmica que defineix el límit entre la mar i la terra, la posició de la qual presenta variacions temporals i espacials que estan lligades, entre altres factors, a l'estat de la marea.

Per a això, es recopilen dades amb una resolució temporal d'un minut provinents dels mareògrafs disponibles distribuïts pel litoral espanyol. Estes dades es relacionen espacial i temporalment amb imatges satel·litàries de les sèries Landsat 8, 9 i Sentinel-2, prèviament harmonitzades. La integració de les dades permet ajustar la posició de la línia en funció de les sèries temporals del nivell de la mar en el moment d'adquisició de la imatge.

La metodologia es desenrotlla en un entorn de programació en Python, utilitzant biblioteques específiques per a la descàrrega, tractament i anàlisi de les dades geoespacials i oceanogràfiques.

Paraules clau: Mareògrafs, línia de costa, imatges satel·litàries, Ports de l'Estat, Python.

Abstract

This master's thesis focuses on the development of a support tool aimed at improving the accuracy in the detection of coastlines obtained from satellite images that the Geo-Environmental Cartography and Remote Sensing (CGAT) research group has developed, incorporating information from Puertos del Estado Tide Gauges Network. A coastline is the dynamic surface that defines the boundary between the sea and the land, whose position presents temporal and spatial variations that are linked, among other factors, to the state of the tide.

For this purpose, data with a temporal resolution of one minute are collected from the available tide gauges distributed along the Spanish coastline. These data are spatially and temporally related to satellite images from the Landsat 8, 9 and Sentinel-2 series, previously harmonised. The integration of these data allows for adjusting the coastline position based on the sea level time series at the moment the image was acquired.

The methodology is developed in a Python programming environment, using specific libraries for the download, processing and analysis of geospatial and oceanographic data.

Keywords: Tide gauges, coastline, satellite images, *Puertos del Estado*, Python.

Índice de figuras

Índice de tablas

Índice

Glosario

API → Interfaz de Programación de Aplicaciones.

BOT → *Bottom Of Atmosphere*.

CGAT → Grupo de Cartografía GeoAmbiental y Teledetección.

ESA → Agencia Espacial Europea.

GEE → *Google Earth Engine.*

ODS → Objetivos de Desarrollo Sostenibles.

REDMAR → Red de Mareógrafos de Puertos del Estado.

TOP → *Top Of Atmosphere*.

1. Introducción

El presente Trabajo Final de Máster se desarrolla en colaboración con el **Grupo de Cartografía GeoAmbiental y Teledetección (CGAT)** de la Universidad Politécnica de Valencia (UPV). Se trata de un grupo de investigación multidisciplinar del Departamento de Ingeniería Cartográfica, Geodesia y Fotogrametría, el cual se centra en el desarrollo de herramientas tecnológicas de geoinformación basadas en teledetección, Sistemas de Información Geográfica (SIG) y procesamiento LiDAR. (Grupo de Cartografía Geoamtiental y Teledetección (CGAT))

ILUSTRACIÓN 1: LOGO GRUPO DE CARTOGRAFÍA GEOAMBIENTAL Y TELEDETECCIÓN (CGAT). FUENTE: (GRUPO DE CARTOGRAFÍA GEOAMTIENTAL Y TELEDETECCIÓN (CGAT))

1.1. Antecedentes

La **extracción precisa de la línea de costa** es fundamental para la gestión y monitorización de los entornos litorales, especialmente ante el avance del cambio climático y la presión antrópica de las zonas costeras. La línea de costa es un indicador clave para el análisis de la dinámica litoral, la erosión, la planificación territorial y la protección de infraestructuras y ecosistemas. (Ojeda et al., 2013)

En las últimas décadas, el uso de **imágenes satelitales** ha revolucionado la capacidad de obtener información espacial y temporal sobre la evolución de la línea de costa, permitiendo análisis históricos y actuales a gran escala y bajo coste. Sin embargo, la extracción automática de la línea de costa a partir de imágenes satelitales plantea desafíos técnicos y científicos (Pardo-Pascual et al., 2008):

- **Variabilidad temporal y espacial** → la ubicación de la línea de costa es extremadamente fluctuante, afectada por factores como el oleaje, la morfología de la playa, la presencia de vegetación, el nivel del mar y las mareas.
- **Errores de interpretación** → la resolución espacial de los sensores, la calidad de las imágenes y las condiciones atmosféricas pueden generar errores significativos en la determinación de la línea de costa.

En particular, en el litoral norte de España, las mareas presentan diferencias de nivel considerables, con variaciones de varios metros entre pleamar y bajamar. Esta situación provoca que, durante la bajamar, amplias zonas de arena permanecen húmedas. Las técnicas empleadas tienden a identificar estas superficies húmedas como cuerpos de agua, lo que produce una **ubicación imprecisa de la línea de costa**, desplazándola tierra adentro respecto a su posición real.

A continuación, se presentan algunos estudios centrados en el análisis de la dinámica y regresión de la línea de costa en España, haciendo uso de tecnologías como la teledetección y los modelos digitales del terreno. Estos trabajos abordan la erosión costera, el retroceso litoral y la necesidad de monitorización continua y modelos predictivos ante los efectos del cambio climático y el desarrollo humano:

1. **Impactos en la costa española por efecto del cambio climático** (Ministerio para la transición ecológica y el reto demográfico, 2004). Este estudio analiza cómo el cambio climático está acelerando la erosión y el retroceso del litoral debido al aumento del nivel del mar y a la modificación de las dinámicas marinas. Estas transformaciones incrementan el riesgo de inundaciones y daños en infraestructuras, agravadas por la presión urbanística en las zonas costeras. También se documenta la pérdida de ecosistemas clave como playas y humedales, y el desplazamiento de especies marinas.

2. ***A remote monitoring approach for coastal engineering projects*** (Cabezas-Rabadán et al., 2025)**.** Se evidencia la alta variabilidad espacial y temporal de la línea de costa en España, con retrocesos acusados en áreas concretas. Estos cambios están vinculados a la alteración de los procesos sedimentarios, la subida del nivel del mar y la frecuencia creciente de eventos extremos. El estudio subraya la necesidad de modelos predictivos para anticipar los impactos y establecer estrategias de mitigación eficaces.

3. ***Assessment of satellite-derived shorelines automatically extracted from Sentinel-2 imagery using SAET*** (Pardo-Pascual et al., 2023). Este trabajo examina la erosión acelerada en playas del litoral español, resaltando la influencia conjunta de procesos naturales y la actividad humana en la regresión costera. Se enfatiza la importancia de la monitorización continua mediante imágenes satelitales para comprender la evolución morfológica del litoral y desarrollar medidas de protección adecuadas.

El seguimiento de los cambios en la línea de costa ha dado lugar al desarrollo de diversas **herramientas** diseñadas específicamente para su **detección y análisis a partir de imágenes satelitales**. Estas soluciones combinan técnicas de procesamiento digital de imágenes, algoritmos de detección de bordes y criterios geomorfológicos para identificar con precisión los cambios en el litoral. Entre las principales herramientas y algoritmos empleados en este ámbito se encuentra:

1. ***SHOReline EXtraction software* (SHOREX)** (Palomar-Vázquez et al., 2018)**.** Herramienta para determinar la posición de la línea de costa de forma automática a partir de imágenes satelitales (Sentinel-2 o Landsat 8, 7 y 5). Integra funciones como la descarga de imágenes, filtrado de nubes y georreferenciación subpíxel.

2. **Software para el análisis y extracción de la línea de costa a partir de imágenes satelitales (SAET)** (Palomar-Vázquez et al., 2023). Aplicación *open-source* que automatiza el proceso de detección y análisis de líneas de costa mediante imágenes satelitales (Sentinel-2 y Landsat 8/9) con precisión subpíxel. Emplea índices espectrales y de análisis de gradiente para identificar con precisión la línea de costa, y ha demostrado una alta fiabilidad en distintos entornos costeros. Es especialmente útil para el monitoreo rutinario, el estudio de eventos extremos y la gestión de riesgos en zonas litorales.

3. ***CoastSat: A Google Earth Engine-enabled Python toolkit to extract shorelines from publicly available satellite imagery*** (Kilian-Vos et al., 2019). Herramienta de código abierto diseñada para extraer series temporales de posiciones de línea de costa a escala global. Gracias a su integración con Google Earth Engine (GEE), es capaz de descargar y preprocesar automáticamente las imágenes, determinando la línea de costa. Para ello, realiza una combinación de técnicas de clasificación, identificando los píxeles correspondientes a agua, arena, espuma u otros, y segmentación, utilizando índices espectrales como el MNDWI (Índice Diferencial Normalizado de Agua de Humedad).

4. ***Coastal Analyst System from Space Imagery Engine* (CASSIE)** (A.H.F. KLEIN et al., 2023). Aplicación web de código abierto para el mapeo y análisis automático de líneas de costa utilizando imágenes satelitales multiespectrales. Su procesamiento se aloja en la nube, aprovechando la infraestructura de Google Earth Engine (GEE). Emplea el umbral Otsu sobre el Índice de Agua de Diferencia Normalizada (NDWI) para la detección inicial, y suaviza los resultados con un filtro gaussiano

unidimensional con el fin de eliminar el efecto escalera generado por los píxeles.

En conjunto, estos avances tecnológicos permiten abordar con mayor precisión los desafíos asociados a la detección y análisis de la línea de costa. No obstante, aún existen dificultades significativas en la gestión litoral, tales como la variabilidad espacial y temporal de la línea, la influencia de condiciones ambientales variables, y la confusión entre áreas húmedas y cuerpos de agua en imágenes satelitales. Estos factores, junto con la presión humana y el aumento de fenómenos extremos relacionados con el cambio climático, dificultan la adquisición de datos fiables y consistentes a lo largo del tiempo. En este contexto, una seguimiento continuo y preciso mediante herramientas especializadas no solo mejora la comprensión de la dinámica costera, sino que resulta **esencial para prever impactos, evaluar riesgos y elaborar estrategias** eficaces de adaptación y protección del entorno litoral.

1.2. Justificación

La gestión eficaz del litoral, especialmente en regiones meso y macro mareales como las costas cantábricas y atlánticas, requiere una delimitación precisa y actualizada de la línea de costa. En un contexto marcado por el cambio climático y sus efectos sobre el medio costero, disponer de métodos automáticos y robustos para extraer esta línea a partir de imágenes satelitales es una prioridad. Sin embargo, estos métodos aún presentan limitaciones importantes, lo que justifica la necesidad de investigar enfoques más adaptativos y específicos que mejoren la precisión y aplicabilidad de los resultados en este tipo de entornos.

Esta problemática evidencia la necesidad de incorporar información adicional, como los datos mareográficos, para corregir dichas imprecisiones. En este contexto, el presente trabajo cobra relevancia al desarrollar una herramienta que integra imágenes satelitales y datos de mareas.

Este tipo de soluciones tecnológicas se alinean directamente con los Objetivos de Desarrollo Sostenible (ODS) propuestos por la Agenda 2030, los cuales se muestran a continuación:

ILUSTRACIÓN 2: OBJETIVOS DE DESARROLLO SOSTENIBLE (ODS). FUENTE: (NACIONES UNIDAS)

Las iniciativas relacionadas con los ODS fomentan la implementación de tecnologías innovadoras para la gestión eficaz y sostenible del medio ambiente. En este contexto, la creación de herramientas basadas en teledetección y datos oceánicos contribuye directamente al objetivo "**Industria, innovación e infraestructura**" (ODS 9), fomentando la investigación y el uso de soluciones tecnológicas avanzadas para el análisis del medio costero.

Por otro lado, la mejora en los algoritmos de detección y ajuste de líneas de costa es esencial para una correcta planificación territorial, especialmente en zonas costeras vulnerables. Esto influye en la creación de "**Ciudades y comunidades sostenibles"** (ODS11), facilitando la gestión litoral y reduciendo la vulnerabilidad de las poblaciones al riesgo de inundaciones y otros impactos derivados del cambio climático.

Asimismo, al incluir el análisis de las variaciones del nivel del mar y su impacto en la dinámica costera, contribuye al objetivo "**Acción por el clima**" (ODS 13). Finalmente, el enfoque en la precisión del límite entre mar y tierra apoya al objetivo "**Vida submarina**" (ODS 14), ya que permite mejorar la delimitación y conservación de ecosistemas marino-costeros.

ILUSTRACIÓN 3: ODS CON UN MAYOR GRADO DE RELACIÓN. FUENTE: (NACIONES UNIDAS)

1.3. Localización

El trabajo se llevará a cabo en España, país situado al suroeste de Europa, en la península ibérica, que comparte con Portugal. Además, cuenta con territorios fuera de la península, como las Islas Baleares en el mar Mediterráneo, las Islas Canarias en el océano Atlántico y las ciudades autónomas de Ceuta y Melilla, ubicadas en el norte de África.

A lo largo de toda la costa del territorio español se dispone de una red de mareógrafos gestionada por Puertos del Estado. Esta red proporciona información en tiempo real, fundamental para el análisis y monitoreo de la dinámica costera.

ILUSTRACIÓN 4: *DISTRIBUCIÓN DE MAREÓGRAFOS REDMAR. FUENTE: ELABORACIÓN PROPIA.*

2. Objetivos

2.1. Objetivo general

El presente proyecto tiene como objetivo general la descarga de imágenes satelitales de la serie Sentinel-2 correspondientes a un área determinada, la extracción de sus metadatos asociados y la determinación del estado de la marea (pleamar o bajamar) en el momento de adquisición. Esta información se integrará con el fin de mejorar la precisión de los algoritmos de detección automática de la línea de costa desarrollados en el CGAT.

2.2. Objetivos específicos

Los objetivos específicos son los siguientes:

OE1 → Creación de una interfaz que simplifique la interacción con la aplicación.

OE2 → Desarrollar un visor geoespacial que permita la selección visual del área de interés para el recorte de imágenes.

OE3 → Establecer conexión con la API de Google Earth Engine (GEE) para la consulta y descarga de datos satelitales.

OE4 → Implementar un sistema automatizado de descarga de datos de los mareógrafos gestionados por Puertos del Estado.

OE5 → Obtención de informes con el estado de marea resultante en diferentes instantes determinados.

3. Datos

Este apartado recoge y detalla los datos empleados para llevar a cabo el proceso, destacando su procedencia y que información aporta. Dado que una parte del trabajo involucra el análisis de datos mareográficos, resulta fundamental, en primer lugar, comprender qué es la **marea** y cuáles son sus conceptos claves.

Las mareas son **variaciones periódicas del nivel del mar** que se manifiestan como movimientos de ascenso y descenso de las aguas marinas. Este fenómeno ocurre en todos los océanos y mares del mundo y puede observarse claramente en las zonas costeras. Aunque comúnmente se asocian solo al agua, las mareas también afectan a la Tierra sólida, que puede llegar a elevarse hasta 30 centímetros, efecto que suele pasar desapercibido. (Martín, 2009)

Su comportamiento está determinado por diversos factores. Uno de los principales es la **posición relativa de la Luna y, en menor medida, el Sol con respecto a la Tierra**, ya que la atracción gravitatoria de ambos astros influye directamente en el movimiento del agua. La Luna es el principal agente causante de las mareas, ya que, a pesar de ser mucho más pequeña que el Sol, se encuentra mucho más cerca de la Tierra, lo que hace que su influencia gravitatoria sea entre dos y tres veces mayor.

Además, la **forma** y la **profundidad** de los océanos y mares condicionan cómo se propagan las mareas en diferentes regiones del planeta. Por otra parte,

también interviene la **configuración geográfica de la costa** y de la **plataforma continental**, que puede amplificar o atenuar el efecto mareal. Por último, aunque las **condiciones meteorológicas** no generan mareas por sí mismas, sí pueden modificarlas a nivel local, alterando su altura o su momento de llegada. (Marqués)

En este sentido, es importante diferenciar entre dos tipos de marea: la **marea astronómica**, causada exclusivamente por la atracción gravitatoria de la Luna y el Sol, y la **marea meteorológica**, que es un efecto adicional determinado por factores atmosféricos como la presión y el viento. Especialmente, esta última puede provocar sobreelevaciones del nivel del mar y afectar a la dinámica costera a corto plazo. Como consecuencia, las olas pueden avanzar más tierra adentro, fenómeno conocido como *run-up* (ilustración 5).

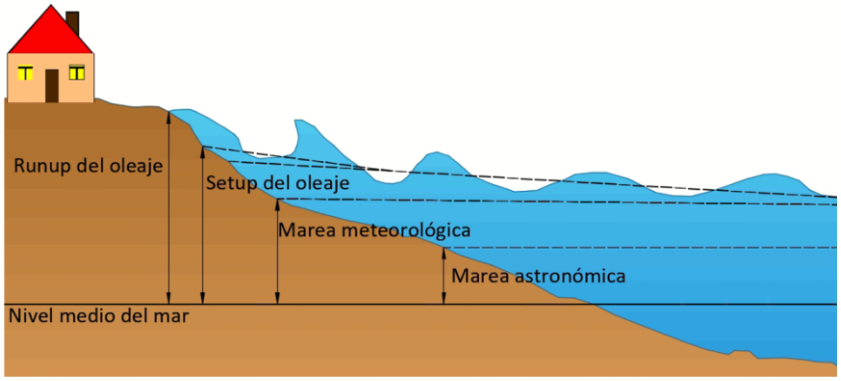

ILUSTRACIÓN 5: *VARIACIONES DEL NIVEL DEL MAR A CORTO PLAZO. FUENTE: (MONTOYA-RODRÍGUEZ ET AL., 2022)*

Tal como se muestra en la ilustración 6, la marea astronómica presenta un comportamiento periódico y predecible, mientras que la meteorológica presenta variaciones más irregulares y difíciles de anticipar.

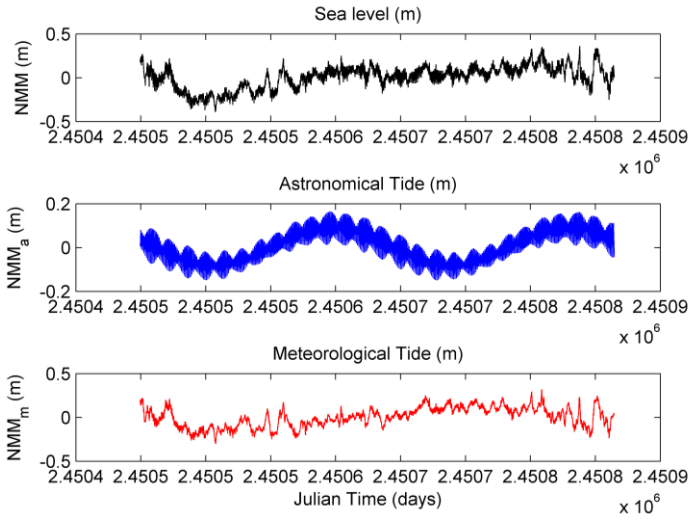

ILUSTRACIÓN 6: *REPRESENTACIÓN GRÁFICA DE LAS SERIES TEMPORALES DE VARIACIÓN DEL NIVEL DEL MAR, MAREA ASTRONÓMICA Y METEOROLÓGICA. FUENTE: (ROJAS, 2013)*

Este comportamiento combinado de las mareas cobra especial relevancia en el contexto del análisis de líneas de costa a partir de imágenes satelitales. Estas líneas representan una posición instantánea del límite mar-tierra, correspondiente al momento exacto de la toma de la imagen. En zonas con un rango mareal elevado (meso o macro mareales), esta línea instantánea puede diferir significativamente de la posición media que correspondería a un nivel de marea de referencia.

Por ello, conocer el nivel de marea en el momento de adquisición satelital, junto con la pendiente de la playa, permite aplicar correcciones que ajusten la línea observada a una posición de referencia más representativa, como el nivel medio del mar. Este paso es esencial para poder comparar múltiples líneas de costa obtenidas en diferentes fechas, asegurando que todas se encuentran referidas al mismo nivel base. Solo entonces es posible analizar con precisión la evolución del litoral y detectar si una zona está experimentando procesos de erosión o acreción.

Para comprender mejor el comportamiento de las mareas y su impacto en el entorno costero, es necesario conocer algunos **conceptos clave** que permiten interpretar adecuadamente los datos mareográficos:

Pleamar → Es el punto más alto que alcanza el nivel del mar durante el ciclo. También se conoce como marea alta y se produce cuando el agua del mar ha subido al máximo antes de comenzar a descender.

Bajamar → Corresponde al punto más bajo del nivel del mar en el ciclo mareal. Se trata de la marea baja, cuando el agua ha descendido al mínimo antes de comenzar a subir nuevamente.

Amplitud → Se refiere a la diferencia de altura entre una pleamar y la bajamar siguiente. Es un indicador de la intensidad de la variación del nivel del mar en un ciclo completo de marea.

Marea viva → Es un tipo de marea que presenta una mayor amplitud. Se produce cuando la Luna y el Sol se encuentran alineados con la Tierra, lo que refuerza la atracción gravitatoria conjunta sobre los océanos.

Marea muerta → Es una marea de menor amplitud que ocurre cuando el Sol y la Luna forman un ángulo recto respecto a la Tierra (durante los cuartos lunares). En este caso, las fuerzas gravitatorias se contrarrestan parcialmente, lo que reduce el efecto sobre el nivel del mar.

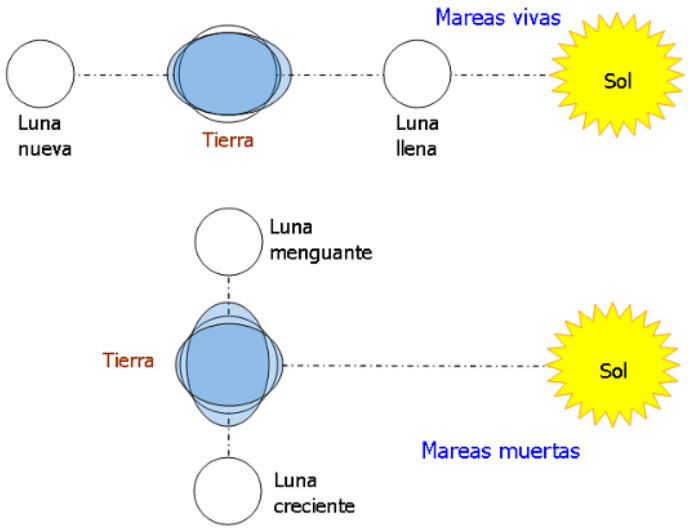

ILUSTRACIÓN 7: MAREAS VIVAS Y MUERTAS. FUENTE: (FERNÁNDEZ)

3.1. Red de mareógrafos de Puertos del Estado (REDMAR)

Puertos del Estado es una entidad pública adscrita al **Ministerio de Transportes y Movilidad Sostenible de España**, encargada de coordinar el conjunto del sistema portuario de titularidad estatal. Este sistema está compuesto por 28 Autoridades Portuarias, responsables de la gestión de 46 puertos catalogados de interés general debido a su relevancia estratégica, tanto por las actividades económicas que albergan como por su influencia en el conjunto del territorio nacional. (Puertos del Estado)

Puertos del Estado

ILUSTRACIÓN 8: LOGO PUERTOS DEL ESTADO. FUENTE: (PUERTOS DEL ESTADO)

Dispone de una infraestructura de observación basada en sensores que monitorizan el estado del mar, e integra conjuntos de datos climáticos que permiten caracterizar las condiciones marítimas actuales, así como proyectar los posibles escenarios de cambio. Entre ellos, se encuentra la **Red de Mareógrafos (REDMAR)**, cuyo objetivo principal es registrar de manera constante el nivel del mar en los principales puertos.

En la actualidad, cuenta con **42 estaciones mareográficas** activas distribuidas a lo largo de la costa española, que abarca el litoral peninsular, Baleares y Canarias, las cuales emplean tecnología radar de barrido de frecuencias para medir la agitación y el oleaje en el interior de los puertos (ilustración 9). Estos mareógrafos miden variables clave como el **nivel del mar, altura de ola, periodo de oleaje y presión atmosférica**. Además, algunas

estaciones incorporan sensores meteorológicos para obtener información de viento y presión. (Puertos del Estado, 2024)

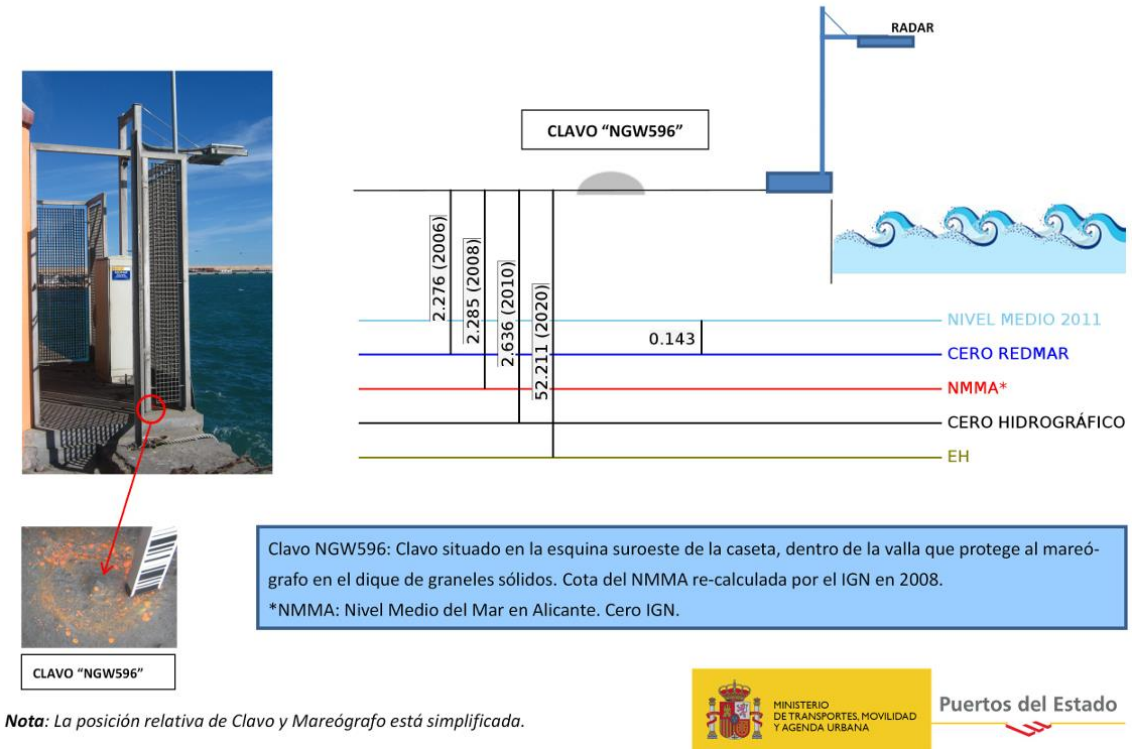

ILUSTRACIÓN 9: *ESQUEMA DATUM MAREÓGRAFO REDMAR "VALENCIA 3". FUENTE: (PUERTOS DEL ESTADO)*

A continuación, la Tabla 1 muestra la información básica de los mareógrafos operativos. En total, se registran **44 estaciones**, donde 2 de ellas pertenecen a un mismo emplazamiento. Es el caso de "Almería" y "Almería 2", que hacen referencia a una misma estación, así como "Langosteira" y "Langosteira 2".

Código BD	Nombre mareógrafo	Latitud	Longitud	Inicio Medida Nivel del Mar	Inicio medida agitación	Inicio Medida meteorología	Muestreo
3108	Gijón 2	43,56	-5,69	29/06/1995	29/02/2008	24/06/2015	1 minuto
3109	Santander 2	43,466	-3,79	30/06/1995	01/03/2008	-	1 minuto
3114	Bilbao 3	43,356	-3,04	01/07/1995	02/03/2008	-	1 minuto
3115	Pasaia	43,32	-1,92	02/07/1995	03/03/2008	25/09/2019	1 minuto
3210	San Cibrao	43,71	-7,46	03/07/1995	04/03/2008	10/09/2020	1 minuto
3212	Ferrol 4	43,47	-8,32	04/07/1995	05/03/2008	-	1 minuto
3214	Langosteira	43,35	-8,53	05/07/1995	06/03/2008	-	1 minuto
3214_2	Langosteira 2	43,35	-8,5	06/07/1995	07/03/2008	-	1 minuto
3215	Ferrol 1	43,46	-8,36	07/07/1995	08/03/2008	02/10/2015	1 minuto
3217	Ferrol 2	43,48	-8,25	08/07/1995	09/03/2008	-	1 minuto
3219	A Coruña 2	43,36	-8,39	09/07/1995	10/03/2008	-	1 minuto
3220	Villagarcia 2	42,6	-8,77	10/07/1995	11/03/2008	09/09/2020	1 minuto
3221	Vigo 2	42,24	-8,73	11/07/1995	12/03/2008	21/07/2010	1 minuto

3223	Marín	42,41	-8,69	12/07/1995	13/03/2008	10/09/2020	1 minuto
3329	Huelva 5	37,13	-6,83	13/07/1995	14/03/2008	23/09/2020	1 minuto
3333	Bonanza 2	36,8	-6,34	14/07/1995	15/03/2008	29/11/2017	1 minuto
3334	Guadalquivir Puntal	36,91	-6,27	15/07/1995	16/03/2008	-	1 minuto
3335	Guadalquivir Caseta	37,1	-6,08	16/07/1995	17/03/2008	-	1 minuto
3337	Sevilla 2	37,32	-6,01	17/07/1995	-	31/02/2016	1 minuto
3450	Las Palmas 2	28,14	-1,541	18/07/1995	01/01/2009	04/11/2015	1 minuto
3459	El Hierro 2	27,78	-1,79	19/07/1995	02/01/2009	-	1 minuto
3463	Gomera	28,09	-1,711	20/07/1995	03/01/2009	-	1 minuto
3465	La Palma	28,68	-1,777	21/07/1995	04/01/2009	-	1 minuto
3469	Fuerteventura 2	28,49	-1,386	22/07/1995	05/01/2009	06/11/2015	1 minuto
3470	Lanzarote-Arrecife	28,97	-1,353	23/07/1995	06/01/2009	10/11/2015	1 minuto
3471	Tenerife 2	28,48	-1,624	24/07/1995	07/01/2009	-	1 minuto
3510	Melilla	35,29	-2,93	25/07/1995	08/01/2009	-	1 minuto
3540	Tarifa	36,01	-5,6	26/07/1995	09/01/2009	-	1 minuto
3541	Algeciras	36,18	-5,4	27/07/1995	10/01/2009	21/07/2010	1 minuto
3543	Motril 2	36,72	-3,52	28/07/1995	11/01/2009	16/05/2019	1 minuto
3545	Almería	36,83	-2,48	29/07/1995	12/01/2009	07/10/2010	1 minuto
3545_2	Almería 2	36,83	-2,48	30/07/1995	13/01/2009	22/12/2015	1 minuto
3546	Málaga 3	36,71	-4,42	31/07/1995	14/01/2009	-	1 minuto
3547	Carboneras	36,97	-1,9	01/08/1995	15/01/2009	25/06/2013	1 minuto
3651	Valencia 3	39,44	-3,1	02/08/1995	16/01/2009	-	1 minuto
3655	Sagunto	39,63	-2,1	03/08/1995	17/01/2009	-	1 minuto
3656	Gandía	38,99	-1,5	04/08/1995	18/01/2009	-	1 minuto
3756	Tarragona	41,08	1,21	05/08/1995	19/01/2009	30/05/2011	1 minuto
3758	Barcelona 2	41,34	2,17	06/08/1995	20/01/2009	-	1 minuto
3851	Palma de Mallorca	39,56	2,64	07/08/1995	21/01/2009	08/07/2015	1 minuto
3853	Alcudia	39,83	3,14	08/08/1995	22/01/2009	10/07/2015	1 minuto
3855	Formentera	38,73	1,42	09/08/1995	23/01/2009	15/09/2015	1 minuto
3856	Ibiza 2	38,91	1,45	10/08/1995	24/01/2009	15/09/2015	1 minuto
3860	Mahón	39,89	4,27	11/08/1995	25/01/2009	14/07/2015	1 minuto

TABLA 1: *MAREÓGRAFOS DE LA REDMAR OPERATIVOS. FUENTE: (CONJUNTO DE DATOS: REDMAR, 2024)*

3.2. Imágenes satelitales Sentinel-2 armonizadas

El programa Copernicus de la **Agencia Espacial Europea (ESA)** cuenta con la misión Sentinel-2, compuesta por los satélites Sentinel-2A y Sentinel-2B, cuyo objetivo es obtener imágenes satelitales de alta resolución y proporcionar datos multiespectrales para diversas aplicaciones medioambientales y territoriales.

Entre los productos disponibles se encuentran los productos **armonizados**, procesados para garantizar la coherencia radiométrica y de escala entre distintos productos y periodos temporales. Esta armonización permite que los datos sean compatibles y comparables con los provenientes de otras misiones satelitales,

como **Landsat**, lo que facilita la integración y combinación de series temporales multisensor.

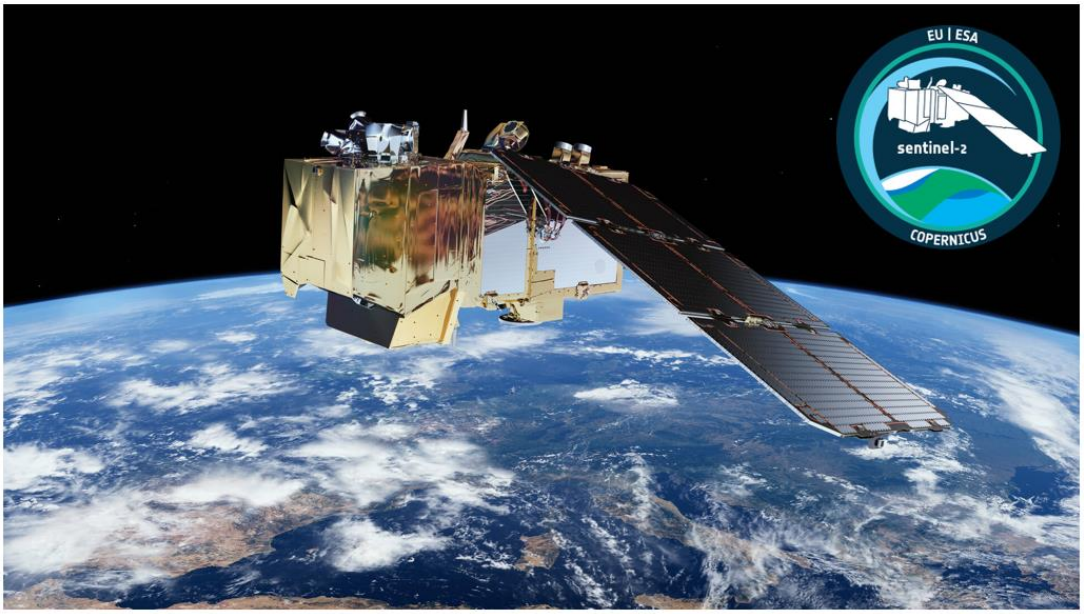

ILUSTRACIÓN 10: *SATÉLITE SENTINEL-2. FUENTE: AGENCIA ESPACIAL EUROPEA (ESA).*

ILUSTRACIÓN 11: *IMAGEN SENTINEL-2 NIVEL 1C. FUENTE: AGENCIA ESPACIAL EUROPEA (ESA).*

Para facilitar el uso y análisis de las imágenes obtenidas, los datos capturados son sometidos a distintos niveles de procesamiento, los cuales corresponden a los **niveles 1C y 2A**. Estos niveles determinan el grado de corrección y tratamiento aplicado a las imágenes, permitiendo adaptar los

productos a diferentes necesidades de análisis. En la ilustración 11 se muestra una imagen Sentinel-2 con un nivel de procesamiento 1C.

La obtención de las imágenes se realiza a través de una **Interfaz de Programación de Aplicaciones** (*Application Programming Interface*, API). Una API es un conjunto de definiciones, protocolos y herramientas que permite la comunicación entre diferentes aplicaciones o componentes de software. Su función principal es facilitar el intercambio de datos, funcionalidades o servicios entre sistemas, actuando como intermediario sin necesidad de conocer los detalles internos de su implementación. (Amazon Web Services (AWS))

3.2.1. Nivel 1C

Las imágenes se han rectificado geométricamente para eliminar distorsiones causadas por la topografía y la inclinación del sensor. Por otra parte, los valores de los píxeles representan la reflectancia en la parte superior de la atmósfera (*Top Of Atmosphere, TOA*), lo que indica que **no incluyen correcciones atmosféricas**. (Harmonized Sentinel-2 MSI: MultiSpectral Instrument, Level-1C (TOA))

3.2.2. Nivel 2A

Se trata de un producto **corregido atmosféricamente**, en el que los valores de los píxeles representan la reflectancia a nivel del suelo (*Bottom of Atmosphere, BOA*). Este tipo de reflectancia, denominada *Surface Reflectance (SR)*, es la proporcionada en los productos de nivel 2A de Sentinel-2. (Harmonized Sentinel-2 MSI: MultiSpectral Instrument, Level-2A (SR))

En la siguiente tabla se muestran las bandas correspondientes a las imágenes Sentinel-2 de los niveles 1C y 2A (tabla 2).

Nombre	Unidades	Mínimo	Máximo	Escala	Longitud de onda	Descripción
B1	-	-	-	0.0001	443.9 nm (S2A) / 442.3 nm (S2B)	Aerosoles
B2	-	-	-	0.0001	496.6 nm (S2A) / 492.1 nm (S2B)	Azul
B3	-	-	-	0.0001	560 nm (S2A) / 559 nm (S2B)	Verde
B4	-	-	-	0.0001	664.5 nm (S2A) / 665 nm (S2B)	Rojo
B5	-	-	-	0.0001	703.9 nm (S2A) / 703.8 nm (S2B)	Borde rojo 1
B6	-	-	-	0.0001	740.2 nm (S2A) / 739.1 nm (S2B)	Red Edge 2

B7	-	-	-	0.0001	782.5 nm (S2A) / 779.7 nm (S2B)	Red Edge 3
B8	-	-	-	0.0001	835.1 nm (S2A) / 833 nm (S2B)	NIR
B8A	-	-	-	0.0001	864.8 nm (S2A) / 864 nm (S2B)	Red Edge 4
B9	-	-	-	0.0001	945 nm (S2A) / 943.2 nm (S2B)	Vapor de agua
B11	-	-	-	0.0001	1,613.7 nm (S2A) / 1,610.4 nm (S2B)	SWIR 1
B12	-	-	-	0.0001	2202.4 nm (S2A) / 2185.7 nm (S2B)	SWIR 2
AOT	-	-	-	0.001	-	Grosor óptico del aerosol
WVP	cm	-	-	0.001	-	Presión del vapor de agua. Es la altura que ocuparía el agua si el vapor se condensara en un líquido y se extendiera de manera uniforme por la columna.
SCL	-	1	11	-	-	Mapa de clasificación de escenas (se oculta el valor "Sin datos" de 0)
TCI_R	-	-	-	-	-	Imagen en colores verdaderos, canal rojo
TCI_G	-	-	-	-	-	Imagen en color verdadero, canal verde
TCI_B	-	-	-	-	-	Imagen en colores verdaderos, canal azul
MSK_CLDPRB	-	0	100	-	-	Mapa de probabilidad de nubes (no disponible en algunos productos)
MSK_SNWPRB	-	0	100	-	-	Mapa de probabilidad de nevadas (falta en algunos productos)
QA10	-	-	-	-	-	Siempre vacía
QA20	-	-	-	-	-	Siempre vacía
QA60	-	-	-	-	-	Máscara de nube. Se oculta entre el 25/01/2022 y el 28/02/2024 inclusive.

TABLA 2: *BANDAS DE LAS IMÁGENES SENTINEL-2 DE LOS NIVELES 1C Y 2A. FUENTE: (HARMONIZED SENTINEL-2 MSI: MULTISPECTRAL INSTRUMENT, LEVEL-2A (SR))*

4. Metodología

El desarrollo de la herramienta propuesta se basa en un flujo de trabajo que integra **datos satelitales y mareográficos** a través de diferentes plataformas. La metodología combina la descarga de imágenes Sentinel-2, con la incorporación de información proporcionada por REDMAR y Puertos del Estado, todo ello gestionado desde una interfaz gráfica. El siguiente diagrama resume el proceso seguido, desde la definición de los parámetros hasta la obtención de los resultados, permitiendo una integración coherente de los datos para mejorar la precisión en la extracción de líneas de costa.

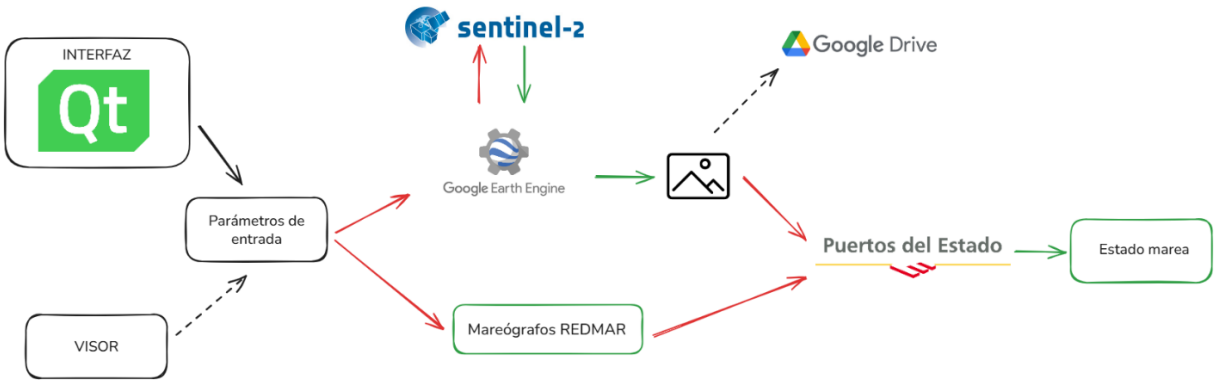

ILUSTRACIÓN 12: FLUJO DE TRABAJO. FUENTE: ELABORACIÓN PROPIA.

4.1. Flujo de trabajo

El proceso comienza con la obtención de los **parámetros de entrada**, proporcionados por el usuario a través de la interfaz gráfica desarrollada en *Qt Designer*, herramienta incluida en el *framework* Qt que permite diseñar interfaces de usuario sin necesidad de escribir código de forma manual, generando un diseño que puede integrarse fácilmente con código Python. Estos parámetros son los siguientes:

- Fecha de inicio.
- Fecha de fin.
- Porcentaje de nubosidad.
- Área de recorte.
- Nivel de procesamiento de las imágenes.

Cabe destacar que el área de recorte también puede definirse de forma interactiva mediante un **visor**, donde el usuario selecciona la zona de interés. Las coordenadas de dicha zona se transfieren automáticamente a la interfaz. Esta acción aparece representada en el esquema (ilustración 12) mediante una flecha discontinua, indicando que es una opción no obligatoria dentro del flujo de trabajo.

Con los parámetros de entrada definidos, el sistema lanza una consulta a la **API de Google Earth Engine (GEE)** para obtener las imágenes Sentinel-2 que cumplan con los criterios especificados. En el esquema, esta acción está representada con una flecha roja, que indica una petición, mientras que la obtención de las imágenes correspondientes se representa con una flecha verde, indicando la respuesta.

De forma paralela, se determina la distancia entre el centroide del área de recorte y todos los **mareógrafos del litoral español**, generando una lista con los cinco más cercanos.

Una vez obtenidas las imágenes y los mareógrafos correspondientes, el usuario puede **descargar la imagen seleccionada** y, adicionalmente, **consultar los datos de nivel del mar**. La descarga de la imagen, se realiza mediante una exportación a Google Drive, especificando previamente la combinación de bandas deseada.

Para acceder a los datos de marea, se accede de forma automática a la página web Portus de Puertos del Estado, donde selecciona el mareógrafo correspondiente y consulta los datos históricos del nivel del mar. A continuación, se generan **gráficas de series temporales**, estableciendo como fecha de inicio el día anterior a la captura de la imagen, y como fecha de fin, el día posterior. Se selecciona el tipo de gráfico resolución temporal por minuto, tras lo cual se obtiene la representación gráfica de los datos. Esta operación se representa en el esquema mediante una petición (flecha roja) al sistema de Puertos del Estado, cuya respuesta (flecha verde) corresponde al estado de la marea en ese intervalo.

La respuesta de los datos mareográficos se almacena en una carpeta de salida previamente definida por el usuario. A partir de esta información, se detectan automáticamente los picos máximos y mínimos del nivel del mar durante el periodo analizado, marcando el instante exacto en el que fue tomada la imagen Sentinel-2.

Finalmente, se genera un **informe** que resume los datos extraídos, indicando los tramos en los que la marea asciende o desciende, así como el **estado de la marea** en el momento de la captura.

4.2. Estructura del código

El código está organizado en cuatro scripts principales, cuyo contenido completo puede consultarse en el Anexo I. Para su desarrollo se ha utilizado **Python**, un lenguaje de programación versátil y fácil de aprender, muy popular en ámbitos como el análisis de datos, la automatización y la programación web, gracias a su gran variedad de librerías y su capacidad para gestionar tareas complejas de manera sencilla.

- **main.py** → Es el script principal que lanza la aplicación y coordina la ejecución del resto de módulos.
- **ui.py** → Gestiona la carga y configuración de la interfaz gráfica de usuario, definida en un archivo .ui gestionado por Qt Designer. Este script permite conectar los elementos visuales de la interfaz con la lógica del programa.
- **funciones.py** → Contiene todas las funciones necesarias para el funcionamiento del código. Incluye operaciones como la descarga de

imágenes Sentinel-2 desde GEE, la selección del mareógrafo más cercano, el *scraping* de datos de marea desde la web de Puertos del Estado, el análisis de series temporales, la generación de gráficas y la exportación de resultados.

- **constants.py** → Almacena todas las variables constantes utilizadas, como rutas de archivos, configuraciones para Selenium, selectores XPath, URLs y directorios de entrada y salida.

4.3. Desarrollo de la aplicación

En este apartado se describe el proceso de desarrollo de la aplicación, haciendo especial énfasis en su interfaz gráfica. El código se encuentra detallado en el Anexo I, junto con una tabla (tabla 8) que incluye una breve descripción de las funciones principales empleadas.

La ilustración 13 muestra la pantalla principal de la aplicación, diseñada para ser intuitiva y funcional. Su objetivo es facilitar el acceso rápido a las principales funcionalidades, compuesta por diferentes elementos que permiten al usuario obtener los resultados deseados.

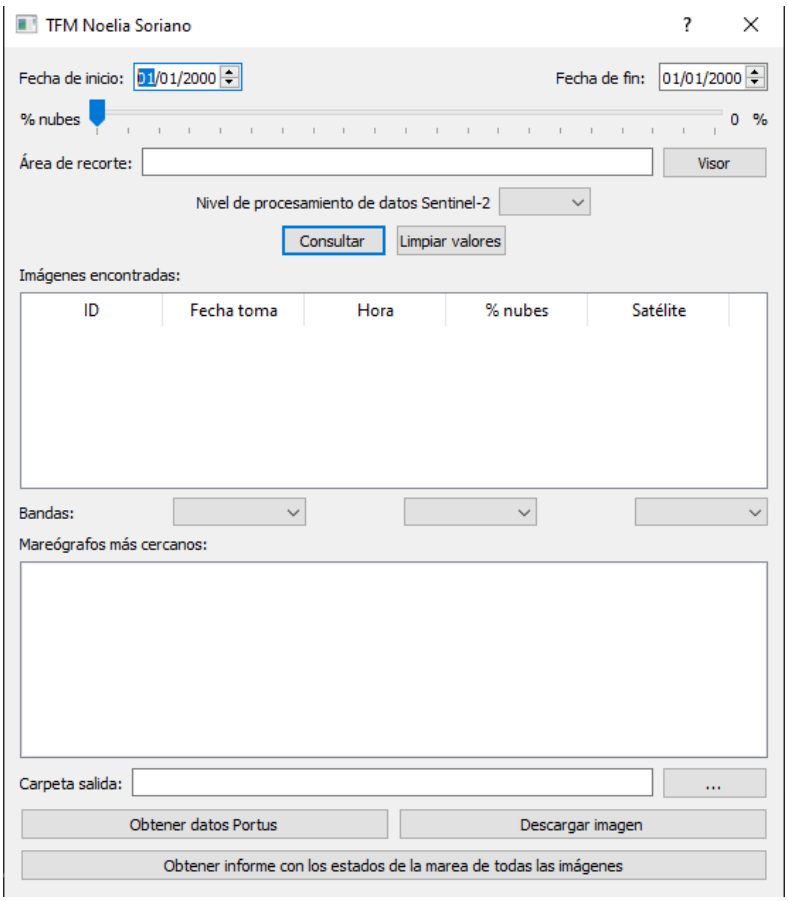

ILUSTRACIÓN 13: *INTERFAZ GRÁFICA. FUENTE: ELABORACIÓN PROPIA.*

En la parte superior (ilustración 14) se encuentran los elementos interactivos, también conocidos como **widgets**, donde se van a definir los parámetros de entrada para poder realizar la consulta. Entre ellos se incluyen dos selectores de fechas, que permiten establecer el rango temporal (inicio y fin), un control deslizante para indicar el porcentaje máximo de nubosidad permitido, un cuadro de texto para especificar el área de recorte y un menú desplegable para seleccionar el nivel de procesamiento de los datos Sentinel-2, según los niveles descritos en el apartado 3.2.

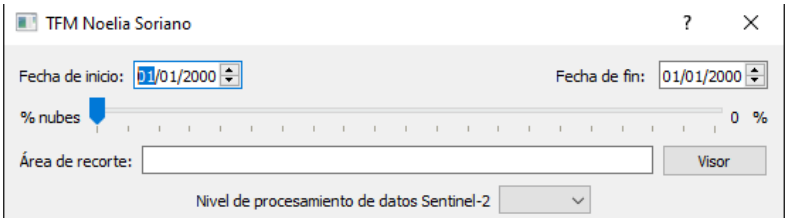

ILUSTRACIÓN 14: *WIDGETS DE LOS PARÁMETROS DE ENTRADA. FUENTE: ELABORACIÓN PROPIA.*

Adicionalmente, se dispone de un botón que da acceso a un visor interactivo (ilustración 15), el cual permite definir la zona de interés de manera visual. En este visor, el usuario puede hacer zoom sobre el área deseada y digitalizar un rectángulo, que posteriormente puede editarse o eliminarse según sea necesario. Una vez definida la geometría, se deben transmitir sus coordenadas al sistema presionando el botón “Obtener coordenadas”.

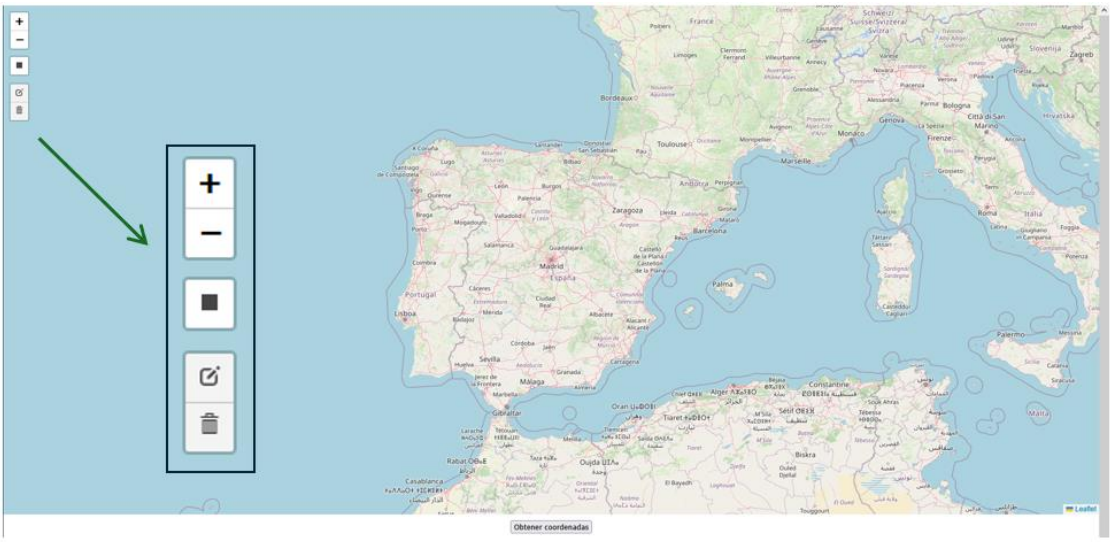

ILUSTRACIÓN 15: *VISOR PARA SELECCIONAR EL ÁREA DE RECORTE. FUENTE: ELABORACIÓN PROPIA.*

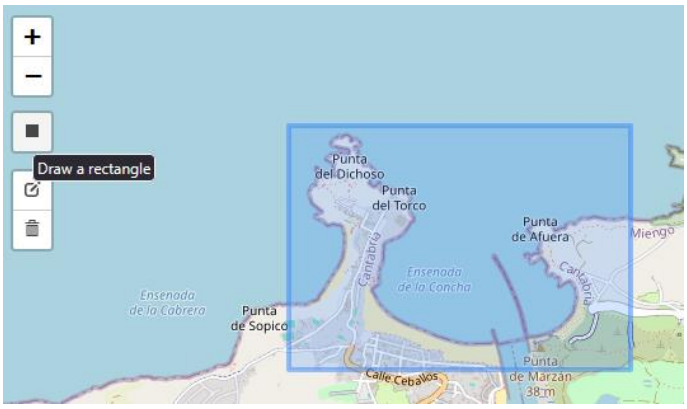

ILUSTRACIÓN 16: HERRAMIENTA DE DIGITALIZAR FORMA. FUENTE: ELABORACIÓN PROPIA.

ILUSTRACIÓN 17: HERRAMIENTA DE EDICIÓN DE FORMA. FUENTE: ELABORACIÓN PROPIA.

ILUSTRACIÓN 18: HERRAMIENTA DE ELIMINAR FORMA. FUENTE: ELABORACIÓN PROPIA.

Para la programación de esta página web interactiva, desarrollada con **HTML**, **CSS** y **JavaScript**, y que puede consultarse en el apartado 10.5 del Anexo I, se han empleado las librerías Leaflet, destinada a la visualización de mapas interactivos, y Leaflet Draw, una extensión que permite la digitalización de elementos sobre el mapa.

Estas tecnologías web fundamentales cumplen roles complementarios: HTML estructura el contenido definiendo elementos como textos, imágenes y

enlaces; CSS se encarga del diseño visual, aplicando estilos como colores, fuentes y disposición; y JavaScript añade interactividad y dinamismo, posibilitando la respuesta a eventos, la modificación del contenido en tiempo real y la comunicación con servicios externos.

Una vez establecidos los parámetros iniciales, se dispone de dos botones que permiten, respectivamente, iniciar la consulta o limpiar los valores introducidos en los campos de entrada.

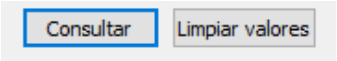

ILUSTRACIÓN 19: *BOTONES PARA INICIAR LA CONSULTA O LIMPIAR LOS VALORES DE ENTRADA.*

Al presionar el botón de consulta, se ejecuta una solicitud a la API de GEE mediante la función **obtener_datos_sentinel**, que devuelve un listado de imágenes satelitales que cumplen con los parámetros definidos previamente por el usuario. Simultáneamente, a partir de un archivo GeoJSON que contiene la localización de todos los mareógrafos del litoral español, se calcula la distancia geodésica entre el centroide del área de recorte y cada uno de estos puntos, generando una lista con los **cinco mareógrafos más cercanos**. La distancia geodésica representa la distancia más corta entre dos puntos sobre la superficie terrestre, considerando la curvatura del planeta, y es la medida más precisa para este tipo de cálculos espaciales. Toda esta información se presenta de forma automática en la interfaz de la aplicación, permitiendo al usuario visualizar los resultados.

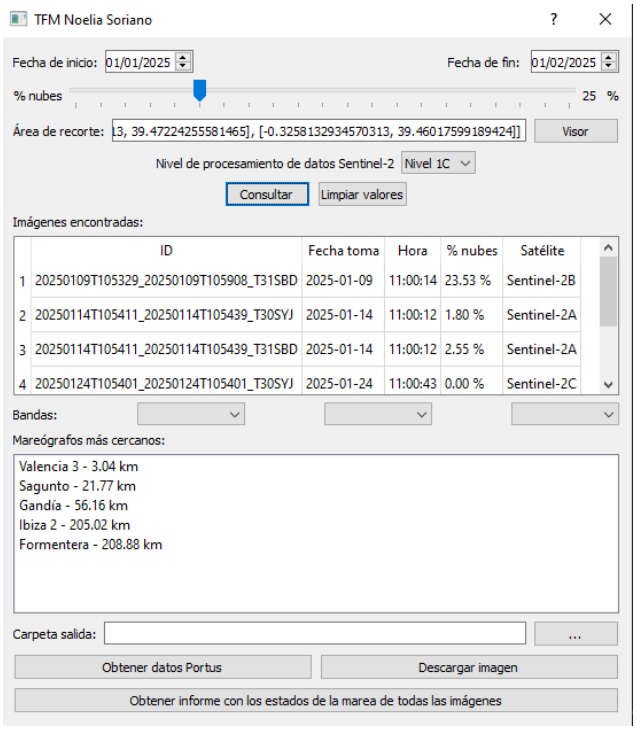

ILUSTRACIÓN 20: *RESULTADO DE LA CONSULTA. FUENTE: ELABORACIÓN PROPIA.*

Una vez se han obtenido las listas de imágenes y mareógrafos, el usuario dispone de tres opciones. Una de ellas consiste en presionar el botón "**Descargar imagen**", para lo cual es necesario indicar previamente las bandas espectrales que se desean combinar. El resultado de este proceso se almacenará automáticamente en Google Drive, y se mostrará un mensaje confirmando que la descarga se ha realizado correctamente. En caso de que no se haya seleccionado ninguna imagen o no se hayan definido las bandas, la aplicación mostrará un mensaje de advertencia.

La descarga se realiza mediante la utilización de la librería ee (Earth Engine Python API), que permite interactuar con la plataforma de GEE desde Python. Todo este procedimiento está implementado en las funciones "descargar_imagen" y "descargar_imagen_gee".

Otra de las opciones disponibles se inicializa al pulsar el botón "**Obtener datos Portus**", que permite consultar automáticamente los datos del mareógrafo seleccionado. Para ello, la aplicación accede a la web de Portus utilizando la librería Selenium, simulando una búsqueda en el portal correspondiente y extrayendo la respuesta de datos al minuto del nivel del mar. Esta información se procesa para generar una gráfica temporal con los niveles registrados, en la que se destacan los picos máximos y mínimos detectados, los cuales se calculan empleando la función *find_peaks* desde el módulo *signal* de la librería *SciPy*. Los resultados se almacenan en formato PNG y CSV, y se muestra un mensaje confirmando que la respuesta ha sido obtenida correctamente. En el caso de no haber seleccionado ninguna imagen, mareógrafo o determinado una carpeta de salida, la aplicación emitirá un mensaje de advertencia (ilustración 21) y el proceso no podrá ser inicializado.

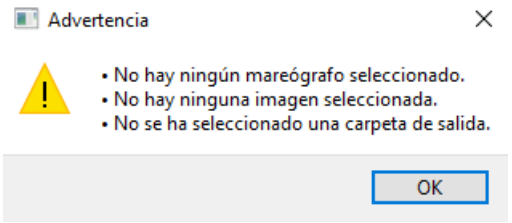

ILUSTRACIÓN 21: *MENSAJE DE ADVERTENCIA AL NO HABER SELECCIONADO UN MAREÓGRAFO, IMAGEN O CARPETA DE SALIDA. FUENTE: ELABORACIÓN PROPIA.*

La última herramienta disponible, al presionar el botón "**Obtener informe con los estados de la marea de todas las imágenes**", permite generar un informe a partir de la descarga de datos desde Portus para todas las imágenes resultantes de la búsqueda por parámetros. El flujo de trabajo es similar al del botón comentado anteriormente; sin embargo, en este caso, la respuesta se centra exclusivamente en determinar el estado de la marea correspondiente a cada imagen, sin detallar los tramos horarios presentes en el periodo indicado.

Al igual que en el procedimiento anterior, es necesario seleccionar previamente un mareógrafo y una carpeta de salida donde se almacenarán los resultados generados.

5. Resultados

En este apartado se presentan los resultados obtenidos tras la aplicación de la metodología descrita en el apartado anterior.

5.1. Imágenes descargadas mediante los parámetros definidos por el usuario

La ilustración 22 muestra un ejemplo representativo de imágenes Sentinel-2 descargadas mediante el proceso automatizado desarrollado, correspondientes a la costa del paseo marítimo de Areaga (Bakio, Vizcaya). Las imágenes, recortadas según el área de interés definida, permiten apreciar la calidad y resolución obtenidas. En este caso, se han utilizado combinaciones de bandas B4, B3 y B2 para generar una visualización en color verdadero. Dado que estas bandas cuentan con una resolución espacial de 10 metros, las imágenes resultantes permiten observar con mayor detalle la morfología costera. Además, se distinguen ligeras diferencias entre los distintos niveles de procesamiento aplicados.

ILUSTRACIÓN 22: *IMÁGENES SENTINEL-2 EN COLOR VERDADERO CON NIVEL DE PROCESAMIENTO 2A Y 1C, RESPECTIVAMENTE. FUENTE: SENTINEL-2.*

5.2. Obtención de datos de Portus para una imagen y mareógrafo seleccionado

La selección de una imagen Sentinel-2 y un mareógrafo, permite realizar una consulta a la plataforma de datos de Portus para obtener información mareográfica correspondiente al instante de captura de la imagen. Como resultado de este proceso, se generan **tres archivos** diferentes que contribuyen al análisis conjunto de la dinámica costera y las condiciones de marea en el momento de adquisición de la imagen.

El primer archivo contiene la **respuesta original** devuelta tras generar el gráfico de datos al minuto (ilustración 23), donde se almacenan los datos de nivel del mar registrados a intervalos de un minuto. Esta información es fundamental para la construcción de una serie temporal de alta resolución, la cual sirve como base para la detección de eventos significativos.


```
{
  "datos": [
    {
      "fecha": "2023-01-01 00:00:00.0",
      "Datos al minuto": "3.85"
    },
    {
      "fecha": "2023-01-01 00:01:00.0",
      "Datos al minuto": "3.864"
    },
    {
      "fecha": "2023-01-01 00:02:00.0",
      "Datos al minuto": "3.834"
    },
    {
      "fecha": "2023-01-01 00:03:00.0",
      "Datos al minuto": "3.849"
    }
  ],
  "parametro": "Nivel del mar",
  "unidad": "m",
  "series": [
    "Datos al minuto"
  ]
}
```


ILUSTRACIÓN 23: *MUESTRA DE LA ESTRUCTURA DE LA RESPUESTA DE PORTUS. FUENTE: ELABORACIÓN PROPIA.*

El segundo archivo incluye el **gráfico generado a partir de la respuesta** de Portus, donde se identifican los picos máximos y mínimos de la marea durante el periodo de referencia. Además, se señala el instante exacto en el que fue adquirida la imagen seleccionada, lo que permite evaluar visualmente en qué punto del ciclo mareal se encontraba la costa en ese momento (ilustraciones 24, 25 y 26).

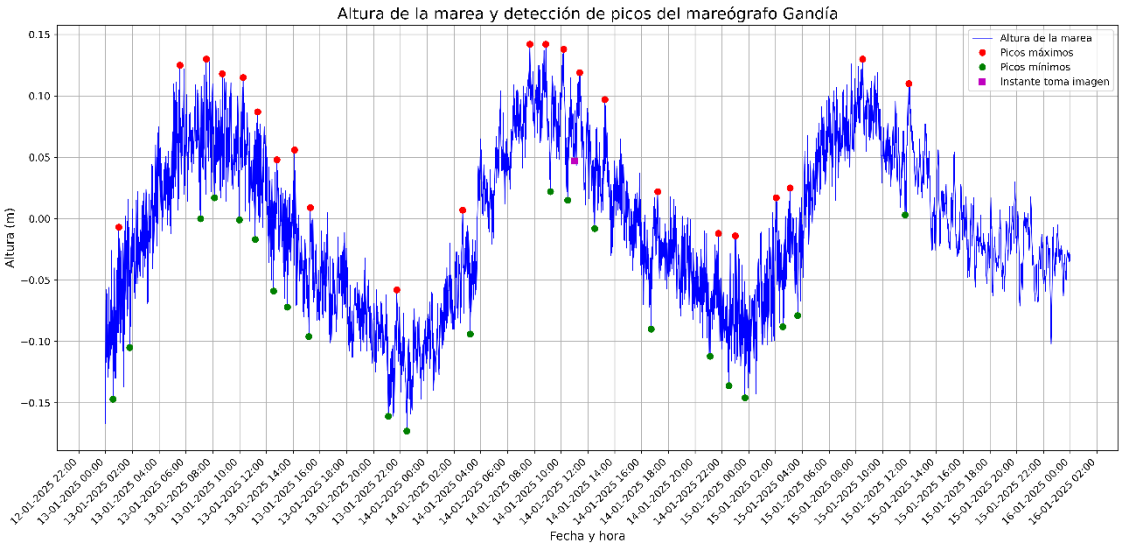

ILUSTRACIÓN 24: *GRÁFICO DE ALTURA DE LA MAREA Y DETECCIÓN DE PICOS DEL MAREÓGRAFO DE GANDÍA ENTRE EL 12/01/2025 Y EL 16/01/2025. FUENTE: ELABORACIÓN PROPIA.*

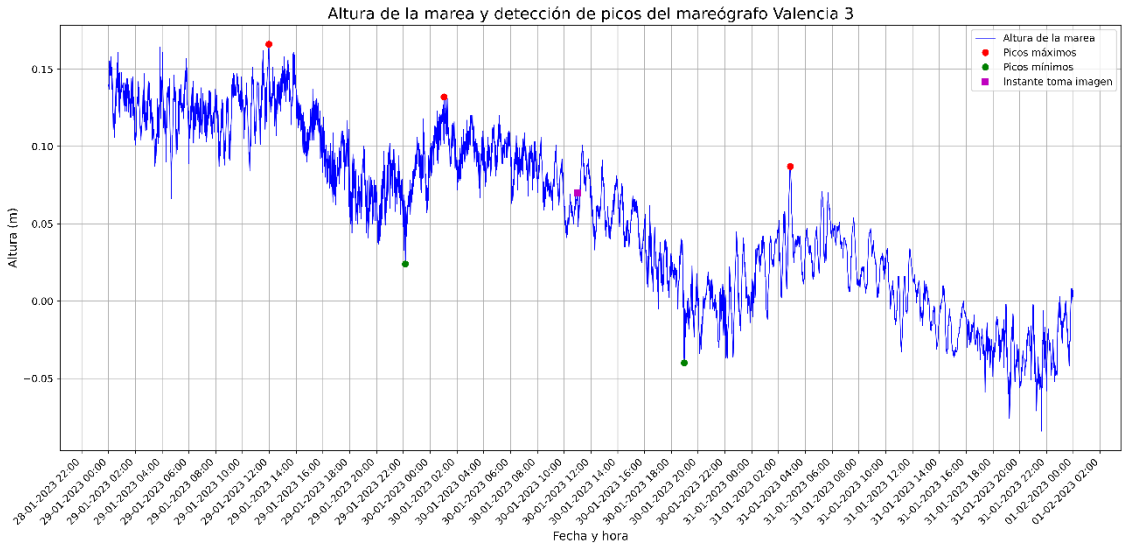

ILUSTRACIÓN 25: *GRÁFICO DE ALTURA DE LA MAREA Y DETECCIÓN DE PICOS DEL MAREÓGRAFO DE VALENCIA 3 ENTRE EL 28/01/2023 Y EL 01/02/2023. FUENTE: ELABORACIÓN PROPIA.*

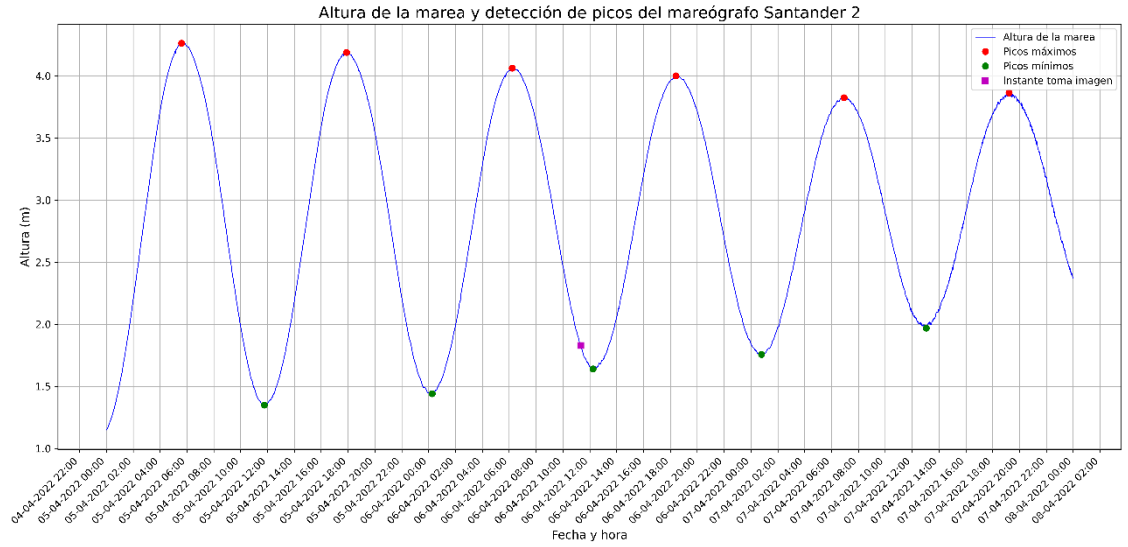

ILUSTRACIÓN 26: *GRÁFICO DE ALTURA DE LA MAREA Y DETECCIÓN DE PICOS DEL MAREÓGRAFO DE SANTANDER 2 ENTRE EL 04/04/2022 Y EL 08/04/2022. FUENTE: ELABORACIÓN PROPIA.*

El análisis comparativo entre los gráficos de los mareógrafos situados en el mar Mediterráneo (Valencia 3 y Gandía) y el océano Atlántico (Santander 2) evidencia las diferencias en el régimen mareal entre estas costas. En el caso del mareógrafo de Santander, se observa un patrón semidiurno claro, con dos plenamares y dos bajamares al día, además de una amplitud significativa que supera los 3 metros. Por el contrario, los gráficos de Valencia y Gandía muestran amplitudes más reducidas y patrones más irregulares y ruidosos.

Por último, el tercer archivo consiste en un **informe** en formato CSV que **resume los tramos de subida y bajada de la marea** detectados en el intervalo analizado. Este informe no solo identifica los periodos de flujo y reflujo, sino que

también determina el estado de la marea en el instante en que se capturó la imagen. Las Ilustraciones 27 y 28 muestran ejemplos representativos de este informe generado.

```
INFORME DE INTERVALOS DE MAREA
Mareógrafo: Gandia
Fecha de inicio del análisis: 2025-01-13 00:00:00
Fecha de fin del análisis: 2025-01-15 23:59:00
Instante de toma de imagen: 2025-01-14 11:00:12
Estado de la marea en el instante de toma de imagen: Sube la marea

DETALLE DE LOS TRAMOS DETECTADOS:

Tramo 1:
  - Inicio: 2025-01-13 00:00:00
  - Fin: 2025-01-13 00:33:00
  - Desde: 2025-01-13 00:00:00 - -0.167 m
  - Hasta: 2025-01-13 00:33:00 - -0.147 m
  - Cambio de marea: Baja la marea (tramo inicial)
Tramo 2:
  - Inicio: 2025-01-13 00:33:00
  - Fin: 2025-01-13 00:59:00
  - Desde: 2025-01-13 00:33:00 - -0.147 m
  - Hasta: 2025-01-13 00:59:00 - -0.007 m
  - Cambio de marea: Sube la marea
Tramo 3:
  - Inicio: 2025-01-13 00:59:00
  - Fin: 2025-01-13 01:48:00
  - Desde: 2025-01-13 00:59:00 - -0.007 m
  - Hasta: 2025-01-13 01:48:00 - -0.105 m
  - Cambio de marea: Baja la marea
```

ILUSTRACIÓN 27: *MUESTRA DEL INFORME GENERADO DEL MAREÓGRAFO GANDÍA ENTRE EL 13/01/2025 Y EL 15/01/2025. FUENTE: ELABORACIÓN PROPIA.*

```
INFORME DE INTERVALOS DE MAREA
Mareógrafo: Santander 2
Fecha de inicio del análisis: 2023-02-14 00:00:00
Fecha de fin del análisis: 2023-02-16 23:59:00
Instante de toma de imagen: 2023-02-15 11:19:04
Estado de la marea en el instante de toma de imagen: Baja la marea

DETALLE DE LOS TRAMOS DETECTADOS:

Tramo 1:
  - Inicio: 2023-02-14 00:00:00
  - Fin: 2023-02-14 02:41:00
  - Desde: 2023-02-14 00:00:00 - 2.686 m
  - Hasta: 2023-02-14 02:41:00 - 1.837 m
  - Cambio de marea: Baja la marea (tramo inicial)
Tramo 2:
  - Inicio: 2023-02-14 02:41:00
  - Fin: 2023-02-14 09:14:00
  - Desde: 2023-02-14 02:41:00 - 1.837 m
  - Hasta: 2023-02-14 09:14:00 - 3.700 m
  - Cambio de marea: Sube la marea
Tramo 3:
  - Inicio: 2023-02-14 09:14:00
  - Fin: 2023-02-14 15:35:00
  - Desde: 2023-02-14 09:14:00 - 3.700 m
  - Hasta: 2023-02-14 15:35:00 - 1.967 m
  - Cambio de marea: Baja la marea
```

ILUSTRACIÓN 28: *MUESTRA DEL INFORME GENERADO DEL MAREÓGRAFO SANTANDER 2 ENTRE EL 14/02/2023 Y EL 15/02/2023. FUENTE: ELABORACIÓN PROPIA.*

5.3. Obtención del informe de las imágenes resultantes

La última herramienta desarrollada en la interfaz permite generar un informe que asocia el estado de la marea a cada una de las imágenes obtenidas tras una búsqueda basada en parámetros definidos por el usuario. Además, almacena la respuesta en crudo que contiene los datos de nivel del mar a resolución temporal de un minuto, correspondiente al periodo en el que existen imágenes disponibles.

Este informe facilita una visualización rápida y estructurada de las condiciones mareográficas presentes en el instante de adquisición de cada imagen. Esta información resulta especialmente útil para realizar filtrados por estado de marea o para comparar distintas imágenes en función de las condiciones oceánicas locales. A continuación, se presenta un ejemplo del informe generado, que evidencia la utilidad de esta herramienta como apoyo al análisis combinado de datos satelitales y mareográficos.

```
ID,Fecha,Hora,Nubosidad,Satelite,Estado Marea
20250109T105329_20250109T105908_T30SYJ,2025-01-09,11:00:14,28.18 %,Sentinel-2B,Baja la marea
20250109T105329_20250109T105908_T31SBD,2025-01-09,11:00:14,26.42 %,Sentinel-2B,Baja la marea
20250114T105411_20250114T105439_T30SYJ,2025-01-14,11:00:12,3.92 %,Sentinel-2A,Sube la marea
20250114T105411_20250114T105439_T31SBD,2025-01-14,11:00:12,6.20 %,Sentinel-2A,Sube la marea
20250124T105401_20250124T105401_T30SYJ,2025-01-24,11:00:43,0.01 %,Sentinel-2C,Baja la marea
20250124T105401_20250124T105401_T31SBD,2025-01-24,11:00:42,0.00 %,Sentinel-2C,Baja la marea
```

ILUSTRACIÓN 29: *RESULTADO DEL INFORME DEL ESTADO DE LA MAREA DEL CONJUNTO DE IMÁGENES. FUENTE: ELABORACIÓN PROPIA.*

6. Presupuesto

Para estimar el presupuesto del proyecto se han considerado únicamente los **costes directos** asociados a su desarrollo. Entre ellos se incluyen tanto el uso de herramientas informáticas como la retribución del personal técnico involucrado.

En cuanto al software empleado para el desarrollo del código de la herramienta, se ha utilizado el lenguaje de programación **Python**, apoyado en librerías de uso libre, así como en **APIs** gratuitas como **GEE**, que han permitido acceder y descargar imágenes satelitales sin coste. La interfaz gráfica se ha diseñado utilizando **Qt Designer**, una herramienta de código abierto que facilita la creación visual de interfaces de usuario sin necesidad de programación directa. Por otro lado, para la elaboración de los planos se ha empleado el software **QGIS**, una plataforma de código abierto y acceso libre, cuya utilización no implica costes económicos adicionales.

Respecto a la estimación salarial, se ha tomado como referencia la **tabla retributiva vigente del convenio colectivo de ingenierías y oficinas técnicas** (tabla 3).

Grupo profesional	Nivel salarial	Personal Técnico	Personal Administrativo
		Puestos de trabajo (relación no exhaustiva)	Puestos de trabajo (relación no exhaustiva)
I	1	INGENIERO; ARQUITECTO; DOCTOR; LICENCIADO; TITULADO 2.º Y 3.º CICLO UNIVERSITARIO; GRADUADO UNIVERSITARIO CON MÁSTER UNIVERSITARIO OFICIAL HABILITANTE O MÁSTER UNIVERSITARIO OFICIAL (MÍN. 60 ECTS), CUANDO APORTE ESPECIALIZACIÓN Y COMPETENCIAS PROFESIONALES NECESARIAS PARA EL DESEMPEÑO DEL PUESTO DE TRABAJO. ANALISTA.	LICENCIADO; TITULADO 2.º Y 3.º CICLO UNIVERSITARIO; GRADUADO UNIVERSITARIO CON MÁSTER UNIVERSITARIO OFICIAL HABILITANTE O MÁSTER UNIVERSITARIO OFICIAL (MÍN. 60 ECTS), CUANDO APORTE ESPECIALIZACIÓN Y COMPETENCIAS PROFESIONALES NECESARIAS PARA EL DESEMPEÑO DEL PUESTO DE TRABAJO.
	2	GRADUADO UNIVERSITARIO; INGENIERO TÉCNICO; ARQUITECTO TÉCNICO; APAREJADOR; DIPLOMADO UNIVERSITARIO; TITULADO 1er. CICLO UNIVERSITARIO.	GRADUADO UNIVERSITARIO; DIPLOMADO UNIVERSITARIO; TITULADO 1er. CICLO UNIVERSITARIO
II	3	TÉCNICO DE CÁLCULO O DISEÑO; PROGRAMADOR INFORMÁTICO.	JEFE 1.ª ADMINISTRATIVO
	4	DELINEANTE-PROYECTISTA.	JEFE 2.ª ADMINISTRATIVO
III	5	DELINEANTE; TÉCNICO 1.ª; TÉCNICO MODELADOR BIM; TÉCNICO INFORMÁTICO.	OFICIAL 1.ª ADMINISTRATIVO; TRADUCTOR E INTÉRPRETE NO JURADO DE UNO O MÁS IDIOMAS EXTRANJEROS
	6	TÉCNICO 2.ª	OFICIAL 2.ª ADMINISTRATIVO
IV	7	VIGILANTE/SUPERVISOR/INSPECTOR DE OBRA; AUXILIAR TÉCNICO.	AUXILIAR ADMINISTRATIVO; TELEFONISTA-RECEPCIONISTA
	8	AYUDANTE.	

TABLA 3: *ESQUEMA DEL AGRUPAMIENTO DE LOS PUESTOS DE TRABAJO POR GRUPO PROFESIONAL Y NIVEL SALARIAL. FUENTE: BOLETÍN OFICIAL DEL ESTADO (BOE).*

Nivel salarial	Tabla salarial art. 33		Plus Convenio según art. 38 Convenio	Total anual
	Mes x 14	Anual		
1	1.827,30	25.582,20	2.444,61.	28.026,81
2	1.377,65	19.287,10	2.444,61.	21.731,71
3	1.328,44	18.598,16	2.444,61.	21.042,77
4	1.217,93	17.051,02	2.444,61.	19.495,63
5	1.088,23	15.235,22	2.444,61.	17.679,83
6	937,58	13.126,12	2.444,61.	15.570,73
7	906,12	12.685,68	2.444,61.	15.130,29
8	905,39	12.675,46	2.444,61.	15.120,07
9	905,39	12.675,46	2.444,61.	15.120,07

TABLA 4: *TABLA SALARIAL Y PLUS CONVENIO AÑO 2023. FUENTE: BOLETÍN OFICIAL DEL ESTADO (BOE).*

El nivel salarial de un personal técnico en Ingeniería en Geomática y Topografía con máster universitario oficial es el siguiente:

Nivel Salarial	1
Personal técnico	INGENIERO; ARQUITECTO; DOCTOR; LICENCIADO; TITULADO 2.º Y 3.º CICLO UNIVERSITARIO; GRADUADO UNIVERSITARIO CON MÁSTER UNIVERSITARIO OFICIAL HABILITANTE O MÁSTER UNIVERSITARIO OFICIAL (MÍN. 60 ECTS)

***TABLA 5:** NIVEL SALARIAL DE UN INGENIERO EN GEOMÁTICA Y TOPOGRAFÍA CON MÁSTER UNIVERSITARIO OFICIAL. FUENTE: ELABORACIÓN PROPIA.*

Para calcular el salario de un Ingeniero en Geomática y Topografía con máster universitario oficial, se ha realizado el siguiente procedimiento:

- Conociendo el salario base anual y el plus convenio, se ha calculado el salario bruto anual:

Salario bruto anual = salario anual + plus convenio (1)
Salario bruto anual = 25.582,20 + 2.444,61
Salario bruto anual = 28.026,81 €

- La contribución a la Seguridad Social equivale al 40% del salario bruto anual:

*Seguridad social = salario bruto * 0,4 (2)*
*Seguridad social = 28.026,81 * 0,4*
Seguridad social = 11.210,72 €

- El coste anual para la empresa es el siguiente:

Coste anual = salario bruto anual + seguridad social (3)
Coste anual = 28.026,81 + 11.210,72
Coste anual = 39.237,53 €

- Dividiendo el coste anual entre 12 meses, se obtiene el coste mensual para la empresa:

Coste mensual = coste anual / 12 meses (4)
Coste mensual = 39.237,53 / 12
Coste mensual = 3.269,79 €

- Para el cálculo del coste diario se divide entre los 20 días laborables:

Coste diario = coste mensual / 20 días laborables (5)
Coste diario = 3.269,79 / 20
Coste diario = 163,49 €

- Finalmente, el coste por hora es el siguiente:

Coste por hora = coste diario / 8 horas (6)
Coste por hora = 163,49 / 8
Coste por hora = 20,44 €

Salario mensual	1.827,30 €
Salario anual	25.582,20 €
Plus convenio	2.444,61 €

Salario bruto	28.026,81 €
Seguridad Social	11.210,72 €

Coste anual	39.237,53 €
Coste mensual	3.269,79 €
Coste diario	163,49 €

€/hora	20,44 €

TABLA 6: CÁLCULO DEL SALARIO POR HORAS DE UN INGENIERO EN GEOMÁTICA Y TOPOGRAFÍA CON MÁSTER UNIVERSITARIO OFICIAL. FUENTE: ELABORACIÓN PROPIA.

A continuación, se presenta un atabla en la que se detalla la distribución del tiempo invertido en cada actividad del proyecto, así como el coste total estimado en función de las horas dedicadas.

Actividad	Horas	Coste (€)
Búsqueda y análisis de información	63	1.287,48 €
Programación del código	215	4.393,79 €
Análisis de los resultados	50	1.021,81 €
Realización de la cartografía	2	40,87 €
Resolución de dudas y reuniones	10	204,36 €
Redacción del proyecto	60	1.226,17 €
TOTAL	**400**	**8.174,49 €**

TABLA 7: CÁLCULO DEL SALARIO POR HORAS DE UN INGENIERO EN GEOMÁTICA Y TOPOGRAFÍA CON MÁSTER UNIVERSITARIO OFICIAL. FUENTE: ELABORACIÓN PROPIA.

7. Conclusiones

Al concluir el desarrollo del proyecto y alcanzar los objetivos planteados, se han podido extraer diversas conclusiones relevantes.

La delimitación precisa y actualizada de la línea de costa es fundamental para una gestión eficaz del litoral, especialmente en regiones meso y macro mareales como el Cantábrico y el Atlántico español. En un contexto marcado por los impactos del cambio climático, contar con información fiable sobre la dinámica costera se vuelve imprescindible para la planificación y protección de estas zonas vulnerables.

Si bien los métodos automáticos existentes permiten extraer la línea de costa a partir de imágenes satelitales, aún presentan limitaciones notables en entornos dinámicos. Por ello, se hace evidente la necesidad de incorporar datos adicionales, como los registros mareográficos, que permitan corregir errores y mejorar la precisión de los análisis.

En este marco, el desarrollo de una herramienta que integra imágenes Sentinel-2 con datos de marea provenientes de la red REDMAR supone un

avance tecnológico significativo. Esta innovación no solo contribuye al conocimiento del medio costero, sino que también se alinea con varios Objetivos de Desarrollo Sostenible (ODS), entre ellos el ODS 9 (Industria, innovación e infraestructura), el ODS 11 (Ciudades y comunidades sostenibles), el ODS 13 (Acción por el clima) y el ODS 14 (Vida submarina).

Una de las principales fortalezas de esta herramienta es la automatización del flujo de trabajo completo, desde la selección de imágenes y áreas de estudio hasta la descarga de datos y generación de informes. Además, su diseño multiplataforma, que combina una interfaz gráfica creada con Qt Designer y un visor web interactivo con Leaflet, permite a los usuarios seleccionar zonas de interés y obtener resultados sin requerir conocimientos técnicos avanzados.

La integración con fuentes oficiales, como la API de Google Earth Engine y la web de Puertos del Estado (Portus), garantiza el acceso a datos actualizados, fiables y oficiales tanto de origen satelital como mareográfico. Esta combinación permite mejorar la exactitud al calcular la posición de la línea de costa observada en función del estado de la marea en el momento exacto de la captura de la imagen.

En conjunto, esta herramienta es una solución eficaz para apoyar la gestión y planificación de las zonas costeras, al ofrecer datos precisos y actualizados que facilitan la identificación de cambios y riesgos en el litoral. Así, contribuye a una toma de decisiones más fundamentada y orientada hacia la sostenibilidad y la resiliencia de los ecosistemas costeros.

8. Trabajos futuros y posibles mejoras

A partir de los resultados obtenidos, se identifican una serie de líneas de mejora y ampliación que podrían abordarse en futuras investigaciones.

1. **Mejora en la detección de los picos máximos y mínimos del nivel del mar.** Perfeccionar los algoritmos de análisis temporal para identificar con mayor precisión los valores de pleamar y bajamar en el litoral mediterráneo, permitiendo una mejor interpretación del estado del mar en el momento de captura de la imagen.

2. **Incorporación de otros tipos de datos históricos y variables adicionales.** Integrar nuevas fuentes de datos oceanográficos o meteorológicos (como presión atmosférica, oleaje o viento), lo que permitiría ajustar la posición de la línea de costa considerando un contexto ambiental más completo.

3. **Ampliación de la extensión geográfica.** Adaptar la herramienta para su aplicación en otras regiones costeras más allá del litoral español, incorporando redes de mareógrafos internacionales y ajustando la lógica espacial del sistema a distintas realidades geográficas.

4. **Optimización del rendimiento y tiempos de procesado.** Mejorar la eficiencia del flujo de trabajo mediante técnicas de paralelización, simplificación de procesos intermedios o integración con entornos de computación en la nube, permitiendo así una mayor escalabilidad del sistema.

5. **Consideración del run-up en la estimación de la línea de costa.** Incorporación de la altura máxima alcanzada por las olas sobre la superficie terrestre como variable adicional para ajustar la posición de la línea de costa. Permitiendo tener en cuenta el efecto combinado de marea, oleaje y *swash* (movimiento de ida y vuelta del agua sobre la playa causado por el impacto de las olas en la rompiente).

9. Bibliografía

A.H.F. KLEIN, L.P. ALMEIDA, R.Q GONÇALVES, P. UNAS, C.H. BUGHI, M.F. TOMASI, W.L.L COSTA, L. POOL, R.L.S DAZZI, A.M.R. FERNANDES, R. LYRA, J.F. CARVALHO, W.C.G. OLIVEIRA, and J.B.R. TALLMANN. (Marzo de 2023). Coastal Analyst System from Space Imagery Engine (CASSIE-CORE).

Amazon Web Services (AWS). (s.f.). *¿Qué es una interfaz de programación de aplicaciones (API)?* Obtenido de https://aws.amazon.com/es/what-is/api/

Cabezas-Rabadán, C., Pardo-Pascual, J.E., Palomar-Vázquez, J. (23 de Enero de 2025). *A remote monitoring approach for coastal engineering projects.* Obtenido de https://doi.org/10.1038/s41598-025-86485-y

Google Earth Engine. (s.f.). *Harmonized Sentinel-2 MSI: MultiSpectral Instrument, Level-1C (TOA)*. Obtenido de Earth Engine Data Catalog: https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_HARMONIZED

Google Earth Engine. (s.f.). *Harmonized Sentinel-2 MSI: MultiSpectral Instrument, Level-2A (SR)*. Obtenido de Earth Engine Data Catalog: https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_SR_HARMONIZED

Grupo de Cartografía Geoamtiental y Teledetección (CGAT). (s.f.). Obtenido de https://cgat.webs.upv.es/

J.E. Pardo-Pascual, J. Almonacid-Caballer, C. Cabezas-Rabadán, A. Fernández-Sarría, C. Armaroli, P. Ciavola, J. Montes, P.E. Souto-Ceccon, J. Palomar-Vázquez. (2 de Diciembre de 2023). Assessment of satellite-derived shorelines automatically extracted from Sentinel-2 imagery using SAET. Obtenido de https://www.sciencedirect.com/science/article/pii/S0378383923001503

Martín, L. M. (31 de Marzo de 2009). *Rodamedia.* Obtenido de https://www.rodamedia.com/navastro/mareas/mareas_LMederos.pdf

Ministerio para la transición ecológica y el reto demográfico. (Noviembre de 2004). *Impactos en la costa española por efecto del cambio climático.* Obtenido de https://www.miteco.gob.es/es/cambio-climatico/temas/impactos-vulnerabilidad-y-adaptacion/plan-nacional-adaptacion-cambio-climatico/impactos-en-la-costa-espanola-por-efecto-del-cambio-climatico.html

Naciones Unidas. (s.f.). *Objetivos de Desarrollo Sostenible*. Obtenido de https://www.un.org/sustainabledevelopment/es/

Ojeda Zújar J., Díaz Cuevas M. P., Prieto Campos A. y Álvarez Francoso J. I. (Julio de 2013). Línea de costa y Sistemas de Información Geográfica: modelo de datos para la caracterización y cálculo de indicadores en la costa andaluza. 37-52.

Palomar-Vázquez, J., Almonacid-Caballer, J., Pardo-Pascual, J. E., & Sanchez-García, E. (2018). SHOREX: A new tool for automatic and massive extraction of shorelines from Landsat and Sentinel 2 imagery. Obtenido de https://cgat.webs.upv.es/BigFiles/papers/SHOREX_Palomar_etal_2018.pdf

Palomar-Vázquez, J., Pardo-Pascual, J. E., Almonacid-Caballer, J., & Cabezas-Rabadán, C. (2023). Shoreline Analysis and Extraction Tool (SAET): A New Tool for the Automatic Extraction of Satellite-Derived Shorelines with Subpixel Accuracy. Obtenido de https://doi.org/10.3390/rs15123198

Pardo Pascual, J.E., Ruiz Fernández, L.A., Almonacid, J. y Calaf, X. (2008). Detección automática de cambios en la línea de costa a partir de imágenes de satélite de resolución media. Obtenido de https://cgat.webs.upv.es/BigFiles/114-Pardo%20Pascual.pdf

Puertos del Estado. (s.f.). Obtenido de https://www.puertos.es

Puertos del Estado. (18 de Noviembre de 2024). *Conjunto de datos: REDMAR.* Obtenido de https://bancodatos.puertos.es/BD/informes/INT_3.pdf

10. Anexo I

En la siguiente tabla se muestra una pequeña descripción de las principales funciones desarrolladas.

Módulo	Función	Descripción
ui.py	__init__	Inicializa la ventana principal del programa, carga los elementos definidos en el archivo del interfas, establece los valores iniciales de algunos atributos clave y conecta los distintos botones y componentes gráficos con sus respectivas funciones.
ui.py	ejecutar_visor	Permite abrir un archivo HTML en el navegador Firefox que muestra un mapa donde el usuario puede dibujar un rectángulo para obtener sus coordenadas.
ui.py	consultar	Recoge los parámetros del usuario y valida que el nivel de procesamiento esté seleccionado. Luego utiliza la función obtener_datos_sentinel para consultar las imágenes Sentinel-2 en función de los parámetros. Si encuentra resultados, los ordena y los muestra en una tabla. A continuación, calcula el centroide del área de interés con calcular_centroide_area_recorte. Empleando este centroide y los datos de los mareógrafos, llama a obtener_mareografos_mas cercanos, que luego se listan en la interfaz.
ui.py	ejecutar	Verifica que el usuario haya seleccionado un mareógrafo, una imagen y una carpeta de salida. Si falta alguno, muestra una advertencia. Con los datos seleccionados, calcula un rango de fechas alrededor de la imagen elegida y usa la función descargar_datos_portus para obtener datos de mareas desde Puertos del Estado, guardándolos en un archivo JSON. Luego, carga esos datos y llama a grafico_marea para crear un gráfico de mareas que guarda en formato PNG en la carpeta indicada. Finalmente, informa al usuario que la descarga y el proceso fueron exitosos.
ui.py	descargar_imagen	Descarga una imagen Sentinel-2 desde Google Earth Engine usando el ID, área de recorte, nivel y tres bandas seleccionadas por el usuario. Verifica que haya una imagen y bandas elegidas, llama a descargar_imagen_gee, espera 20 segundos y muestra un mensaje de confirmación.
ui.py	informes imágenes	Genera un informe CSV que asocia imágenes Sentinel-2 con los estados de marea de un mareógrafo cercano, usando las funciones descargar_datos_portus, informe_imagenes y guardar_informe_csv. Muestra advertencias si faltan datos clave.
ui.py	start_app	Lanza la interfaz gráfica de la aplicación, creando y mostrando la ventana principal y ejecutando el bucle de eventos de Qt.

funciones.py	obtener_datos_sentinel	Consulta imágenes Sentinel-2 en Google Earth Engine para un área, rango de fechas, nivel y porcentaje de nubosidad específicos. Filtra la colección según estos parámetros y, si encuentra imágenes, extrae sus metadatos relevantes (como fecha, hora, nubosidad y coordenadas) usando la función extraer_info_completa. Finalmente, devuelve estos datos organizados en un DataFrame.
funciones.py	obtener_mareografos_mas_cercanos	Recibe el centroide de la zona de recorte y un conjunto de mareógrafos en formato GeoJSON. Calcula la distancia geodésica (en kilómetros) entre el centroide y cada mareógrafo, guarda nombre, código y distancia, ordena los mareógrafos por distancia creciente y devolviendo los 5 más cercanos.
funciones.py	descargar_datos_portus	Utiliza Selenium para abrir la web de Puertos del Estado, seleccionar un mareógrafo, introducir fechas, generar un gráfico de datos de nivel del mar, capturar la respuesta JSON de la petición con esos datos, descomprimirla y guardarla en un archivo local. Luego cierra el navegador.
funciones.py	descargar_imagen_gee	Realiza la descarga de una imagen Sentinel-2 desde Google Earth Engine (GEE). Primero selecciona la colección según el nivel ("Nivel 1C" o "Nivel 2A"), recorta la imagen al área definida, selecciona las bandas especificadas proporcionadas, y luego crea y lanza una tarea de exportación para guardar la imagen en Google Drive.
funciones.py	grafico_marea	La función genera un gráfico de la marea con detección de picos, identifica intervalos de subida y bajada, marca el estado de la marea en una fecha dada, y guarda un informe CSV y un gráfico para una imagen y un mareógrafo seleccionado.
funciones.py	estado_marea	Función similar a grafico_marea pero para todas las imágenes mostradas tras la consulta. En este caso, la salia será un CSV únicamente.

TABLA 8: FUNCIONES PRINCIPALES. FUENTE: ELABORACIÓN PROPIA.

10.1. main.py

```
from ui import start_app

if __name__ == '__main__':
    start_app()
```

10.2. ui.py

```
import sys
import ast
from PyQt5 import QtWidgets, uic
from PyQt5.QtCore import Qt, QDate
from PyQt5.QtWidgets import QMessageBox, QFileDialog
import json
from datetime import datetime, timedelta
```

```
import ee
import time
import os

# Funciones propias
from constants import INTERFACE_UI
from constants import MAREOGRAFOS_GEOJSON, MAREOGRAFOS_WEBSCRAPING_JSON
from constants import SALIDA, AREA_RECORTE_JSON
from funciones import obtener_datos_sentinel
from funciones import calcular_centroide_area_recorte
from funciones import obtener_mareografos_mas_cercanos
from funciones import descargar_datos_portus
from funciones import descargar_imagen_gee
from funciones import visor
from funciones import mover_archivo
from funciones import grafico_marea
from funciones import informe_imagenes
from funciones import guardar_informe_csv

# Cargar interfaz .ui
form_class = uic.loadUiType(INTERFACE_UI)[0]

class MyDialogClass(QtWidgets.QDialog, form_class):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)

        # Inicializar atributos
        self.mareografos_cercanos = []
        self.mareografo_seleccionado = None
        self.imagen_seleccionada = None

        # Conexiones de botones y sliders
        self.slider_nubes.valueChanged.connect(self.desplazamiento_5)
        self.btn_visor.clicked.connect(self.ejecutar_visor)
        self.btn_consultar.clicked.connect(self.consultar)
        self.btn_limpiar.clicked.connect(self.limpiar)
        self.list_mareografos.itemClicked.connect(self.lista_mareografos)
        self.table_imagenes.itemClicked.connect(self.tabla_imagenes)
        self.btn_ejecutar.clicked.connect(self.ejecutar)
        self.btn_descarga_img.clicked.connect(self.descargar_imagen)
        self.btn_output.clicked.connect(self.salida)
        self.btn_informe.clicked.connect(self.informe_imagenes)


    def desplazamiento_5(self, valor):
        nuevo_valor = round(valor / 5) * 5
        if self.slider_nubes.value() != nuevo_valor:
            self.slider_nubes.setValue(nuevo_valor)
```

```
        else:
            self.porcentaje_nubes(nuevo_valor)


    def porcentaje_nubes(self, valor=None):
        if valor is None:
            valor = self.slider_nubes.value()
        self.txt_porcentaje.setText(str(valor))


    def ejecutar_visor(self):
        visor()

        max_espera = 300
        tiempo_esperado = 0
        intervalo = 1

        while not os.path.exists(AREA_RECORTE_JSON):
            if tiempo_esperado >= max_espera:
                QMessageBox.warning(self, "Advertencia", "No se ha generado el archivo de
área de recorte en el tiempo esperado.")
                return
            time.sleep(intervalo)
            tiempo_esperado += intervalo

        with open(AREA_RECORTE_JSON, 'r', encoding='utf-8') as f:
            area_recorte = json.load(f)
            texto_formateado = json.dumps(area_recorte, ensure_ascii=False)
            self.txt_area.setText(texto_formateado)

        mover_archivo(AREA_RECORTE_JSON, SALIDA)


    def consultar(self):
        fecha_inicio = self.txt_fechaini.date().toString("yyyy-MM-dd")
        fecha_fin = self.txt_fechafin.date().toString("yyyy-MM-dd")
        porcentaje_nubes = self.slider_nubes.value()
        area_recorte = ast.literal_eval(self.txt_area.text())
        nivel = self.comboBox_nivel.currentText()
        centroide_area_recorte = calcular_centroide_area_recorte(area_recorte)

        if nivel == " ":
            QMessageBox.warning(self, "Advertencia", "Debe seleccionar el nivel de
Sentinel-2.")
            return

        # 1. Datos GEE
        gee = obtener_datos_sentinel(fecha_inicio, fecha_fin, porcentaje_nubes,
area_recorte, nivel)
```

```
        if gee is None or gee.empty:
            QMessageBox.warning(self, "Advertencia", "No se encontraron imágenes para los
parámetros seleccionados.")
            return

        gee = gee.sort_values(by="fecha")
        self.table_imagenes.setRowCount(0)

        for i, row in gee.iterrows():
            self.table_imagenes.insertRow(i)
            self.table_imagenes.setItem(i, 0, QtWidgets.QTableWidgetItem(str(row["id"])))
            self.table_imagenes.setItem(i, 1, QtWidgets.QTableWidgetItem(row["fecha"]))
            self.table_imagenes.setItem(i, 2,
QtWidgets.QTableWidgetItem(str(row["hora"])))
            self.table_imagenes.setItem(i, 3,
QtWidgets.QTableWidgetItem(f"{row['nubosidad']:.2f} %"))
            self.table_imagenes.setItem(i, 4, QtWidgets.QTableWidgetItem(row["satelite"]))

            for col in range(2):
                self.table_imagenes.item(i, col).setData(Qt.UserRole, row.to_dict())

        self.table_imagenes.resizeColumnsToContents()

        # 2. Calcular mareógrafo más cercano
        with open(MAREOGRAFOS_GEOJSON, 'r', encoding='utf-8') as f:
            geojson_data = json.load(f)

        self.mareografos_cercanos =
obtener_mareografos_mas_cercanos(centroide_area_recorte, geojson_data)

        self.list_mareografos.clear()
        for mareografo in self.mareografos_cercanos:
            nombre = mareografo["nombre"]
            distancia = mareografo["distancia_km"]
            item_text = f"{nombre} - {distancia:.2f} km"

            item = QtWidgets.QListWidgetItem(item_text)
            item.setData(Qt.UserRole, mareografo)
            self.list_mareografos.addItem(item)

    def lista_mareografos(self, item):
        self.mareografo_seleccionado = item.data(Qt.UserRole)

    def tabla_imagenes(self, item):
        fila = item.row()
        self.imagen_seleccionada = self.table_imagenes.item(fila, 0).data(Qt.UserRole)

    def ejecutar(self):
```

```
        item_mareografo = self.list_mareografos.currentItem()
        item_imagen = self.table_imagenes.currentItem()
        path_output = self.txt_output.text()
        if item_mareografo is not None and item_imagen is not None and path_output != "":
            self.mareografo_seleccionado = item_mareografo.data(Qt.UserRole)
            fila = item_imagen.row()
            self.imagen_seleccionada = self.table_imagenes.item(fila, 0).data(Qt.UserRole)

            fecha_imagen = datetime.strptime(self.imagen_seleccionada["fecha"], "%Y-%m-
%d")
            fecha_hora_imagen = datetime.strptime(self.imagen_seleccionada["fecha"] + " "
+ self.imagen_seleccionada["hora"], "%Y-%m-%d %H:%M:%S")
            fecha_inicio = (fecha_imagen - timedelta(days=1)).strftime("%Y-%m-%d")
            fecha_fin = (fecha_imagen + timedelta(days=1)).strftime("%Y-%m-%d")
            salida = self.txt_output.text()
            nombre_json =
f"{self.mareografo_seleccionado['nombre']}_{fecha_inicio}_{fecha_fin}.json"
            nombre_png =
f"{self.mareografo_seleccionado['nombre']}_{fecha_inicio}_{fecha_fin}.png"

            # Web scraping de Puertos del Estado
            with open(MAREOGRAFOS_WEBSCRAPING_JSON, 'r', encoding='utf-8') as f:
                mareografos_ws_json = json.load(f)

            descargar_datos_portus(self.mareografo_seleccionado, mareografos_ws_json,
fecha_inicio, fecha_fin, salida, nombre_json)

            with open(os.path.join(salida, nombre_json), "r", encoding="utf-8") as f:
                respuesta_portus = json.load(f)

            grafico_marea(respuesta_portus, os.path.join(salida, nombre_png),
fecha_hora_imagen, self.mareografo_seleccionado['nombre'], salida)

            QMessageBox.information(self, "Información", "Datos descargados
correctamente.")

        errores = []

        if item_mareografo is None:
            errores.append("• No hay ningún mareógrafo seleccionado.")
        if item_imagen is None:
            errores.append("• No hay ninguna imagen seleccionada.")
        if path_output == "":
            errores.append("• No se ha seleccionado una carpeta de salida.")

        if errores:
            mensaje = "\n".join(errores)
            QMessageBox.warning(self, "Advertencia", mensaje)
            return
```

```
    def descargar_imagen(self):
        item_imagen = self.table_imagenes.currentItem()
        if item_imagen is not None:
            id_imagen = self.imagen_seleccionada['id']
            area_recorte = ast.literal_eval(self.txt_area.text())
            geometry = ee.Geometry.Polygon(area_recorte)
            nivel = self.comboBox_nivel.currentText()
            b1 = self.b1.currentText()
            b2 = self.b2.currentText()
            b3 = self.b3.currentText()

            if b1 == " " or b2 == " " or b3 == " ":
                QMessageBox.warning(self, "Advertencia", "Debe seleccionar al menos 3
bandas para la descarga.")
                return

            descargar_imagen_gee(id_imagen, geometry, nivel, bandas=[b1, b2, b3],
escala=10)
            time.sleep(20)
            QMessageBox.information(self, "Información", "Imagen descargada
correctamente.")
        else:
            QMessageBox.warning(self, "Advertencia", "No hay ninguna imagen
seleccionada.")
            return


    def obtener_lista_imagenes_y_fechas(self):
        lista = []
        fechas = []
        filas = self.table_imagenes.rowCount()
        for i in range(filas):
            id_item = self.table_imagenes.item(i, 0)
            fecha_item = self.table_imagenes.item(i, 1)
            hora_item = self.table_imagenes.item(i, 2)
            nubosidad_item = self.table_imagenes.item(i, 3)
            satelite_item = self.table_imagenes.item(i, 4)

            if id_item and fecha_item and hora_item:
                fecha_str = fecha_item.text()
                hora_str = hora_item.text()
                try:
                    fecha_dt = datetime.strptime(fecha_str, "%Y-%m-%d")
                    fechas.append(fecha_dt)
                except ValueError:
                    continue
```

```
                lista.append({
                    "id": id_item.text(),
                    "fecha": fecha_str,
                    "hora": hora_str,
                    "nubosidad": nubosidad_item.text(),
                    "satelite": satelite_item.text(),
                })

        if not fechas:
            QMessageBox.warning(self, "Advertencia", "No hay imágenes en la tabla o las
fechas no son válidas.")
            return lista, None, None

        primera_fecha = min(fechas)
        ultima_fecha = max(fechas)
        fecha_inicio = (primera_fecha - timedelta(days=1)).strftime("%Y-%m-%d")
        fecha_fin = (ultima_fecha + timedelta(days=1)).strftime("%Y-%m-%d")

        return lista, fecha_inicio, fecha_fin


    def informe_imagenes(self):
        item_mareografo = self.list_mareografos.currentItem()
        path_output = self.txt_output.text()
        lista_imagenes, fecha_inicio, fecha_fin = self.obtener_lista_imagenes_y_fechas()

        if item_mareografo is not None and path_output != "":
            with open(MAREOGRAFOS_WEBSCRAPING_JSON, 'r', encoding='utf-8') as f:
                mareografos_ws_json = json.load(f)

            salida = self.txt_output.text()
            nombre_json =
f"{self.mareografo_seleccionado['nombre']}_{fecha_inicio}_{fecha_fin}.json"
            descargar_datos_portus(self.mareografo_seleccionado, mareografos_ws_json,
fecha_inicio, fecha_fin, salida, nombre_json)

            with open(os.path.join(salida, nombre_json), "r", encoding="utf-8") as f:
                respuesta_portus = json.load(f)

            estados_marea = informe_imagenes(lista_imagenes, respuesta_portus)
            ruta_csv = os.path.join(salida,
f"{self.mareografo_seleccionado['nombre']}_informe_global_{fecha_inicio}_{fecha_fin}.csv")

            with open(ruta_csv, mode='w', newline='', encoding='utf-8') as f:
                guardar_informe_csv(lista_imagenes, estados_marea, ruta_csv)

            time.sleep(5)
            QMessageBox.information(self, "Información", "Informe global generado
correctamente.")
```

```
        else:
            errores = []

            if item_mareografo is None:
                errores.append("• No hay ningún mareógrafo seleccionado.")
            if path_output == "":
                errores.append("• No se ha seleccionado una carpeta de salida.")

            if errores:
                mensaje = "\n".join(errores)
                QMessageBox.warning(self, "Advertencia", mensaje)
                return


    def limpiar(self):
        # Limpiar campos de la interfaz
        self.txt_fechaini.setDate(QDate(2000, 1, 1))
        self.txt_fechafin.setDate(QDate(2000, 1, 1))
        self.slider_nubes.setValue(0)
        self.txt_porcentaje.setText("0")
        self.txt_area.clear()
        self.list_mareografos.clear()
        self.table_imagenes.setRowCount(0)

        # Reiniciar atributos internos
        self.mareografos_cercanos = []
        self.mareografo_seleccionado = None
        self.imagen_seleccionada = None


    def salida(self):
        carpeta = QFileDialog.getExistingDirectory(
        None, 'Selecciona la carpeta de salida'
        )
        if carpeta:
            self.txt_output.setText(carpeta)


def start_app():
    app = QtWidgets.QApplication(sys.argv)
    myDialog = MyDialogClass(None)
    myDialog.show()
    app.exec_()
```

10.3. funciones.py

```
import ee
import pandas as pd
```

```
from geopy.distance import geodesic
from seleniumwire import webdriver
from selenium.webdriver.firefox.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
import time
import gzip
import io
import json
from datetime import datetime
import numpy as np
from scipy.signal import find_peaks
import os
import shutil
import webbrowser
import tempfile
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import csv

# Funciones propias
from constants import GECKODRIVER_PATH
from constants import WEB_PORTUS_URL
from constants import BTT_HISTORICO_NIVEL_MAR
from constants import BTT_GRAFICO_SERIES_TEMPORALES
from constants import BTT_FECHA_INICIO, BTT_FECHA_FIN
from constants import BTT_TIPO_GRAFICO
from constants import BTT_DATOS_AL_MINUTO
from constants import BTT_GENERAR_GRAFICO
from constants import HTML

def extraer_info_completa(imagen):
    id = imagen.id()
    nombre_producto = imagen.get('PRODUCT_ID')
    fecha_completa = imagen.date()
    fecha_formateada = fecha_completa.format('YYYY-MM-dd')
    hora_formateada = fecha_completa.format('HH:mm:ss')
    nubosidad = imagen.get('CLOUDY_PIXEL_PERCENTAGE')
    satelite = imagen.get('SPACECRAFT_NAME')
    bbox = imagen.geometry().bounds(1)
    centroide = bbox.centroid(1)

    # Coordenadas
    coord_centroide = centroide.coordinates()
    coords_bbox = ee.List(bbox.coordinates().get(0))
    esquina_1 = ee.List(coords_bbox.get(0))
    esquina_2 = ee.List(coords_bbox.get(1))
```

```
    esquina_3 = ee.List(coords_bbox.get(2))
    esquina_4 = ee.List(coords_bbox.get(3))

    return ee.Feature(None, {
        'id': id,
        'nombre_producto': nombre_producto,
        'fecha': fecha_formateada,
        'hora': hora_formateada,
        'nubosidad': nubosidad,
        'satelite': satelite,
        'lon_centroide': coord_centroide.get(0),
        'lat_centroide': coord_centroide.get(1),
        'esq1_lon': esquina_1.get(0),
        'esq1_lat': esquina_1.get(1),
        'esq2_lon': esquina_2.get(0),
        'esq2_lat': esquina_2.get(1),
        'esq3_lon': esquina_3.get(0),
        'esq3_lat': esquina_3.get(1),
        'esq4_lon': esquina_4.get(0),
        'esq4_lat': esquina_4.get(1)
    })


def obtener_datos_sentinel(fecha_inicio, fecha_fin, porcentaje_nubes, area_recorte,
nivel):
    ee.Initialize(project='ee-noelieta19')

    if nivel == "Nivel 1C":
        nivel_S2 = "COPERNICUS/S2_HARMONIZED/"
    elif nivel == "Nivel 2A":
        nivel_S2 = "COPERNICUS/S2_SR_HARMONIZED/"

    coleccion = ee.ImageCollection(nivel_S2) \
        .filterDate(fecha_inicio, fecha_fin) \
        .filterBounds(ee.Geometry.Polygon([area_recorte])) \
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', porcentaje_nubes))

    num_images = coleccion.size().getInfo()

    if num_images == 0:
        return None

    tabla = coleccion.map(extraer_info_completa)
    lista = tabla.getInfo()

    datos = []
    for feature in lista['features']:
        propiedades = feature['properties']
        datos.append({
```

```
            'id': propiedades['id'],
            'nombre_producto': propiedades['nombre_producto'],
            'fecha': propiedades['fecha'],
            'hora': propiedades['hora'],
            'nubosidad': propiedades['nubosidad'],
            'satelite': propiedades['satelite'],
            'lat_centroide': propiedades['lat_centroide'],
            'lon_centroide': propiedades['lon_centroide'],
            'esq1_lat': propiedades['esq1_lat'],
            'esq1_lon': propiedades['esq1_lon'],
            'esq2_lat': propiedades['esq2_lat'],
            'esq2_lon': propiedades['esq2_lon'],
            'esq3_lat': propiedades['esq3_lat'],
            'esq3_lon': propiedades['esq3_lon'],
            'esq4_lat': propiedades['esq4_lat'],
            'esq4_lon': propiedades['esq4_lon']
        })

    df = pd.DataFrame(datos)
    return df


def calcular_centroide_area_recorte(area_recorte):
    # Extraer x (lon) e y (lat) por separado
    xs = [p[0] for p in area_recorte]
    ys = [p[1] for p in area_recorte]

    centroide_area_recorte = [
        (min(xs) + max(xs)) / 2,
        (min(ys) + max(ys)) / 2
    ]
    return centroide_area_recorte


def obtener_mareografos_mas_cercanos(centroide_area_recorte, mareografos_geojson):
    distancias = []

    # Calcular distancias
    for feature in mareografos_geojson['features']:
        coordenadas = feature['geometry']['coordinates']
        coord_mareografo = (coordenadas[0], coordenadas[1])  # lat, lon
        distancia = geodesic(centroide_area_recorte, coord_mareografo).kilometers
        distancias.append({
            'nombre': feature['properties']['nombre'],
            'codigo_bd': feature['properties']['codigo_bd'],
            'distancia_km': distancia
        })

    distancias_ordenadas = sorted(distancias, key=lambda x: x['distancia_km'])
```

```
    return distancias_ordenadas[:5]


def descargar_datos_portus(mareografo_mas_cercano, mareografos_ws_json, fecha_inicio_ui,
fecha_fin_ui, salida, nombre_archivo):
    service = Service(GECKODRIVER_PATH)
    driver = webdriver.Firefox(service=service)
    driver.maximize_window()

    driver.get(WEB_PORTUS_URL)
    time.sleep(5)
    wait = WebDriverWait(driver, 10)

    # Mareógrafo más cercano al centroide del área de recorte
    nombre_mareografo = mareografo_mas_cercano['nombre']
    mareografos_dict = {m["nombre"]: m for m in mareografos_ws_json}

    wait.until(EC.element_to_be_clickable((By.XPATH, BTT_HISTORICO_NIVEL_MAR))).click()
    time.sleep(5)

    wait.until(EC.element_to_be_clickable((By.XPATH,
mareografos_dict[nombre_mareografo]["xpath"]))).click()
    wait.until(EC.element_to_be_clickable((By.XPATH,
BTT_GRAFICO_SERIES_TEMPORALES))).click()
    time.sleep(5)

    # Fechas
    fecha_inicio = datetime.strptime(fecha_inicio_ui, "%Y-%m-%d").strftime("%d/%m/%Y")
    fecha_fin = datetime.strptime(fecha_fin_ui, "%Y-%m-%d").strftime("%d/%m/%Y")

    # Introducir fechas
    input_fecha_ini = wait.until(EC.presence_of_element_located((By.XPATH,
BTT_FECHA_INICIO)))
    input_fecha_ini.send_keys(Keys.CONTROL, 'a')
    input_fecha_ini.send_keys(Keys.DELETE)
    input_fecha_ini.send_keys(fecha_inicio)
    input_fecha_ini.send_keys(Keys.TAB)

    input_fecha_fin = wait.until(EC.presence_of_element_located((By.XPATH,
BTT_FECHA_FIN)))
    input_fecha_fin.send_keys(Keys.CONTROL, 'a')
    input_fecha_fin.send_keys(Keys.DELETE)
    input_fecha_fin.send_keys(fecha_fin)
    input_fecha_fin.send_keys(Keys.TAB)

    # Gráfico de series temporales
    wait.until(EC.element_to_be_clickable((By.XPATH, BTT_TIPO_GRAFICO))).click()
    wait.until(EC.element_to_be_clickable((By.XPATH, BTT_DATOS_AL_MINUTO))).click()
```

```
    wait.until(EC.element_to_be_clickable((By.XPATH, BTT_GENERAR_GRAFICO))).click()
    time.sleep(5)

    # Extraer datos del gráfico
    post_requests = [r for r in driver.requests if r.method == 'POST' and
'/historicosSerialTime/estacion/SEA_LEVEL' in r.url]

    if post_requests:
        req = post_requests[-1]
        try:
            with gzip.GzipFile(fileobj=io.BytesIO(req.response.body)) as gz:
                descomprimido = gz.read()
                respuesta = json.loads(descomprimido.decode('utf-8'))

                os.makedirs(salida, exist_ok=True)
                ruta_json = os.path.join(salida, nombre_archivo)

                with open(ruta_json, "w", encoding="utf-8") as f_json:
                    json.dump(respuesta, f_json, ensure_ascii=False, indent=2)

        except Exception as e:
            print("Error al descomprimir o decodificar JSON:", e)
    else:
        print("No se encontró la petición POST esperada.")
    driver.quit()


def descargar_imagen_gee(id_imagen, geometry, nivel, bandas=None, escala=10):
    if nivel == "Nivel 1C":
        prefijo = "COPERNICUS/S2_HARMONIZED"
    elif nivel == "Nivel 2A":
        prefijo = "COPERNICUS/S2_SR_HARMONIZED"

    imagen = prefijo + id_imagen
    imagen = ee.Image(imagen).clip(geometry)
    if bandas:
        imagen = imagen.select(bandas)

    n_imagen = id_imagen + "-" + nivel.replace(" ", "") + "-" + "_".join(bandas)

    task = ee.batch.Export.image.toDrive(
        image=imagen,
        folder='TFM',
        fileNamePrefix=n_imagen,
        scale=escala,
        region=geometry,
        crs='EPSG:25830',
        maxPixels=1e13
    )
```

```
    try:
        task.start()
    except Exception as e:
        print(f"Error al iniciar la tarea de exportación: {e}")
        return


def visor(html_path = HTML):
    with open(html_path, "r", encoding="utf-8") as file:
        html_content = file.read()

    with tempfile.NamedTemporaryFile("w", delete=False, suffix=".html") as f:
        f.write(html_content)
        temp_path = f.name

    browser = webbrowser.get('"C:\\Program Files\\Mozilla Firefox\\firefox.exe" %s')
    browser.open(f"file://{temp_path}")


def mover_archivo(ruta_origen, ruta_destino):
    if not os.path.isfile(ruta_origen):
        print(f"El archivo no existe en la ruta: {ruta_origen}")
        return False

    if os.path.exists(ruta_destino):
        try:
            os.remove(os.path.join(ruta_destino, "coordenadas_rectangulo.json"))
        except PermissionError as e:
            print(f"No se pudo eliminar el archivo destino: {e}")
            return False

    shutil.move(ruta_origen, ruta_destino)
    return True


def grafico_marea(respuesta_portus, png, fecha, nombre_mareografo, salida):
    fechas = [datetime.strptime(d["fecha"], "%Y-%m-%d %H:%M:%S.%f") for d in
respuesta_portus["datos"]]
    valores = [float(d["Datos al minuto"]) for d in respuesta_portus["datos"]]
    valores_np = np.array(valores)

    # Detectar picos máximos y mínimos
    picos_max, _ = find_peaks(valores_np, distance=60, prominence=0.1)
    picos_min, _ = find_peaks(-valores_np, distance=60, prominence=0.1)

    # Calcular valor más cercano a la fecha proporcionada
    diffs = [abs((f - fecha).total_seconds()) for f in fechas]
    idx_marcado = diffs.index(min(diffs))
```

```
    valor_marcado = valores[idx_marcado]

    plt.figure(figsize=(16, 8))
    plt.plot(fechas, valores, linestyle='-', color='blue', linewidth=0.8, label='Altura de
la marea')
    plt.plot([fechas[i] for i in picos_max], [valores[i] for i in picos_max], 'ro',
label='Picos máximos')
    plt.plot([fechas[i] for i in picos_min], [valores[i] for i in picos_min], 'go',
label='Picos mínimos')
    plt.plot(fecha, valor_marcado, 'ms', label='Instante toma imagen')

    ax = plt.gca()
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%d-%m-%Y %H:%M'))
    ax.xaxis.set_major_locator(mdates.HourLocator(byhour=range(0, 24, 2)))

    plt.title(f"Altura de la marea y detección de picos del mareógrafo
{nombre_mareografo}", fontsize=16)
    plt.xlabel("Fecha y hora", fontsize=12)
    plt.ylabel("Altura (m)", fontsize=12)
    plt.grid(True)
    plt.xticks(rotation=45, ha='right')
    plt.legend()
    plt.tight_layout()
    plt.subplots_adjust(bottom=0.2)
    plt.savefig(png, dpi=600)
    plt.close()

    picos = [(i, "max") for i in picos_max] + [(i, "min") for i in picos_min]
    picos.sort(key=lambda x: x[0])
    intervalos = []

    if picos:
        # Primer tramo
        i0 = 0
        i1, tipo1 = picos[0]
        sentido_inicial = "Sube la marea" if tipo1 == "max" else "Baja la marea"
        intervalos.append({
            "inicio": fechas[i0],
            "fin": fechas[i1],
            "desde": f"{fechas[i0]} - {valores[i0]:.3f} m",
            "hasta": f"{fechas[i1]} - {valores[i1]:.3f} m",
            "cambio": sentido_inicial + " (tramo inicial)"
        })

        # Tramos intermedios
        for i in range(len(picos) - 1):
            i1, tipo1 = picos[i]
            i2, tipo2 = picos[i + 1]
            if tipo1 == "min" and tipo2 == "max":
```

```
                sentido = "Sube la marea"
            elif tipo1 == "max" and tipo2 == "min":
                sentido = "Baja la marea"
            else:
                continue

            intervalos.append({
                "inicio": fechas[i1],
                "fin": fechas[i2],
                "desde": f"{fechas[i1]} - {valores[i1]:.3f} m",
                "hasta": f"{fechas[i2]} - {valores[i2]:.3f} m",
                "cambio": sentido
            })

        # Último tramo
        i_last_pico, tipo_last = picos[-1]
        i_final = len(fechas) - 1
        sentido_final = "Sube la marea" if tipo_last == "min" else "Baja la marea"
        intervalos.append({
            "inicio": fechas[i_last_pico],
            "fin": fechas[i_final],
            "desde": f"{fechas[i_last_pico]} - {valores[i_last_pico]:.3f} m",
            "hasta": f"{fechas[i_final]} - {valores[i_final]:.3f} m",
            "cambio": sentido_final + " (tramo final)"
        })

    # Determinar el estado de la marea en el instante de toma de imagen
    estado_marea_actual = "Desconocido"
    for tramo in intervalos:
        if tramo["inicio"] <= fecha <= tramo["fin"]:
            estado_marea_actual = tramo["cambio"]
            break

    # Guardar informe en CSV
    fecha_inicio = fechas[0].strftime('%Y-%m-%d %H:%M:%S')
    fecha_fin = fechas[-1].strftime('%Y-%m-%d %H:%M:%S')
    csv_filename = os.path.join(salida, f"{nombre_mareografo}_{fechas[0].date()}_{fechas[-
1].date()}_informe_marea.csv")

    with open(csv_filename, mode='w', newline='', encoding='utf-8') as f:
        f.write(f"INFORME DE INTERVALOS DE MAREA\n")
        f.write(f"Mareógrafo: {nombre_mareografo}\n")
        f.write(f"Fecha de inicio del análisis: {fecha_inicio}\n")
        f.write(f"Fecha de fin del análisis: {fecha_fin}\n")
        f.write(f"Instante de toma de imagen: {fecha.strftime('%Y-%m-%d %H:%M:%S')}\n")
        f.write(f"Estado de la marea en el instante de toma de imagen:
{estado_marea_actual}\n")

        f.write("\nDETALLE DE LOS TRAMOS DETECTADOS:\n\n")
```

```
        for idx, tramo in enumerate(intervalos, 1):
            f.write(f"Tramo {idx}:\n")
            f.write(f"  - Inicio: {tramo['inicio'].strftime('%Y-%m-%d %H:%M:%S')}\n")
            f.write(f"  - Fin: {tramo['fin'].strftime('%Y-%m-%d %H:%M:%S')}\n")
            f.write(f"  - Desde: {tramo['desde']}\n")
            f.write(f"  - Hasta: {tramo['hasta']}\n")
            f.write(f"  - Cambio de marea: {tramo['cambio']}\n")


def estado_marea(respuesta_portus, lista_fechas):
    fechas = [datetime.strptime(d["fecha"], "%Y-%m-%d %H:%M:%S.%f") for d in
respuesta_portus["datos"]]
    valores = [float(d["Datos al minuto"]) for d in respuesta_portus["datos"]]
    valores_np = np.array(valores)

    picos_max, _ = find_peaks(valores_np, distance=60, prominence=0.1)
    picos_min, _ = find_peaks(-valores_np, distance=60, prominence=0.1)

    picos = [(i, "max") for i in picos_max] + [(i, "min") for i in picos_min]
    picos.sort(key=lambda x: x[0])

    intervalos = []
    if picos:
        # Intervalo inicial
        intervalos.append({
            "inicio": fechas[0],
            "fin": fechas[picos[0][0]],
            "estado": "Sube la marea" if picos[0][1] == "max" else "Baja la marea"
        })
        # Intervalos intermedios
        for i in range(len(picos) - 1):
            i1, t1 = picos[i]
            i2, t2 = picos[i + 1]
            estado = "Sube la marea" if t1 == "min" and t2 == "max" else "Baja la marea"
            intervalos.append({
                "inicio": fechas[i1],
                "fin": fechas[i2],
                "estado": estado
            })
        # Intervalo final
        intervalos.append({
            "inicio": fechas[picos[-1][0]],
            "fin": fechas[-1],
            "estado": "Sube la marea" if picos[-1][1] == "min" else "Baja la marea"
        })
    else:
        intervalos.append({
            "inicio": fechas[0],
            "fin": fechas[-1],
```

```
            "estado": "Desconocido"
        })

    estados_por_fecha = []
    for fecha_img in lista_fechas:
        estado_actual = "Desconocido"
        for tramo in intervalos:
            if tramo["inicio"] <= fecha_img <= tramo["fin"]:
                estado_actual = tramo["estado"]
                break
        estados_por_fecha.append(estado_actual)
    return estados_por_fecha


def informe_imagenes(lista_imagenes, respuesta_portus):
    lista_fechas = [datetime.strptime(imagen['fecha'] + " " + imagen['hora'], "%Y-%m-%d
%H:%M:%S") for imagen in lista_imagenes]
    estados_marea = estado_marea(respuesta_portus, lista_fechas)

    informe = []
    for imagen, estado in zip(lista_imagenes, estados_marea):
        informe.append({
            "id": imagen["id"],
            "fecha_hora": imagen['fecha'] + " " + imagen['hora'],
            "estado_marea": estado
        })
    return informe


def guardar_informe_csv(lista_imagenes, estados_marea, ruta_csv):
    with open(ruta_csv, mode='w', newline='', encoding='utf-8') as archivo:
        writer = csv.writer(archivo)
        writer.writerow(['ID', 'Fecha', 'Hora', 'Nubosidad', 'Satelite', 'Estado Marea'])
        for img, estado in zip(lista_imagenes, estados_marea):
            writer.writerow([img['id'], img['fecha'], img['hora'], img['nubosidad'],
img['satelite'], estado['estado_marea']])
```

10.4. constants.py

```
# Rutas para archivos de datos
INTERFACE_UI = r"C:\Users\noeli\UNIVERSIDAD\TFM\Codigo\Interfaz_TFM.ui"
MAREOGRAFOS_GEOJSON = r"C:\Users\noeli\TFM\Codigo\data\mareografos.geojson"
MAREOGRAFOS_WEBSCRAPING_JSON =
r"C:\Users\noeli\TFM\Codigo\data\mareografos_webscraping.json"
SALIDA = r"C:\Users\noeli\TFM\Codigo\output"
HTML = r"C:\Users\noeli\TFM\Codigo\mapa_leaflet.html"
AREA_RECORTE_JSON = r"C:\Users\noeli\Downloads\coordenadas_rectangulo.json"
```

```
# Ruta al driver para Selenium
GECKODRIVER_PATH = r"C:\Users\noeli\OneDrive\Documentos\00\web_scraping\geckodriver.exe"

# Web scraping
WEB_PORTUS_URL = r"https://portus.puertos.es/#/"
BTT_HISTORICO_NIVEL_MAR =
'/html/body/div/div/div[2]/div/div[3]/div/div/div/div/div/div[2]/label'
BTT_GRAFICO_SERIES_TEMPORALES =
'/html/body/div[2]/div/div/div[2]/div[2]/div[1]/div/div[2]/div/div/div/div[2]/div[8]/label
/span[1]'
BTT_FECHA_INICIO = "//input[contains(@id, '_dateFrom') and @type='text']"
BTT_FECHA_FIN = "//input[contains(@id, '_dateTo') and @type='text']"
BTT_TIPO_GRAFICO = '//div[contains(@class, "dx-texteditor-container") and contains(@class,
"dx-tag-container")]'
BTT_DATOS_AL_MINUTO = '//div[contains(text(), "Datos al minuto")]'
BTT_GENERAR_GRAFICO = '//span[text()="GENERAR GRÁFICO"]/ancestor::div[contains(@class,
"dx-button-content")]'
```

10.5. mapa_leaflet.html

```
<!DOCTYPE html>
<html>
<head>
  <title>TFM Noelia Soriano</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!-- Leaflet -->
  <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
  <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>

  <!-- Leaflet Draw -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet-
draw@1.0.4/dist/leaflet.draw.css" />
  <script src="https://cdn.jsdelivr.net/npm/leaflet-
draw@1.0.4/dist/leaflet.draw.js"></script>

  <style>
    html, body { height: 100%; margin: 0; }
    #map { height: 90%; width: 100%; }
    #boton { height: 10%; text-align: center; padding: 10px; }
  </style>
</head>
<body>
  <div id="map"></div>
  <div id="boton">
    <button onclick="mostrarCoordenadas()">Obtener coordenadas</button>
  </div>
```

```
<script>
  const map = L.map('map').setView([40.4168, -3.7038], 6);

  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 18,
  }).addTo(map);

  let drawnItems = new L.FeatureGroup();
  map.addLayer(drawnItems);

  const drawControl = new L.Control.Draw({
    draw: {
      polygon: false,
      polyline: false,
      circle: false,
      marker: false,
      circlemarker: false,
      rectangle: true
    },
    edit: {
      featureGroup: drawnItems
    }
  });
  map.addControl(drawControl);

  let rectCoords = null;

  map.on(L.Draw.Event.CREATED, function (event) {
    drawnItems.clearLayers();
    const layer = event.layer;
    drawnItems.addLayer(layer);
    rectCoords = layer.getBounds();
  });

  function downloadJSON(data, filename) {
    const mime = "data:application/json;charset=utf-8,";
    const file = encodeURIComponent(JSON.stringify(data, null, 2));
    const a = document.createElement("a");
    a.setAttribute("href", mime + file);
    a.setAttribute("download", filename);
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
  }

  function mostrarCoordenadas() {
    if (rectCoords) {
      const sw = rectCoords.getSouthWest();
      const ne = rectCoords.getNorthEast();
```

```
        const nw = L.latLng(ne.lat, sw.lng);
        const se = L.latLng(sw.lat, ne.lng);

        const coordenadas = [
          [sw.lng, sw.lat],
          [se.lng, se.lat],
          [ne.lng, ne.lat],
          [nw.lng, nw.lat],
          [sw.lng, sw.lat]
        ];

        console.log(JSON.stringify(coordenadas));
        downloadJSON(coordenadas, "coordenadas_rectangulo.json");
      } else {
        alert("Dibuja primero un rectángulo.");
      }
    }
  </script>
</body>
</html>
```

11. Anexo II

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.				X
ODS 4. Educación de calidad.			X	
ODS 5. Igualdad de género.				X
ODS 6. Agua limpia y saneamiento.		X		
ODS 7. Energía asequible y no contaminante.			X	
ODS 8. Trabajo decente y crecimiento económico.			X	
ODS 9. Industria, innovación e infraestructuras.	X			
ODS 10. Reducción de las desigualdades.				X
ODS 11. Ciudades y comunidades sostenibles.	X			
ODS 12. Producción y consumo responsables.		X		
ODS 13. Acción por el clima.	X			
ODS 14. Vida submarina.	X			
ODS 15. Vida de ecosistemas terrestres.		X		
ODS 16. Paz, justicia e instituciones sólidas.				X
ODS 17. Alianzas para lograr objetivos		X		

TABLA 9: GRADO DE RELACIÓN DEL TRABAJO CON LOS ODS. FUENTE: ELABORACIÓN PROPIA.

Descripción de la alineación del TFM con los ODS con un grado de relación más alto:

Como se puede observar en la tabla anterior, los ODS con un mayor grado de relación son “ODS 9 Industria, innovación e infraestructuras”, “ODS 11 Ciudades y comunidades sostenibles”, “ODS 13 Acción por el clima” y “ODS 14 Vida submarina”. Todos ellos impulsan el desarrollo y uso de soluciones tecnológicas innovadoras que favorecen una gestión costera más precisa y sostenible, a través de la integración de información mareográfica en los procesos de extracción de línea de costa a partir de imágenes satelitales.

12. Cartografía

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA GEODÉSICA CARTOGRÁFICA Y TOPOGRÁFICA

Título:
Distribución de mareógrafos REDMAR

Autora: Noelia Soriano Dolz

Escala: E: 1/7.000.000

Sistema de coordenadas: ETRS89 - UTM 30N

Fecha: Julio 2025