DataAccess API

Manual simple de usuario

1. Puesta en marcha

1.1 Instalando dependencias

Para que este API pueda funcionar son necesarias las siguientes dependencias:

- SDK de .Net Core 3.1

- ASP de .Net Core 3.1

- .Net Core 3.1

- PostgreSQL 11.8

Pueden seguirse los siguientes tutoriales:

- https://docs.microsoft.com/es-es/dotnet/core/install/linux

- https://www.postgresql.org/docs/11/installation.html

1.2 Arrancando la API

1. El primer paso será la configuración de una base de datos sobre la cual realizar la persistencia y consulta de la información referente a los nuevos usuarios. Desde la página oficial de PostgreSQL puede consultarse una guía completa: https://www.postgresql.org/docs/11 .

2. Una vez generada una base de datos, será necesario configurar el programa para que acceda a la misma. Para ello se ha generado un archivo .json en el cual es posible introducir las credenciales y url de la base de datos. La API se encargará de gestionar esa información tras el primer arranque sin que el usuario tenga que realizar ninguna tarea adicional.

La ruta al archivo es **/APIs/DataAccess_API/DAOs/Connection/connectionProperties.json** . Se trata de sustituir la información de cada campo por la generada en el paso anterior:

```
{
   "server": "introducir url",
   "port": "introducir puerto",
   "userId": "introducir usuario",
   "pass": "introducir contraseña",
   "dataBase": "introducir base de datos"
}
```

1.3 Levantando la aplicación

1. Navegar hasta la ruta **APIs/DataAccess_API**.

2. Ejecutar el comando **dotnet build** desde la consola de comandos. Ref: https://docs.microsoft.com/es-es/dotnet/core/tools/dotnet-build

3. Ejecutar el comando **dotnet run** desde la consola de comandos. Ref: https://docs.microsoft.com/es-es/dotnet/core/tools/dotnet-run

4. Es posible sustituir los dos pasos anteriores por al arranque desde un **IDE**.

5. Esperar varias horas. La aplicación generará todas las bases de datos necesarias para su empleo e insertará todos los datos necesarios previa consulta de los mismos a organismos externos a la misma. Si tras varios minutos no se observase la creación de ninguna tabla en base de datos se recomienda detener la aplicación y comprobar la conexión al servidor de base de datos.

2. Usando la aplicación. Los métodos.

La aplicación se encuentra configurada para atender peticiones POST enviadas a la url https://localhost:5005/CovidDataBase . Si desea cambiarse el puerto de acceso es posible modificando el archivo **APIs/DataAccess_API/Properties/launchProperties.json** .

2.1 GetGeoZoneData

Este método está pensado para obtener los datos relativos a la evolución histórica de la pandemia de Covid-19 a nivel mundial desglosados por países. A través de él se pueden consultar los mismos para una lista de países y dentro de un rango de fechas predeterminado. La petición POST en formato json sería la siguiente:


```
{
   "method": "GetGeoZoneData",

   "covid_data": {

      "countries": [
         "ES",
         "AN"
      ],

      "dates": {
         "startDate": "31/12/2019",
         "endDate": "01/01/2020",
         "separator": "/"
      }
   }
}
```


El elemento "**method**" indica a qué método del API se está dirigiendo la consulta. "**covid_data**" posee dos componentes. Por un lado una lista "**countries**". Ésta es ampliable tanto como se necesite y consta de los códigos **ISO2** de todos los países de los que se desea obtener información. Por otro lado el elemento "dates" indica desde qué fecha hasta qué día se desea obtener datos. El formato ha de ser siempre el mismo: dd/MM/yyyy y el separador "/". En este punto el elemento "**separator**" no es obligatorio pero está presente pensando en un posible futuro desarrollo de nuevas funcionalidades como la admisión de varios formatos de consulta.

Una respuesta de ejemplo sería la que sigue:


```
[
   {
      "father": null,
      "sonList": null,
      "geoID": "ES",
      "code": "ESP",
      "name": "Spain",
      "population": 46723749,
      "dataList": [
         {
            "id": 2,
            "cases": 0,
            "deaths": 0,
            "cured": 0,
            "date": {
               "id": 2,
               "date": "01/01/2020",
               "dateSeparator": "/",
               "dateFormat": "dd/MM/yyyy"
            }
         },
         {
            "id": 1,
            "cases": 0,
            "deaths": 0,
            "cured": 0,
            "date": {
               "id": 1,
               "date": "31/12/2019",
               "dateSeparator": "/",
               "dateFormat": "dd/MM/yyyy"
            }
         }
      ]
   }
]
```


En el título siguiente – **3. El modelo de datos** - puede consultarse el modelo de datos empleado.

2.2 GetAllGeoZoneData

En esta ocasión el método está pensado para obtener la información de todos los países cuyos datos hayan sido registrados filtrando únicamente por rango de fechas. La petición a emplear sería la siguiente:


```
{
"method": "GetAllGeoZoneData",
   "covid_data": {
      "countries": [],
      "dates": {
         "startDate": "31/12/2019",
         "endDate": "01/01/2020",
         "separator": "/"
      }
   }
}
```


En este caso el elemento “countries” se ignorará completamente siendo únicamente necesario haber establecido un rango de fechas consistente. La respuesta seguirá el mismo modelo que la del método anterior pero su volumen será mucho mayor. Puede encontrarse un ejemplo completo en el archivo /Documentation/DataAccess_API/InputOutputExamples/POSTCompleteResponseExample.json .

2.3 GetAllGeoZoneDataForAllDates

Ha sido diseñado como medio para obtener absolutamente todos los datos de los que se dispone. La respuesta contendrá un listado de países que incluirán los datos diarios desde que se comenzó su recogida hasta el día en el que se realice la petición.

Un ejemplo sería el siguiente:


```
{
   "method": "GetAllGeoZoneData",

   "covid_data": {

      "countries": [],

      "dates": {
         "startDate": "",
         "endDate": "",
         "separator": ""
      }
   }
}
```


Como ejemplo de respuesta puede tomarse nuevamente el archivo **/Documentation/DataAccess_API/InputOutputExamples/POSTCompleteResponseExample.json** .

2.4 GetAllCountries

A través de él es posible conseguir una lista completa de países disponibles para consulta con su correspondiente código ISO2. Resulta útil si se necesita obtener los códigos ISO2 en tiempo de ejecución de una aplicación desde la cual se consultarán los datos de ésta.


```
{
   "method": "GetAllCountries",

   "covid_data": {

      "countries": [],

      "dates": {
         "startDate": "",
         "endDate": "",
         "separator": ""
      }
   }
}
```


Un ejemplo de respuesta acortada es:


```
[
   {
      "father": null,
      "sonList": null,
      "geoID": "AD",
      "code": "AND",
      "name": "Andorra",
      "population": 77006,
      "dataList": null
   },
   {
      "father": null,
      "sonList": null,
      "geoID": "AE",
      "code": "ARE",
      "name": "United_Arab_Emirates",
      "population": 9630959,
      "dataList": null }, ….]
```

Puede consultarse la respuesta completa en **/Documentation/DataAccess_API/InputOutputExamples/AllCountries.json** .

2.5 GetAllDates

Se trata de un método pensado para conocer de antemano las fechas disponibles para consulta. Simplemente devolvería una lista de todas ellas. La petición a emplear sería la siguiente:

```
{
   "method": "GetAllDates",

   "covid_data": {

      "countries": [],

      "dates": {
         "startDate": "",
         "endDate": "",
         "separator": ""
      }
   }
}
```

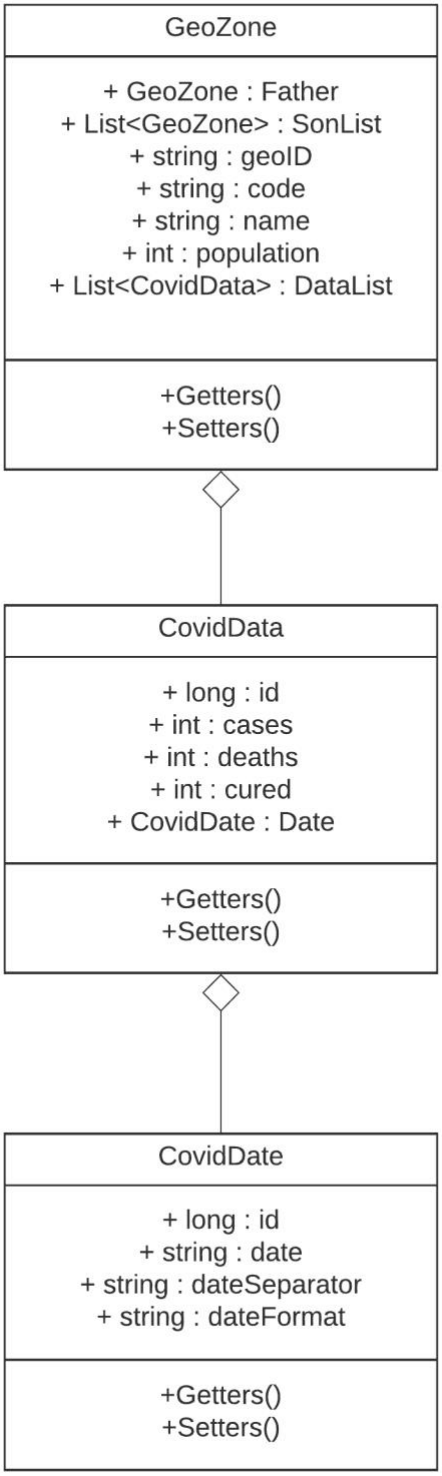
Y un ejemplo de respuesta acortada es el que sigue:

```
[
   {
      "id": 1,
      "date": "31/12/2019",
      "dateSeparator": "/",
      "dateFormat": "dd/MM/yyyy"
   },
   {
      "id": 2,
      "date": "01/01/2020",
      "dateSeparator": "/",
      "dateFormat": "dd/MM/yyyy"
   }, …
]
```

Puede consultarse la respuesta completa en **/Documentation/DataAccess_API/InputOutputExamples/AllDates.json** .

3. El modelo de datos

3.1 El modelo de datos propio



3.2 El modelo de datos de la petición de entrada