Plataforma Inteligente de Destinos (PID): resumen y acceso a datos

11/09/2025

Contents

1 Introducción global

Resumen de acceso a datos a través de la **Plataforma Inteligente de Destinos (PID)** impulsada por la Secretaría de Estado de Turismo (SEGITTUR).

Veremos :

- Un **resumen** claro del proyecto PID.
- **Vías de acceso a datos** para investigación (DATAESTUR, SIT y PIA).
- Una **lista de enlaces directos** a los recursos clave.
- Un **script en R** listo para adaptar y consumir la API de DATAESTUR.

2 Resumen del Proyecto PID

La **Plataforma Inteligente de Destinos (PID)** es una iniciativa de la Secretaría de Estado de Turismo, operada por **SEGITTUR** y financiada con fondos **NextGenerationEU** (PRTR, Componente 14). Su objetivo es **digitalizar y mejorar la gestión de destinos** y empresas turísticas mediante servicios y módulos interoperables a lo largo del ciclo del viaje y de la gestión del destino.

Bloques y módulos destacados (no exhaustivo): - **SIT – Sistema de Inteligencia Turística:** integración de múltiples fuentes para análisis, cuadros de mando e informes. - **Gestión avanzada de datos y cuadros de mando** para destinos y empresas (incluye reputación online y otros servicios). - **Aplicación turística estatal** (servicios al visitante). - **Innovación abierta (PIA):** *Sandbox* para pruebas, **Espacio de Datos** para compartición segura y **Marketplace** para soluciones/algoritmos.

Despliegue y elegibilidad:
La incorporación de destinos a la PID se articula por fases y exige pertenecer a la **Red DTI** (Destinos Turísticos Inteligentes).

3 Acceso a datos para investigación

Existen tres vías principales, según necesidades de granularidad, licencia y experimentación:

3.1 1) Datos abiertos — DATAESTUR (acceso inmediato)

Portal público con series e indicadores turísticos (por ejemplo, FRONTUR, EGATUR, ocupación, transporte, sostenibilidad).
- **API** para automatizar descargas: desde la página de cada dataset, pulsa **"Pruébalo / Execute"**, copia la URL con parámetros y úsala en tu script. - **Calendario de datos** y **buenas prácticas** disponibles para planificar pipelines y uso eficiente.

Cuándo usarlo: análisis generales, series temporales, automatización reproducible.

3.2 2) Datos específicos para destinos — SIT (acceso con acuerdo)

El **SIT** ofrece piezas públicas (divulgadas también a través de DATAESTUR) y **componentes restringidos** adaptados a destinos, con cuadros de mando y analítica avanzada.
- Para investigación con **granularidad** o datos no abiertos, suele requerirse **acuerdo con SEGITTUR** y/o con el **destino** participante.

Cuándo usarlo: si necesitas detalle operativo del destino, cruces avanzados o paneles específicos.

3.3 3) Experimentación / desarrollo — PIA (Sandbox + Espacio de Datos)

La **Plataforma para la Innovación Abierta (PIA)** facilita: - **Sandbox**: entorno de pruebas en la nube con datos reales/realistas. - **Espacio de Datos de Turismo**: compartición segura y trazable de datos públicos/privados. - **Marketplace**: catálogo de soluciones y componentes.

Acceso: mediante **formulario de solicitud**, describiendo tu proyecto académico, necesidades de datos y finalidad.

3.4 Ruta práctica recomendada

1. **Arranca con DATAESTUR**: identifica series relevantes y crea llamadas API reproducibles.
2. **Ajusta la pregunta de investigación** y lista variables que no aparecen en datos abiertos.
3. **Si necesitas granularidad o datos no abiertos**, contacta con **SEGITTUR/SIT** (o un destino de la Red DTI).
4. **Si vas a prototipar modelos**, solicita acceso a **PIA** (Sandbox + Espacio de Datos) vía formulario.

4 Enlaces de acceso a datos

4.1 DATAESTUR (portal y utilidades)

- Portal principal: https://www.dataestur.es/
- API (guía y endpoints): https://www.dataestur.es/en/apidata/
- Calendario de actualización de datos: https://www.dataestur.es/en/data-calendar/
- Mapa del sitio (índice de fuentes): https://www.dataestur.es/en/site-map/
- Buenas prácticas de uso de la API (PDF): https://www.dataestur.es/wp-content/uploads/Buenas-practicas-uso-API-Dataestur.pdf

4.2 SIT – Sistema de Inteligencia Turística

- Información del proyecto SIT: https://www.segittur.es/transformacion-digital/proyectos-transformacion-digital/sistema-de-inteligencia-turistica-2/
- Portal SIT (acceso): https://sistemainteligenciaturistica.es/
- Registro/solicitud de usuario: https://sistemainteligenciaturistica.es/registro-sit/

4.3 PIA – Plataforma de Innovación Abierta

- Página general (visión, Sandbox, Espacio de Datos, Marketplace): https://www.segittur.es/plataforma-inteligente/proyectos-plataforma-inteligente/plataforma-de-innovacion-abierta-pia/
- Solicitud de información/acceso (formulario): https://www.segittur.es/plataforma-del-ecosistema-para-la-innovacion-abierta-pia-de-segittur/
- Espacio de Datos de Turismo (descripción): https://www.segittur.es/plataforma-inteligente/proyectos-plataforma-inteligente/espacio-de-datos-de-turismo/

5 Script en R – API de DATAESTUR

A continuación tienes un script **base** en R para consumir la **API de DATAESTUR**.
Uso recomendado: ve a la página **API** de DATAESTUR, elige un dataset, pulsa **"Pruébalo / Execute"**, copia la **URL con parámetros** y pégala en `API_URL`.

```r
# install.packages(c("httr2","jsonlite","dplyr","janitor","readr","lubridate"))
library(httr2)
library(jsonlite)
```

```r
library(dplyr)
library(janitor)
library(readr)
library(lubridate)

# --- CONFIGURACIÓN ------------------------------------------------------------

# 1) En DATAESTUR > API, elige dataset y pulsa "Pruébalo/Execute".
# 2) Copia la URL completa (con parámetros) y pégala aquí:
API_URL <- "PEGA_AQUI_LA_URL_DE_DATAESTUR"

# (Opcional) Si la API requiriera clave:
API_KEY <- Sys.getenv("DATAESTUR_API_KEY")  # o pon la clave literal entre comillas
USE_API_KEY <- nchar(API_KEY) > 0

# Carpeta y nombre del CSV de salida
OUT_DIR  <- "data"
OUT_FILE <- file.path(OUT_DIR, "dataestur_export.csv")

# --- FUNCIÓN AUXILIAR ---------------------------------------------------------

fetch_dataestur <- function(url, api_key = NULL) {
  req <- request(url) |>
    req_user_agent("R (httr2) - research script")

  if (!is.null(api_key) && nzchar(api_key)) {
    # Ajusta el header si la API lo requiere (p. ej. "x-api-key")
    req <- req |> req_headers(Authorization = paste("Bearer", api_key))
  }

  resp <- req |> req_perform()
  resp |> resp_check_status()

  # Parseo JSON
  txt <- resp |> resp_body_string()
  j <- jsonlite::fromJSON(txt, flatten = TRUE, simplifyDataFrame = TRUE)

  # Si hay un campo 'data', úsalo; si no, intenta coerción directa
  if (is.list(j) && !is.null(j$data)) {
    df <- j$data
  } else if (is.data.frame(j)) {
    df <- j
  } else {
    df <- as.data.frame(j)
  }

  # Limpieza básica y detección de fechas
  df <- df |>
    janitor::clean_names() |>
    mutate(
      across(matches("fecha|date|period|mes|anio|año"),
             \(x) suppressWarnings(parse_date_time(as.character(x),
                                                   orders = c("Ymd","Ym","Y-m","Y-m-d","d/m/Y","m/Y","Y
```

```
    )

  tibble::as_tibble(df)
}

# --- DESCARGA ----------------------------------------------------------------

dir.create(OUT_DIR, showWarnings = FALSE, recursive = TRUE)

data <- fetch_dataestur(API_URL, if (USE_API_KEY) API_KEY else NULL)

# Vista rápida
print(glimpse(data), n = 50)

# Guardado CSV (UTF-8)
readr::write_csv(data, OUT_FILE)
message("Guardado: ", OUT_FILE)

# --- EXTRA: filtrar por rango temporal (opcional) ----------------------------
# data_filtrada <- data |>
#   filter(fecha >= as.Date("2023-01-01") & fecha <= as.Date("2024-12-31"))
# write_csv(data_filtrada, file.path(OUT_DIR, "dataestur_2023_2024.csv"))
```

6 APIs en Python para acceso a datos de DATAESTUR

Esta sección ofrece utilidades en **Python** para conectarse a la API de DATAESTUR.
Incluye instalación de dependencias, un cliente sencillo, ejemplos de descarga a CSV/Parquet y descargas concurrentes.

6.1 Instalación rápida

```
# pip install requests pandas aiohttp python-dateutil pyarrow
```

6.2 Cliente reutilizable

```
import requests
import pandas as pd
from typing import Optional, Dict, Any

class DataesturAPI:
    BASE = "https://api.dataestur.es/v1"

    def __init__(self, session: Optional[requests.Session] = None, timeout: int = 60):
        self.s = session or requests.Session()
        self.timeout = timeout

    def _get(self, path: str, params: Optional[Dict[str, Any]] = None) -> Dict[str, Any]:
        url = f"{self.BASE}{path}"
```

```
        r = self.s.get(url, params=params, timeout=self.timeout)
        r.raise_for_status()
        return r.json()

    def series_metadata(self, series_id: str) -> Dict[str, Any]:
        return self._get(f"/series/{series_id}")

    def series_data(self, series_id: str, **params) -> Dict[str, Any]:
        return self._get(f"/series/{series_id}/datos", params=params)

    @staticmethod
    def to_dataframe(payload: Dict[str, Any]) -> pd.DataFrame:
        data = payload.get("Data", [])
        return pd.DataFrame(data)
```

6.3 Ejemplo de uso: FRONTUR y EGATUR

```
from pathlib import Path

api = DataesturAPI()
SERIES = {
    "FRONTUR_TURISTAS": {"fecha_desde": "2019-01"},
    "EGATUR_GASTO_TOTAL": {"fecha_desde": "2019-01"},
}

out_dir = Path("data/raw")
out_dir.mkdir(parents=True, exist_ok=True)

for sid, params in SERIES.items():
    payload = api.series_data(sid, **params)
    df = api.to_dataframe(payload)
    df.to_csv(out_dir / f"{sid}.csv", index=False)
    df.to_parquet(out_dir / f"{sid}.parquet", index=False)
    print(sid, df.shape)
```

6.4 Descarga concurrente de múltiples series

```
import asyncio, aiohttp

BASE = "https://api.dataestur.es/v1"

async def fetch_series(session, sid, **params):
    url = f"{BASE}/series/{sid}/datos"
    async with session.get(url, params=params) as resp:
        resp.raise_for_status()
        return sid, await resp.json()

async def main(series_params, out_dir):
    out_dir.mkdir(parents=True, exist_ok=True)
```

```
    async with aiohttp.ClientSession() as session:
        tasks = [fetch_series(session, sid, **p) for sid, p in series_params.items()]
        for coro in asyncio.as_completed(tasks):
            sid, payload = await coro
            df = pd.DataFrame(payload.get("Data", []))
            df.to_csv(out_dir / f"{sid}.csv", index=False)
            print("OK:", sid, df.shape)

SERIES = {
    "FRONTUR_TURISTAS": {"fecha_desde": "2018-01"},
    "EGATUR_GASTO_TOTAL": {"fecha_desde": "2018-01"},
}

asyncio.run(main(SERIES, Path("data/async")))
```