DOCUMENTACION SCRIPT ESTADÍSTICA BÁSICA

GRUPO 1

Contenido

Introducción

API en FastAPI que calcula los KPIs necesarios de los tickets almacenado, en nuestro caso en MongoDB Atlas para tres tipos.

- Combustible (“Combustible”)
- Vehiculo eléctrico (“eléctrico”)
- Peaje(“Peaje”)

Incluye:

- Endpoints de KPIs por dominio y un “todo en uno” (/kpis/all).
- Endpoints alias por gráfica (/charts/{domain}/{section}/{field}).
- Utilidades de debug/salud.
- Sección de Sostenibilidad unificada (EV + ICE) bajo /v1/sustainability/*, con agregados por empresa, usuario, vehículo y mes.

Requirements

- Python (3.12.10) + → Tambien testado en Python en Python 3.13)
- Mongo DB Atlas o Mongo DB accesible por red.
- Paquetes de Python
  - FastAPI
  - Uvicorn
  - Pymongo
  - Dnspython
  - Pandas
  - Numpy
  - Python-Dotenv

Configuración

Crea un archivo “.env” en el mismo directorio que “main.py”

Bloque de código

```
DB_URL=mongodb+srv://<usuario>:<pass>@<cluster>/<params>
DB_NAME=Prueba1


# (opcionales, si no se definen se usan los defaults)
COL_FUEL=”Combustible”
COL_EV=”eléctrico”
COL_TOLL=”Peaje”
COL_SUS=Sustainability
```

La app carga automáticamente este “.env” al iniciar

Arranque

Desde la carpeta donde esta “main.py” ejecutar el script, para luego ejecutar en el terminal el siguiente bloque:

Bloque de código

```
uvicorn main:app –reload
```

- Documentacion interactiva: http://127.0.0.1:8000/docs
- ReDoc: http://127.0.0.1:8000/redoc

Colecciones y esquema mínimo

La API es tolerante: si faltan ciertos campos, genera defaults para que no fallen las agregaciones

Combustible (Combustible)

Cabecera (usados):

- “idTicket”
- “empresaNombre”
- “idUsuario”
- “fechaEmision”
- “horaEmision” *(opcional; default “00:00:00”)*
- “metodoPago”
- “baseImponible”
- “iva”
- “total”
- “lineas” *(lista/dict/string JSON-ish)*

Líneas (dentro de “**líneas**”):

- “producto”
- “litros”
- “precioPorLitro” *(alias: “precio_unitario”, “precio”)*
- “importe” *(opcional; si falta “precioPorLitro”, se usa “importe/litros”)*

Derivados por la API:

- Temporales: “dt”, “mes” formato (YYYY-MM), “hora”, “dia_semana”
- Normalizados líneas: “precio_litro”, “importe_linea”
- Para líneas: “empresaTransporte” = “empresaNombre”

Vehículo Eléctrico (electrico)

Cabecera (usados):

- Igual que combustible

Líneas (dentro de lineas):

- "producto"
- Energía: "kwh" *(alias: "energía"," energia_kwh")*
- Precio: "precio_kwh" *(alias: "precioUnitario", "precio_unitario", "precio")*
- "importe" *(opcional; si falta "precio_kwh", se usa "importe/kwh")*
- "tipoCorriente"/"tarifa", "potenciaKW"/"potenciaMaxKW"/ "power_kw"

Derivados por API:

- Temporales: "dt", "mes", "hora", "dia_semana"
- Normalizados: "kwh", "precio_kwh", "importe_linea"

Peaje (Peaje)

Campos usados:

- "fechaHora"
- "importe" *(puede ser texto: "5,50 €"; la API lo convierte SIEMPRE a float)*
- "idUsuario"
- "empresaNombre"
- "autopista"
- "formaPago"
- "referencia" *(se usa como "idTicket" si no existe)*
- *(opcionales)* "tipoDocumento", "concesionaria", "categoriaVehiculo", "localización", "provincia"

Derivados por API:

- Temporales: "mes", "hora", "dia_semana", "is_weekend"
- "idTicket" (desde referencia si falta)
- "empresa" (copia de "empresaNombre" para agregaciones)

Normalización y parsing:

- **Fechas/tiempo**
  - `add_time_cols_fuel_ev(df)`: crea `dt, mes, hora, dia_semana` desde `fechaEmision/horaEmision.`
  - `add_time_cols_toll(df)`: crea `mes, hora, dia_semana, is_weekend` desde `fechaHora`.

- **Explosión de líneas**
  - `explode_fuel_lines(df_fuel)`: obtiene por línea `litros, precio_litro, importe_linea.`
  - `explode_ev_lines(df_ev)`: obtiene por `línea kwh, precio_kwh, importe_linea, tipoCorriente/tarifa, potenciaKW.`

- **Parse genérico de líneas**
  - `parse_lineas(x)`: acepta lista/dict/string JSON-ish (convierte '→" y `None/True/False` a JSON antes de `json.loads/ast.literal_eval`).

- **Importe string → float** (Peaje)
  - `_to_num_eur(x)`: soporta formatos con miles y coma decimal; quita símbolos, normaliza y castea.

- **Filtros por fecha (Mongo)**
  - Cabeceras Combustible/EV: `fechaEmision` con `start_date, end_date`.
  - Peaje: `fechaHora` (nota: `end_date` se expande a **23:59:59** del día indicado).

Definición detallada de KPIs

Combustible

KPIs de usuario

- **"gasto_usuario_mes"**
  - Gasto total por usuario mes
  - Agrupación: "`sum(total)`" por "`idUsuario`","`mes`"
  - Salida: `[{"idUsuario","mes","total"}]`

- **"tickets_usuario_mes"**
  - Numero de tickets por usuario y mes
  - Agrupación → Filtrado desde "`lineas`": "`nunique("idTicket")`" por "`idUsuario","mes"`)
  - Salida: `[{"idUsuario","mes","tickets"}]`

- **"litros_usuario_mes"**
  - Litros por usuario y mes
  - Agrupacion →Filtrado desde "`lineas`": "`nunique("idTicket")`" por "`idUsuario","mes`")

- **“litros_medio_ticket_usuario”**
  - Litros medio por ticket de usuario
  - Calculo
    - Filtrado desde “lineas” → “sum(litros)” por “idUsuario”, “idTicket”
    - Media por “idUsuario”
  - Salida: [{“idUsuario”,”empresaTransporte”, “eur_l”}]

- “**precio_usuario_marca**” (Ponderado por litros)
  - Precio en €/L por empresa en base a CPO/estación
  - Agrupacion → Filtrado desde “lineas”: media ponderada por “litros” en “idUsuario”,”empresaTransporte”
  - Salida: [{“idUsuario”, “empresaTransporte”, “eur_l”}]

- **“métodos_usuario”**
  - Numero de tickets por método de pago para el periodo completo
  - Agrupacion: nunique(idTicket) por “idUsuario”, “metodoPago”.
  - Salida: [{“idUsuario”, “metodoPago”, “tickets”}]

- **“metodos_usuario_mes”**
  - Nº de tickets por método y mes, y su % sobre el total mensual del usuario
  - Calculo
    - “tickets = nunique(idTicket)” por “idUsuario”, “mes”, “metodoPago”
    - “pct = 100 * tickets / sum(tickets por idUsuario, mes)”
  - Salida: [{“idUsuario”, “mes”, “metodoPago”, “tickets”, “pct”}]

- **“días_mediana”**
  - Mediana de días entre repostajes
  - Salida: [{“idUsuario”, “dias_mediana”}]

- **“anomalias_usuario”**
  - Nº de tickets con líneas cuyo “precio_litro < 0.8” o “> 3.0”
  - Agrupación → Filtrado desde “lineas”: “nunique(“idTicket)” por “idUsuario”

- **“gasto_dia_semana”**
  - Gasto por dia semana
  - Agrupación → “sum(total”) por “idUsuario”, “dia_semana”.
  - Salida: [{“idUsuario”, “dia_semana”, “total”}]

KPIs de Empresa

- **"gasto_mes_emp"**:
  - Gasto total por empresa y mes
  - Agrupación: "sum(total)" por "empresaNombre", "mes" → renombrado "empresa".
  - Salida → [{"empresa","mes","total"}]

- **"gasto_total_emp"**:
  - Gasto total en el periodo por empresa
  - Agrupación → sum(total) por "empresaNombre" aplicado a → "empresa"
  - Salida → [{"empresa","gasto_total_periodo"}]

- **"vehiculos_emp"**:
  - Numero de vehículos activos en el periodo por empresa
  - Agrupación → "nunique(idUsuario)" por "empresaNombre" aplicado a "empresa"
  - Salida → [{"empresa","num_vehiculos"}]

- **"gasto_medio_veh":**
  - Gasto medio por vehículo dentro de la empresa
  - Calculo
    - "sum(total)" por "empresaNombre", "idUsuario"
    - media por "empresaNombre"
  - Salida: [{"empresa","gasto_medio_por_vehiculo"}]

- **"ranking_veh" –** (vehiculo): "sum(total)" por "empresaNombre", "idUsuario".

- **"litros_mes_emp":** Filtrado desde "líneas" → "sum(litros)" por "empresaTransporte", "mes" aplicado a "empresa"

- **"litros_veh_emp"**:
  - Litros por empresa y usuario
  - Agrupacion → sum(litros) por "empresaTransporte","idUsuario
  - Salida: [{"empresa","idUsuario","litros}]
- **"precio_global_emp"** (ponderado): Filtrado desde "lineas" → media ponderada por "litros" en "empresaTransporte".

- **“gasto_emp_user_mes”**

    - Gasto por empresa, usuario y mes
    - Agrupación → `sum(total)` por **“**`empresaNombre”, “idUsuario”, “mes”`
    - Salida: `[{“empresa”, “idUsuario”, “mes”, “total”}]`

- **“tickets_emp_user_mes”**

    - Nº de tickets por empresa, usuario y mes
    - Agrupación → `nunique(idTicket)` por “`empresaNombre”, “idUsuario”, “mes”`
    - Salida: `[{“empresa”, “idUsuario”, “mes”, “tickets”}]`

- **“litros_emp_user_mes”**

    - Litros por empresa, usuario y mes
    - Agrupación → `sum(litros)` por “`empresaTransporte”, “idUsuario”, “mes”`
    - o Salida: `[{“empresa”, “idUsuario”, “mes”, “litros”}]`

- **“precio_emp_user”** `(ponderado por litros)”`

    - Precio medio €/L por empresa y usuario
    - Agrupación → desde `lineas:` media ponderada por `“litros”` en `“empresaTransporte”, “idUsuario” * eur_l = sum(precio_litro * litros) / sum(litros)`
    - Salida: `[{“empresa”, “idUsuario”, “eur_l”}]`

- **“consumo_emp_user_mes”**

    - Alias del consumo mensual por usuario → igual que `“litros_emp_user_mes”`
    - Salida: `[{“empresa”, “idUsuario”, “mes”, “litros”}]`

Vehiculo eléctrico

KPIs de usuario

Básicamente igual que combustible, únicamente cambiando “litros” por → “kWh” y “eur_l” por → eur_kwh”

- **“gasto_usuario_mes”:** `“sum(total)”` por `“idUsuario”, “mes”`.

- **“tickets_usuario_mes”**: `“nunique(idTicket”)` por `“idUsuario”, “mes”`.

- **“kwh_usuario_mes”**: Filtrado por “líneas” → “sum(kwh”) por “idUsuario”, “mes”

- **“kwh_medio_ticket_usuario”** : Filtrado por “lineas” → “sum(kwh)” por “idUsuario”, “idTicket” → media por “idUsuario”.

- **“precio_usuario_cpo”** (ponderado): Filtrado por “líneas” → media ponderada por “kwh” en “idUsuario”, “empresaTransporte”.

- **“metodos_usuario”**, **“metodos_usuario_mes”**, **“dias_mediana”**, **“gasto_dia_semana”**

    - Básicamente igual a combustibles , mismo calculo aplicado a coche eléctrico

- **“anomalias_usuario”**: Filtrado por “líneas” con “precio_kwh < 0.20” o “> 1.50” → “nunique(idTicket”) por “idUsuario”.

KPIs de Empresa

- **“gasto_mes_emp”**

    - Gasto total por empresa y mes.
    - Agrupación: “sum(total)” por “empresaNombre”, “mes” → renombrar “empresaNombre” a “empresa”.
    - Salida: `[{"empresa","mes","total"}]`

- **“gasto_total_emp”**

    - Gasto total del período por empresa.
    - Agrupación: “sum(total)” por “empresaNombre” → “empresa”.
    - Salida: `[{"empresa","gasto_total_periodo"}]`

- **“vehiculos_emp”**

    - **N**úmero de vehículos (usuarios) activos por empresa en el período.
    - Agrupación: “nunique(idUsuario)” por “empresaNombre” → “empresa”.
    - Salida: `[{"empresa","num_vehiculos"}]`

- **“gasto_medio_veh”**

    - Gasto medio por vehículo dentro de cada empresa.
    - **S**umar “total” por “empresaNombre,idUsuario” y luego hacer media por “empresaNombre (cabecera)” → “empresa”.
    - Salida: `[{"empresa","gasto_medio_por_vehiculo"}]`

- **“ranking_veh”**

    - Gasto por vehículo para ranking interno de la empresa.

- Agrupación: "sum(total" por "empresaNombre","idUsuario" → "empresa".
- Salida: [{"empresa","idUsuario","gasto_usuario"}]

- **"kwh_mes_emp"**

  - Energía total (kWh) por empresa y mes.
  - Agrupación: "sum(kwh)" por "empresaTransporte,mes" Filtrado desde "líneas"→ renombrar "empresaTransporte" a "empresa".
  - Salida: [{"empresa","mes","kwh"}]

- **"kwh_veh_emp"**

  - "kWh" consumidos por vehículo dentro de cada empresa.
  - Agrupación: "sum(kwh") por "empresaTransporte,idUsuario" Filtrado desde "líneas" → "empresa".
  - Salida: [{"empresa","idUsuario","kwh"}]

- **"precio_global_emp" (ponderado por kWh)"**

  - Precio efectivo medio en €/kWh de la empresa (ponderado por energía).
  - Agrupación; Media ponderada por "kwh" en "empresaTransporte" Filtrado desde "líneas": → "empresa".
  - Salida: [{"empresa","eur_kwh"}]

- **"gasto_emp_user_mes"**

  - Gasto por empresa, usuario y mes
  - Agrupación → sum(total) por "empresaNombre", "idUsuario", "mes"
  - Salida: [{"empresa", "idUsuario", "mes", "total"}]

- **"tickets_emp_user_mes"**

  - Nº de tickets por empresa, usuario y mes
  - Agrupación → nunique(idTicket) por "empresaNombre", "idUsuario", "mes"
  - Salida: [{"empresa", "idUsuario", "mes", "tickets"}]

- **"kwh_emp_user_mes"**

  - kWh por empresa, usuario y mes
  - Agrupación → sum(kwh) por "empresaTransporte", "idUsuario", "mes"
  - Salida: [{"empresa", "idUsuario", "mes", "kwh"}]

- **"precio_emp_user"** (ponderado por kWh)

  - Precio medio €/kWh por empresa y usuario
  - Agrupación → desde `lineas: media ponderada` por `"kwh"` en `"empresaTransporte", "idUsuario"▪ eur_kwh = sum(precio_kwh * kwh) / sum(kwh)`
  - Salida: `[{"empresa", "idUsuario", "eur_kwh"}]`

- **"consumo_emp_user_mes"**

  - Alias del consumo mensual por usuario → igual que `"kwh_emp_user_mes"`
  - `Salida: [{"empresa", "idUsuario", "mes", "kwh"}]`

Peaje

La API convierte siempre "`importe`" a "`float`" (acepta "5,50 €", "1.234,56 €", etc.).

KPIs de usuario

- **"gasto_usuario_mes"**

  - Gasto de peajes por usuario y mes
  - Agrupación: `sum(importe)` por `idUsuario, mes`
  - Salida: `[{"idUsuario","mes","gasto_mes"}]`

- **"tickets_usuario_mes"**

  - Nº de pasos/tickets por usuario y mes
  - Agrupación: "`nunique(idTicket)`" por "`idUsuario`", "`mes`" (si falta, "`idTicket`" usa "`referencia`")
  - Salida: `[{"idUsuario","mes","tickets"}]`

- **"usuario_autopista_mes"**

  - Desglose por autopista dentro de cada mes del usuario
  - Cálculo (por "`idUsuario`", "`mes`", "`autopista`")
    - "`gasto_autopista_mes`" = `sum(importe)`
    - "`tickets_autopista_mes`" = `count()`
    - "`coste_medio_ticket`" = "`gasto_autopista_mes`" / "`tickets_autopista_mes`"
    - "`pct_gasto_usuario_mes`" = "`100 * gasto_autopista_mes`" / "`sum(gasto_autopista_mes`" del "`usuario`" en el mes)
  - Salida: `[{"idUsuario","mes","autopista","gasto_autopista_mes","tickets_autopista_mes","coste_medio_ticket","pct_gasto_usuario_mes"}]`

- **"metodos_usuario_mes"**

  - Nº de tickets por método y mes + % mensual
  - Cálculo

    - "tickets" = "nunique(idTicket") por "idUsuario","mes", "formaPago"
    - "pct" = 100 * "tickets" / "sum("tickets" por "idUsuario", "me"s)
  - Salida: [{"idUsuario","mes","formaPago","tickets","pct"}]

- **"dias_mediana"**
  - Mediana de días entre pasos de peaje
  - Agrupación/Calc: mediana de diferencias de "fechaHora" por "idUsuario"
  - Salida: [{"idUsuario","dias_mediana"}]

- **"pct_finde_usuario"**
  - % de pasos por peaje que caen en fin de semana
  - Agrupación: mean("is_weekend")*100 por "idUsuario"
  - Salida: [{"idUsuario","pct_finde"}]

- **"gasto_dia_semana"**
  - Gasto por día de la semana
  - Agrupación → **sum(importe)** por **"idUsuario", "dia_semana"**
  - Salida: **[{"idUsuario", "dia_semana", "gasto"}]**

- **"gasto_usuario_autopista_mes"**
  - Gasto por autopista y mes del usuario
  - Agrupación → **sum(importe)** por **"idUsuario", "autopista", "mes"**
  - Salida: **[{"idUsuario", "autopista", "mes", "gasto"}]**

<u>KPIs de Empresa</u>

- **"gasto_mes_emp"**
  - Gasto total por empresa y mes
  - Agrupación → sum(importe) por "empresaNombre", "mes"
  - Salida → [{"empresa", "mes", "gasto"}]

- **"gasto_total_emp"**
  - Gasto total del período por empresa
  - Agrupación → sum(importe) por "empresaNombre"
  - Salida → [{"empresa", "gasto_total_periodo"}]

- **"vehiculos_emp"**
  - Nº de usuarios/vehículos por empresa
  - Agrupación → nunique(idUsuario) por "empresaNombre"
  - Salida → [{"empresa", "num_vehiculos"}]

- **"gasto_medio_veh"**

  - Gasto medio por vehículo (usuario) en la empresa
  - Cálculo → `sum(importe)` por `("empresaNombre", "idUsuario")` → `mean` por `"empresaNombre"`
  - Salida → `[{"empresa", "gasto_medio_por_vehiculo"}]`

- **"ranking_veh"**

  - Gasto por usuario dentro de la empresa
  - Agrupación → `sum(importe)` por `"empresaNombre", "idUsuario"`
  - Salida → `[{"empresa", "idUsuario", "gasto_usuario"}]`

- **"gasto_emp_user_mes"**

  - Gasto por empresa, usuario y mes
  - Agrupación → `sum(importe)` por `"empresaNombre", "idUsuario", "mes"`
  - Salida → `[{"empresa", "idUsuario", "mes", "gasto"}]`

- **"tickets_emp_user_mes"**

  - Nº de tickets por empresa, usuario y mes
  - Agrupación → `nunique(idTicket)` por `"empresaNombre", "idUsuario", "mes"` *(si no existiese `idTicket`, usar `nunique(referencia)` o `size`)*
  - Salida → `[{"empresa", "idUsuario", "mes", "tickets"}]`

- **"metodos_emp_user_mes"**

  - Nº de tickets por método de pago, por empresa, usuario y mes
  - Cálculo → `tickets = nunique(idTicket)` por `"empresaNombre", "idUsuario", "mes", "formaPago"`
  - Salida → `[{"empresa", "idUsuario", "mes", "formaPago", "tickets"}]`

- **"gasto_emp_autopista_mes"**

  - Gasto por autopista y mes de la empresa
  - Agrupación → `sum(importe)` por `"empresaNombre", "autopista", "mes"`
  - Salida → `[{"empresa", "autopista", "mes", "gasto"}]`

- **"gasto_dia_semana_emp"**

  - Gasto por día de la semana de la empresa
  - Agrupación → `sum(importe)` por `"empresaNombre", "dia_semana"`
  - Salida → `[{"empresa", "dia_semana", "gasto"}]`

- **"gasto_hora_emp"**

  - Gasto por franja horaria de la empresa
  - Agrupación → `sum(importe) por "empresaNombre", "hora"`
  - Salida → `[{"empresa", "hora", "gasto"}]`

- **"tickets_mes_emp"**

  - Nº de tickets por empresa y mes
  - Agrupación → `nunique(idTicket) por "empresaNombre", "mes"`
  - Salida → `[{"empresa", "mes", "tickets"}]`

- **"gasto_autopista_emp"**

  - Gasto total por autopista (todo el período) de la empresa
  - Agrupación → `sum(importe) por "empresaNombre", "autopista"`
  - Salida → `[{"empresa", "autopista", "gasto"}]`

- **"gasto_forma_pago_emp"**

  - Gasto total por método de pago de la empresa
  - Agrupación → **`sum(importe)`** `por` **`"empresaNombre", "formaPago"`**
  - Salida → **`[{"empresa", "formaPago", "gasto"}]`**

- **"tickets_forma_pago_emp_mes"**

  - Nº de tickets por método de pago y mes de la empresa
  - Agrupación → **`nunique(idTicket)`** `por` **`"empresaNombre", "formaPago", "mes"`**
  - Salida → **`[{"empresa", "formaPago", "mes", "tickets"}]`**

KPIs de empresa – ALL

- **"gasto_mes_emp_all"**

  - Gasto mensual por empresa desglosado por dominio (combustible, ev, peaje)
  - Agrupación / Cálculo → concatenación de:
    - Combustible: `sum(total)` por `"empresaNombre", "mes"` → se renombra `a {empresa, mes, gasto}` y se añade `domain="combustible"`
    - EV: `sum(total)` por `"empresaNombre", "mes"` → `{empresa, mes, gasto}, domain="ev"`
    - Peaje: `sum(importe)` por `"empresaNombre", "mes"` → `{empresa, mes, gasto}, domain="peaje"`
  - Salida → `[{"empresa", "mes", "gasto", "domain"}]`

- **"gasto_emp_user_mes_all"**
  - Gasto mensual por empresa y usuario desglosado por dominio
  - Agrupación / Cálculo → concatenación de:
    - Combustible: `sum(total)` por "empresaNombre", "idUsuario", "mes" → {empresa, idUsuario, mes, gasto}, domain="combustible"
    - EV: `sum(total)` por "empresaNombre", "idUsuario", "mes" → {empresa, idUsuario, mes, gasto}, domain="ev"
    - Peaje: `sum(importe)` por "empresaNombre", "idUsuario", "mes" → {empresa, idUsuario, mes, gasto}, domain="peaje"
  - Salida → [{"empresa", "idUsuario", "mes", "gasto", "domain"}]

KPIs Sostenibilidad - EV + ICE (colección "Sustainability" o materialización desde dominios)

Cálculos base cuando se materializa (si no hay colección):

- ICE: `kgCO2_ice = litros * 2.69`
- EV: `kgCO2e_ev = kwh * 1.096 * 0.283`
- **kgCO2_total = kgCO2_ice + kgCO2e_ev`
- `idVehiculo` autogenerado si falta: "{idEmpresa}-{idUsuario}-ICE|EV"
- `mes` como `YYYY-MM` (o "NA" si falta)
- Filtros: `from` / `to` (YYYY-MM, filtro lexical), `propulsion ∈ {EV, ICE}`

/v1/sustainability/companies

- "**companies**"
  - Métricas agregadas por empresa en el rango
  - Agrupación → por "idEmpresa":
    - `sum(kwh), sum(litros), sum(kgCO2e_ev), sum(kgCO2_ice), sum(kgCO2_total)`
    - `n_usuarios = nunique(idUsuario)`
    - `n_vehiculos = nunique(idVehiculo)`
    - `share_co2_ev = kgCO2e_ev / (kgCO2e_ev + kgCO2_ice)` (si denom. > 0, si no → null)
  - Salida (paginada): `{ "count": N, "items": [ {"idEmpresa","kwh","litros","kgCO2e_ev","kgCO2_ice","kgCO2_total","n_usuarios","n_vehiculos","share_co2_ev"} ] }`

/v1/sustainability/companies/{idEmpresa}/summary

- **"company_summary"**
  - Resumen agregado de una empresa
  - Agrupación → igual que "companies", filtrado por `idEmpresa` (un registro)
  - Salida: `{"idEmpresa","kwh","litros","kgCO2e_ev","kgCO2_ice","kgCO2_total","n_usuarios","n_vehiculos","share_co2_ev"}`

/v1/sustainability/companies/{idEmpresa}/months

- **"company_months"**
  - Serie mensual por empresa
  - Agrupación → por "idEmpresa", "mes":
    - `sum(kwh), sum(litros), sum(kgCO2e_ev), sum(kgCO2_ice), sum(kgCO2_total)`
    - `share_co2_ev`
  - Salida (paginada):
    `{ "count": N, "items": [ {"idEmpresa", "mes", "kwh", "litros", "kgCO2e_ev", "kgCO2_ice", "kgCO2_total", "share_co2_ev"} ] }`

/v1/sustainability/companies/{idEmpresa}/users

- **"company_users"**
  - Métricas por usuario dentro de una empresa
  - Agrupación → por "idEmpresa", "idUsuario":
    - `sum(kwh), sum(litros), sum(kgCO2e_ev), sum(kgCO2_ice), sum(kgCO2_total`
    - `n_vehiculos = nunique(idVehiculo)`
    - `share_co2_ev`
  - Salida (paginada):
    `{ "count": N, "items": [ {"idEmpresa", "idUsuario", "kwh", "litros", "kgCO2e_ev", "kgCO2_ice", "kgCO2_total", "n_vehiculos", "share_co2_ev"} ] }`

/v1/sustainability/companies/{idEmpresa}/users/{idUsuario}/months

- **"company_user_months"**
  - Serie mensual por usuario dentro de la empresa
  - Agrupación → por "idEmpresa", "idUsuario", "mes":
    - `sum(kwh), sum(litros), sum(kgCO2e_ev), sum(kgCO2_ice), sum(kgCO2_total)`
    - `share_co2_ev`
  - Salida (paginada):
    `{ "count": N, "items": [ {"idEmpresa", "idUsuario", "mes",`

"kwh", "litros", "kgCO2e_ev", "kgCO2_ice", "kgCO2_total", "share_co2_ev"}] }

/v1/sustainability/vehicles

- **"vehicles"**
  - Métricas por vehículo (filtrable por empresa, usuario y propulsión)
  - Agrupación → por "idEmpresa", "idUsuario", "idVehiculo", "propulsion":
    - sum(kwh), sum(litros), sum(kgCO2e_ev), sum(kgCO2_ice), sum(kgCO2_total)
  - Salida (paginada): { "count": N, "items": [{"idEmpresa", "idUsuario", "idVehiculo", "propulsion", "kwh", "litros", "kgCO2e_ev", "kgCO2_ice", "kgCO2_total"}] }

Endpoints

Salud y debugs

- "GET /health" — resumen y conteos de colecciones
- "GET /debug/config" — DB/colecciones/config
- "GET /debug/peek?n=3" — columnas y muestra por colección
- "GET /debug/distinct?collection=<Colección>&field=<campo>&limit=50" — distintos y top conteos
- "GET /debug/peaje_sample?n=5" — sample crudo de Peaje
- "GET /debug/peaje_after?..." — cómo queda Peaje tras normalizar (conversión de importe, fechas, etc.)

KPIs

- "GET /kpis/combustible"
- "GET /kpis/ev"
- "GET /kpis/peaje"

Endpoints Sostenibilidad

- `GET /v1/sustainability/companies`
  - Agregado por empresa (suma kwh, litros, kgCO2e_ev, kgCO2_ice, kgCO2_total,
  - n_usuarios, n_vehiculos) + share_co2_ev

- `GET /v1/sustainability/companies/{idEmpresa}/summary`
  - Resumen single empresa con métricas y share_co2_ev

- `GET /v1/sustainability/companies/{idEmpresa}/months`
  - Serie mensual por empresa (kwh, litros, kgCO2*, share_co2_ev)

- `GET /v1/sustainability/companies/{idEmpresa}/users`
  - Agregado por usuario dentro de la empresa

- `GET /v1/sustainability/companies/{idEmpresa}/users/{idUsuario}/months`
  - Serie mensual por usuario

- `GET /v1/sustainability/vehicles`
  - Agregado por vehículo (empresa, usuario, idVehiculo, propulsion)

Query params (opcionales en todos):

- `start_date`
  - `YYYY-MM-DD` (Combustible/EV usan fechaEmision; Peaje usa fechaHora)
- `end_date`
  - `YYYY-MM-DD` (en Peaje se incluye hasta 23:59:59 del día)
- `Empresa`
  - filtra por `empresaNombre`
- `idUsuario`
  - filtra por `idUsuario`
- `section`
  - "empresa" | "usuario" (para devolver solo esa rama)
- `fields`
  - claves separadas por coma para filtrar datasets devueltos (p.ej. "gasto_mes_emp,precio_global_emp")

Helpers de debug recomendados

- **Estado rápido**:
  `GET /health`
- **Config y conteos**:
  `GET /debug/config`

- **Muestra por colección**:
  `GET /debug/peek?n=5`
- **Distribución de valores**:
  `GET /debug/distinct?collection=Peaje&field=idUsuario`
- **Peaje normalizado**:
  `GET /debug/peaje_after?n=5`

Resolución de problemas (Troubleshooting)

• Could **`not import module "main"`**: instala dependencias ("`python-dotenv`", "`pandas`", "`numpy`", …) y ejecuta uvicorn desde el venv correcto.

• Internal **Server Error en Peaje**: suele ser "`importe`" en string. La API ya lo convierte SIEMPRE; si persiste, revisa "`/debug/peaje_after`" y asegúrate de que "`importe_all_nan`" es "`false`".

• Colecciones vacías: confirma nombres exactos en "`.env`" ("`Combustible`", "`eléctrico`", "`Peaje`") y la "`DB_NAME`".

• Fechas: formato "`YYYY-MM-DD`". Para Peaje, "`end_date`" incluye hasta "`23:59:59`" del día.