**Data import Service for VPD program**

**Documentación Técnica**

Versión 1.2

17 de septiembre de 2022

Presentación:

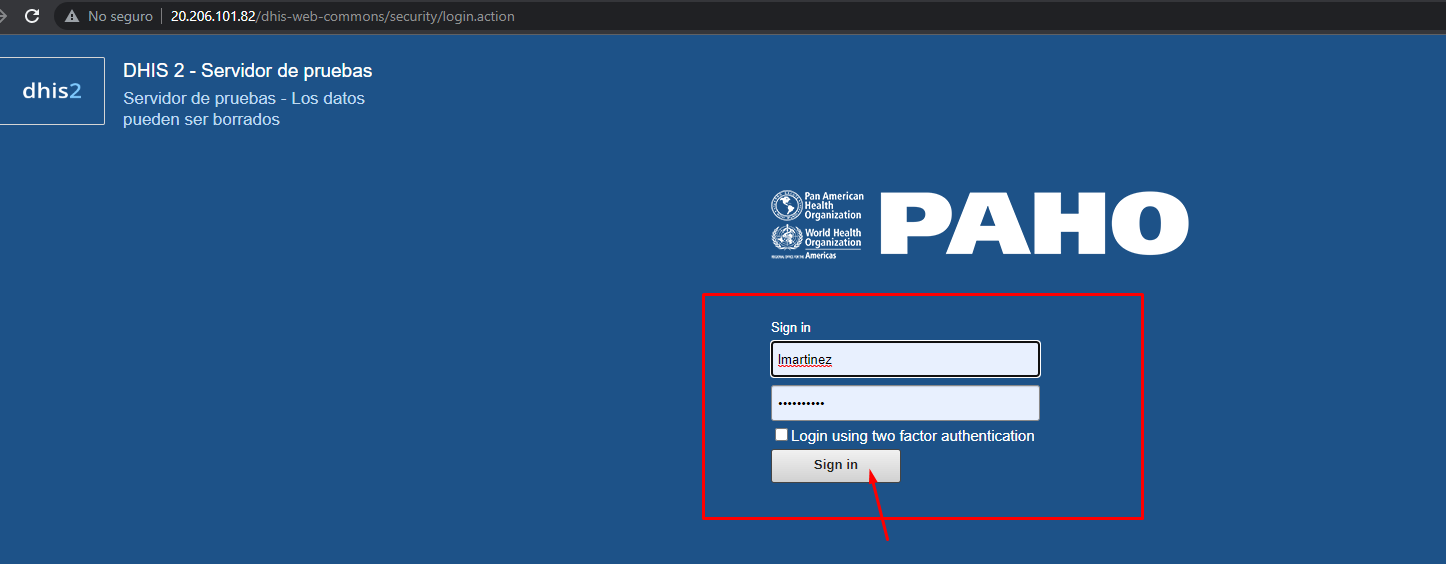
En el marco de colaboración entre la Universidad de Oslo, Hisp Colombia y la Organización Panamericana de la Salud (PAHO) se ha desarrollado una solución web que se compone de un servicio web que integra con el sistema DHIS2 y los datos en crudo recibidos en formatos estándar, y una aplicación web que ofrece al usuario final la capacidad de importar sus archivos estándar por país. Para su adecuada implementación y uso a continuación se presenta el documento técnico guía se que distribuye con las secciones descritas a continuación.

1. Configuración de correo para la notificación
2. Guía de uso de la aplicación web
3. Consulta del historial de las importaciones
4. Consumo de la API del servicio de importación
5. Publicación de servicio de importación(backend) y aplicación web (frontend)
6. Código fuente, arquitectura y flujo de trabajo
7. **¿Cómo ver el correo electrónico e idioma configurado en DHIS2?**

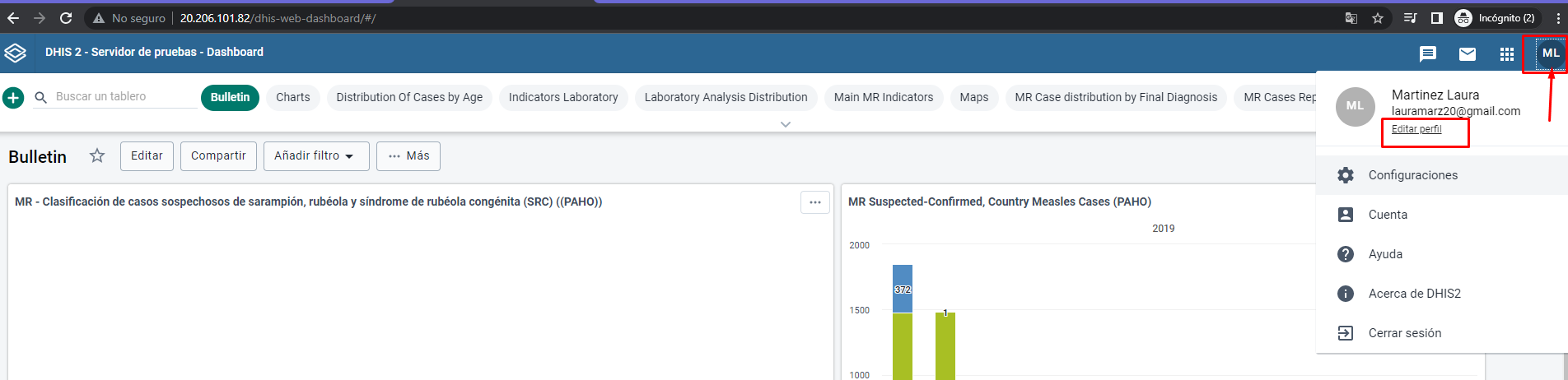
Para revisar las configuraciones de DHIS2, lo primer que se debe hacer es dirigirse a la siguiente ruta:

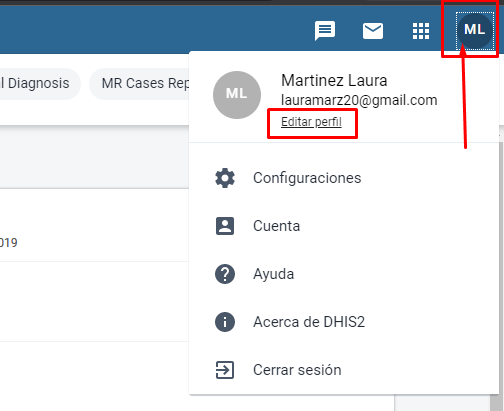
<http://20.206.101.82/dhis-web-commons/security/login.action>

Una vez aquí, se diligencia los datos de usuario y contraseña

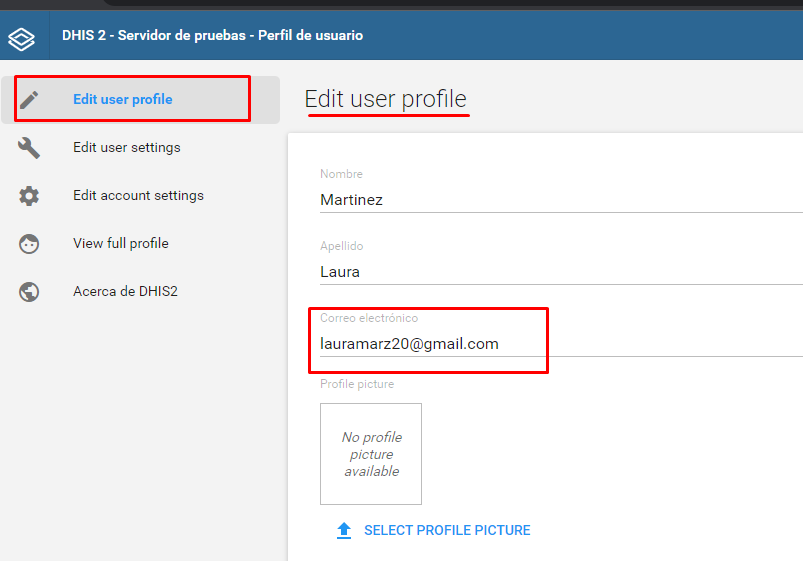


Una vez la autenticación sea correcta, la pantalla que carga es la de dashboard, en la opción del lado derecho donde están las iniciales del usuario autenticado, se da clic y se selecciona la opción

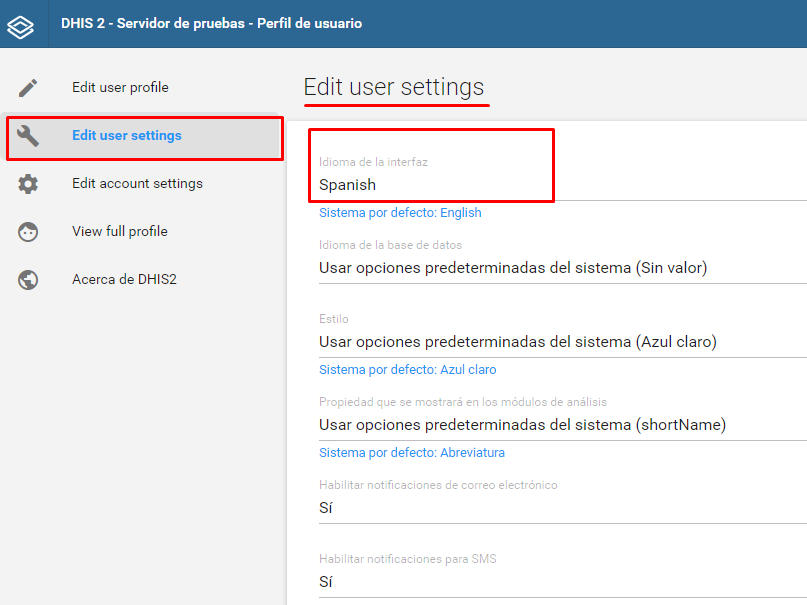




Una vez en la opción de Editar perfil, en lateral izquierdo hay varias opciones para revisar el correo electrónico en la primera (Edit user profile)



Para revisar el idioma, en la misma opción de editar perfil, en la opción lateral izquierda en la segunda opción (Edit user settings)

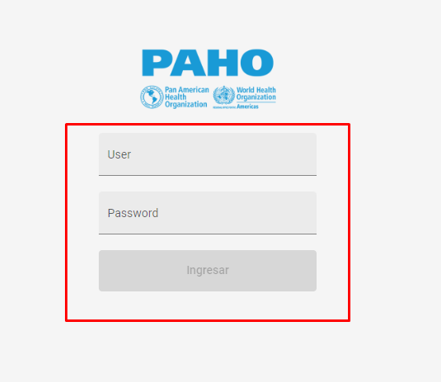


**Nota importante:** La aplicación de **VPD Data Import**, tiene toda su configuración basada en lo creado en DHIS2, así que para revisar el tema del idioma y del correo electrónico, se puede consultar así:

1. **Guía de uso de la aplicación web VPD data Import**
   1. **¿Cómo autenticarse en VPD Data Import?**

Para ingresar a la aplicación VPD Data Import, se ingresa la siguiente ruta por el navegador: <http://20.206.101.82/import/>

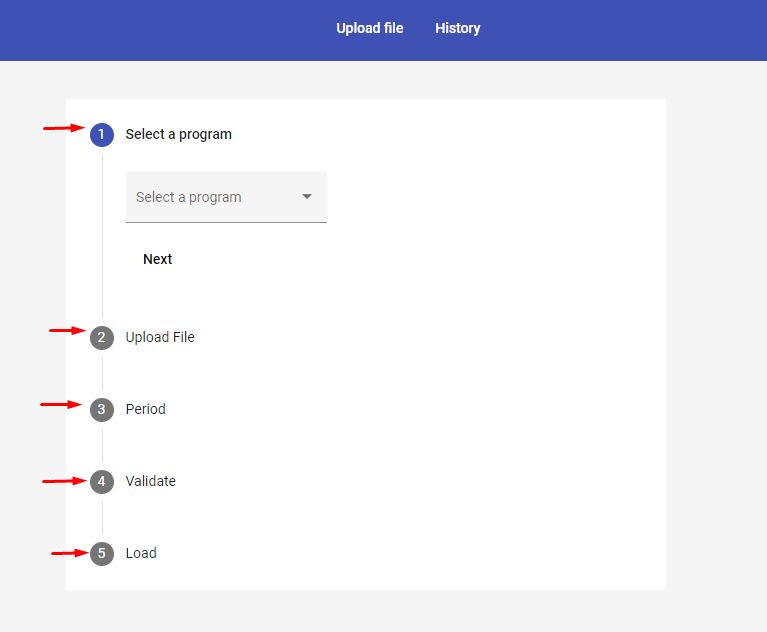
Una vez estando ahí, se diligencian los datos de usuario y contraseña (los mismos de DHIS2)



Una vez el proceso de autenticación sea correcto ya se puede continuar con el proceso de importación de la data.

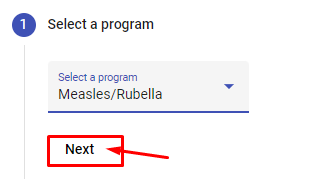
* 1. **Pasos de la aplicación VPD Data Import**

El proceso de importación de la data consta de 5 pasos, de los cuales unos son obligatorios y otros opcionales, a continuación, se detalla cada uno de estos:



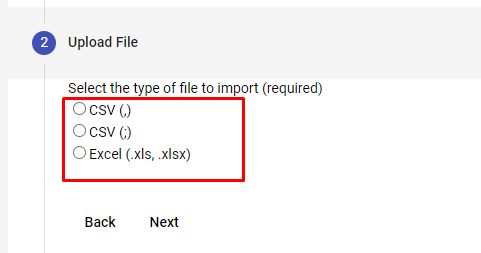
* + 1. **Paso 1: Selección de programa**

En este paso, se selecciona el programa al cual se va registrar la información que se va importar, para esta demostración se selecciona la opción Measles/Rubella, y siguiente.



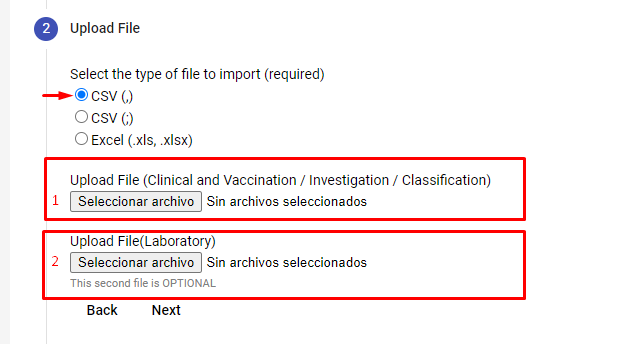
* + 1. **Paso 2: Cargar Archivo(s)**

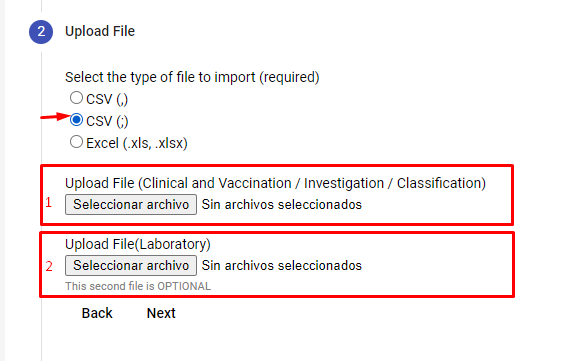
En esta este paso, se debe tener en cuenta varios detalles, el primero que los archivos permitidos para este proceso de importación son 3 **(CSV, XLS y XLSX)**, solo los antes mencionados procesan la aplicación, para iniciar se debe seleccionar que tipo de archivo se va a subir (obligatorio), las opciones de archivos CSV permiten 2 tipos de separador, la coma (,) y el punto y coma (;), es importante revisar cual es el separador que tiene el archivo a importar.



Se debe seleccionar alguna de las 2 opciones, los archivos XLS y XLSX están en una opción, esto para desplegar los controles donde se deben subir el o los archivos, si se selecciona cualquiera de las opciones de CSV, se hacen visibles 2 controles de adjuntar archivos en ese mismo formato (CSV).

* **Opción CSV**





El primer control de adjuntar es **Obligatorio** y es información relacionada con el tracked (registro de la persona), el Enrollment (registro de la persona al programa) y los Events (los datos del registro).

El segundo control de adjuntar es **Opcional** y es información relacionada con los Events (los datos del registro, pero *exclusivamente* para *Laboratorio*).

* **Opción Excel (xls, xlsx)**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

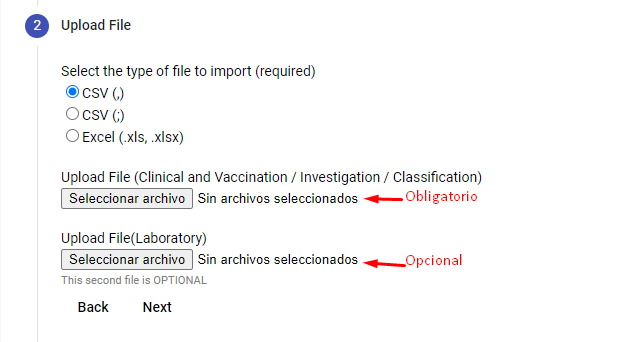
Solo se hará visible el primer control de adjuntar y como el caso anterior es **obligatorio** y puede contener 1 o 2 hojas.

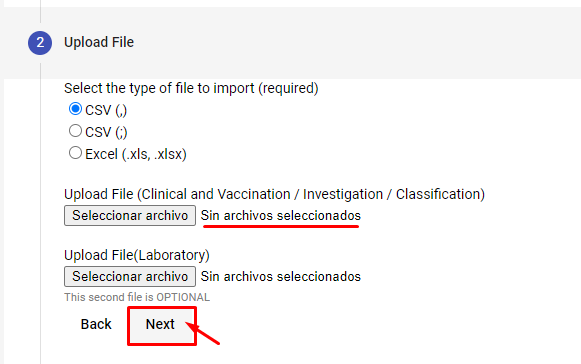
La primera hoja es **obligatoria** y tendrá información relacionada con el tracked (registro de la persona), el Enrollment (registro de la persona al programa) y los Events (los datos del registro).

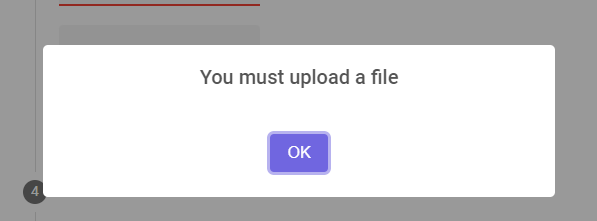
La segunda hoja es **Opcional** y tendrá información relacionada con los Events (los datos del registro, pero *exclusivamente* para *Laboratorio*).

* **Archivo Obligatorio (primer control de adjuntar)**

Como se mencionó anteriormente el contenido de control de adjuntar uno, que corresponde a la información de *Clinical and Vaccination /Investiogation/ Classification,* es obligatoria y en caso de que se continúe sin información aparecerá una alerta que no podrá avanzar hasta no completar la información de ese primer control.

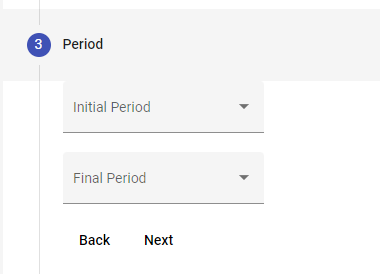






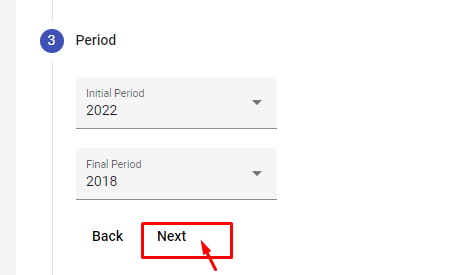
* + 1. **Paso 3: Periodos**

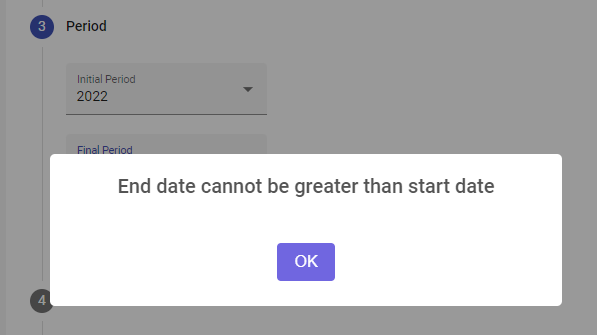
En este paso se deben seleccionar los periodos en cuales se va a registrar la data a importar que coincida con lo que está en los archivos (CSV, xls o xlsx).



Es importante resaltar que se debe ser muy cuidadoso con la elección de los periodos, puesto que estos son usados internamente para borrar data ya almacenada (enrollment y events), para poder actualizar sin problema la nueva que se va a importar.

También esta opción tiene una restricción que el periodo final no puede ser menor al periodo inicial, para tener presente:

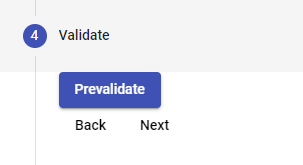




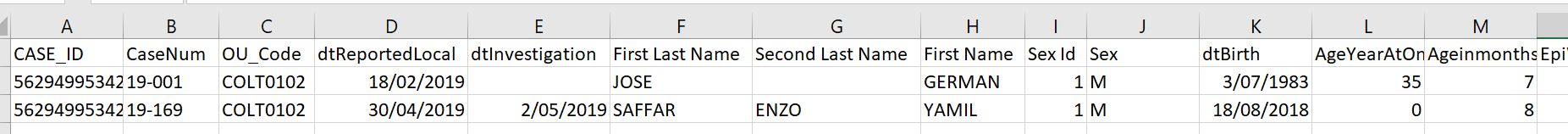
* + 1. **Paso 4: Pre-validación**

Este paso nos permite validar que el archivo tenga la estructura y que coincida con el tipo de dato y los tipos de valores que puede permitir.

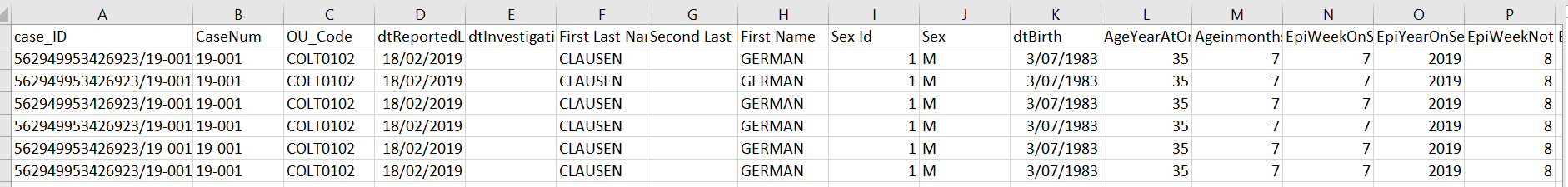
Esta pre-validación es funcional siempre y cuando se cumpla con el formato estándar y es **Opcional**, dentro de la aplicación.



Para un ejemplo de formato estándar de Rubeola a continuación imágenes de la estructura de los archivos, ya que admite 2 archivos el primero recibe los datos del tacked (registro de la persona), el Enrollment (registro de la persona al programa) y los Events (los datos del registro), el segundo corresponde solo a los Events relacionados a Laboratorio.



1er archivo (ejemplo)



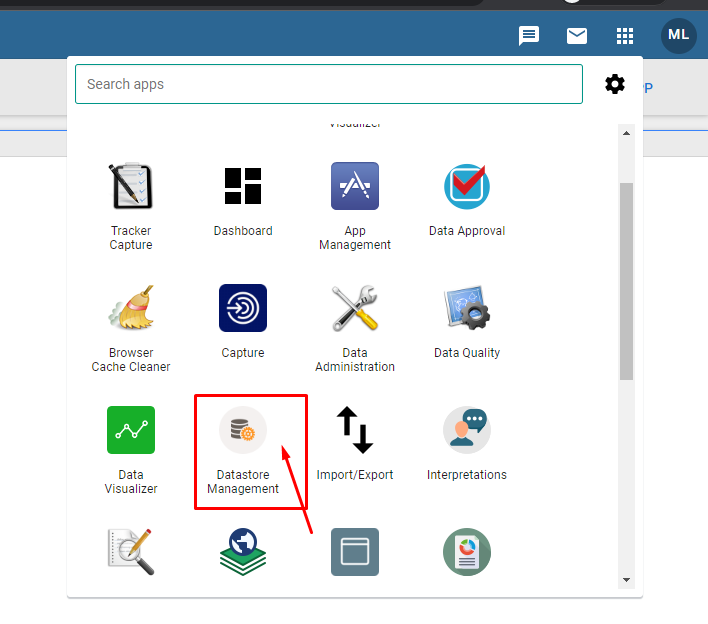
2do archivo (ejemplo)

A continuación, se detallan todas las columnas para el programa de Rubeola, en los dos archivos:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Columnas Estándar para Rubeola - 1er Archivo** | | CASE\_ID | | CaseNum | | OU\_Code | | dtReportedLocal | | dtInvestigation | | First Last Name | | Second Last Name | | First Name | | Sex Id | | Sex | | dtBirth | | AgeYearAtOnset | | AgeinmonthsAtOnset | | EpiWeekOnSet | | EpiYearOnSet | | EpiWeekNotification | | EpiYearNotification | | dtRashOnset | | FinalClass | | FinalClass | | BasisForConfirming | | BasisForConfirming | | BasisForDiscarding | | BasisForDiscarding | | OtherDiagnosis | | SourceOfInfection | | SourceOfInfection | | NumDosesMeaslesRubella | | dtLastDoseMeaslesRubella | | VaccineMeaslesRubellaType | | VaccineMeaslesRubellaType | | VacSource | | VacSource | | ActiveSearches | | ActiveSearches | | Traveled | | Traveled | | SettingWhereInfected | | Hospitalized | | Hospitalized | | CountryOfImportation | | RecDate | | VaccinatedAgainstMeaslesRubella | | VaccinatedAgainstMeaslesRubella | | patientFullName | | |  | | --- | | **Columnas Estándar para Rubeola - 2do Archivo** | | case\_ID | | CaseNum | | OU\_Code | | dtReportedLocal | | dtInvestigation | | First Last Name | | Second Last Name | | First Name | | Sex Id | | Sex | | dtBirth | | AgeYearAtOnset | | AgeinmonthsAtOnset | | EpiWeekOnSet | | EpiYearOnSet | | EpiWeekNotification | | EpiYearNotification | | dtRashOnset | | SampleType | | TestType | | TestTypeName | | SampleID | | dtTaken | | LabCode | | dtSent | | dtReceived | | IDinLab | | Antigen | | result | | dtResult | | genotype | |

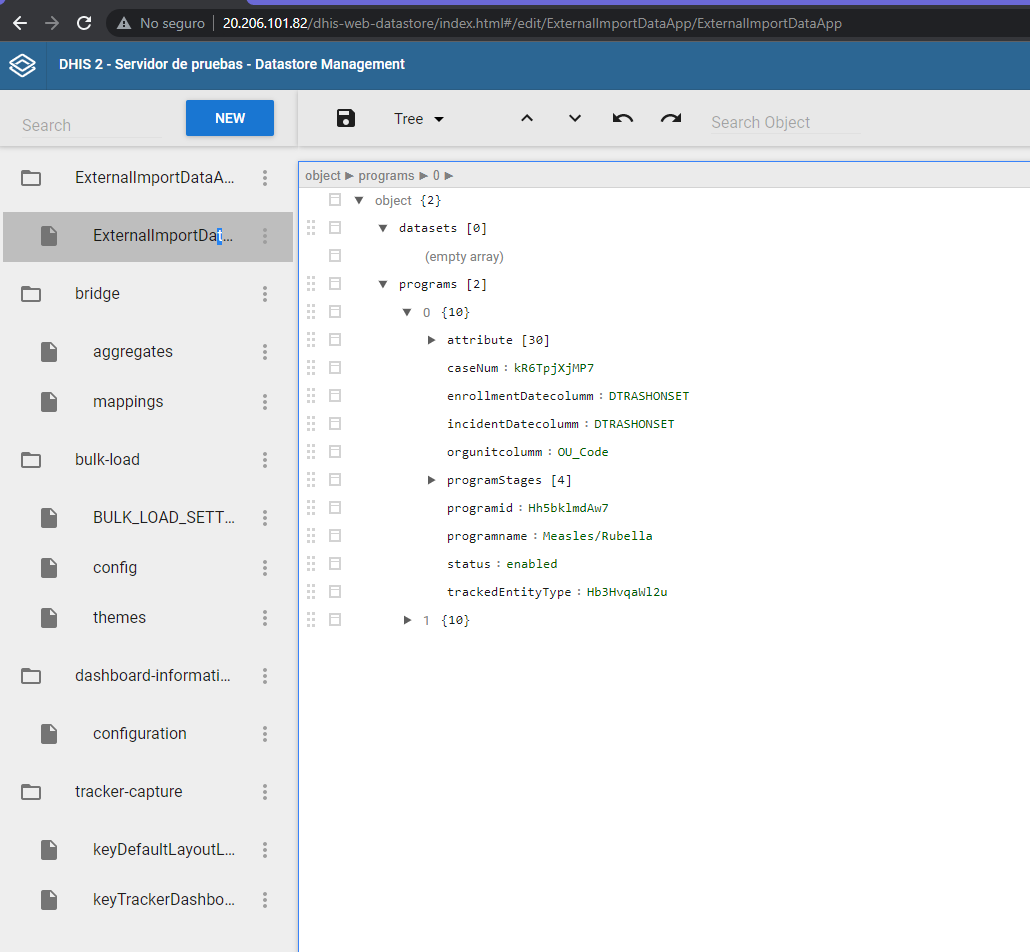
La plantilla del formato estándar lo pueden descargar en: **Enlace pendiente**

Esta configuración se tiene en cuenta en la pre-validación está en DHIS2, en la opción del menú de Datastore Managment, y ahí se puede encontrar todo lo relacionado con el mapeo para los diferentes programas.



Esta configuración se encuentra en formato JSON, aquí se encuentran estipuladas las columnas que deben coincidir con las de los archivos (CSV, XLSX o XLS), a que programa pertenece, los atributos y los elementos de datos, cada uno de estos tiene el tipo de dato que le corresponde que valores son permitidos en los mismos.

Cuando hace el proceso de pre-validación, lo primero que verifica es el formato del archivo y lo segundo con los tipos de datos de los atributos y los elementos de datos.



Ahora si se ejecuta, el proceso dentro de la aplicación VPD DataImport, se mostrará así:

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* + 1. **Paso 5: Carga**

En este paso donde finalmente se inicia todo el proceso de la importación de la data, pero este proceso tiene unas fases internas, para llevar a cabo todo este proceso.

1: Cargue del archivo

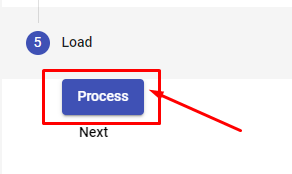
2: Borrado de events

3: Borrado de Enrollments

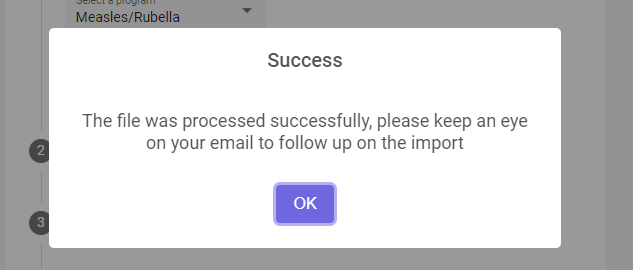
4: Importación de la data (tacked= registro de la persona Enrollment= registro de la persona al programa y Events= los datos del registro)

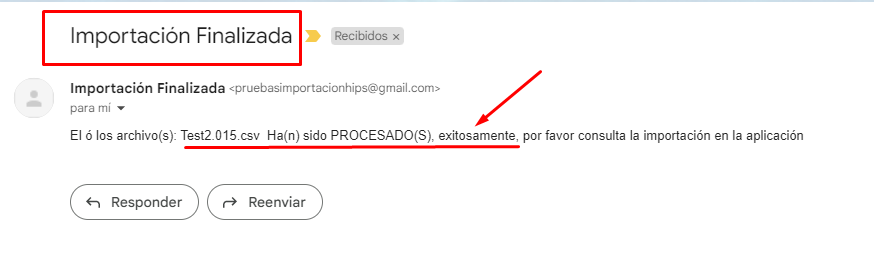
5: Guardado del resumen de la importación,

6: Notificación por email de la importación



Cuando se inicia el este proceso, puede tardar unos segundos al menos mientras valida el paso 1 (Cargue del archivo) y notificar que hubo algún error o el archivo se está procesando.

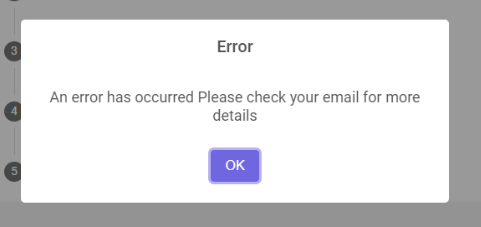




Para hacer el seguimiento a ambos casos, se debe estar pendiente del correo electrónico que se explicó al inicio de esta guía, ahí nos notificara en qué estado quedo el proceso o si se presentaron errores, nos detalla que sucedió.

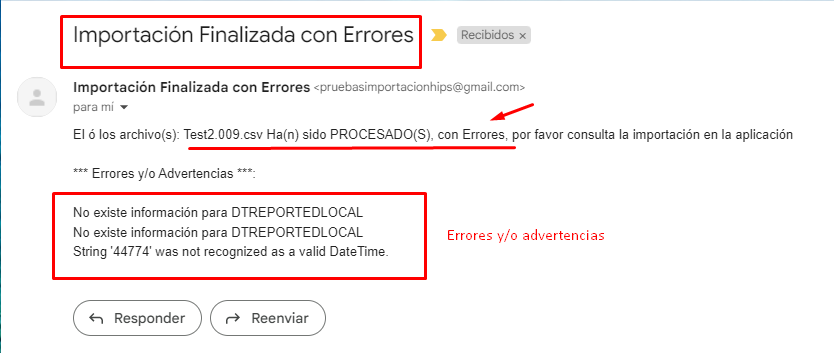
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



1. **History: Consulta del historial de las importaciones**

Para consultar el historial de las importaciones, se dirige a la opción de History en la parte superior de la pantalla:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

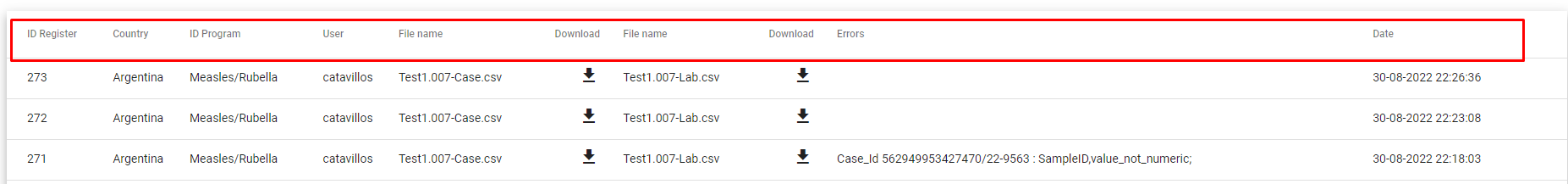
Al ingresar a la opción, esperamos a que cargue la data:

Patrón de fondo

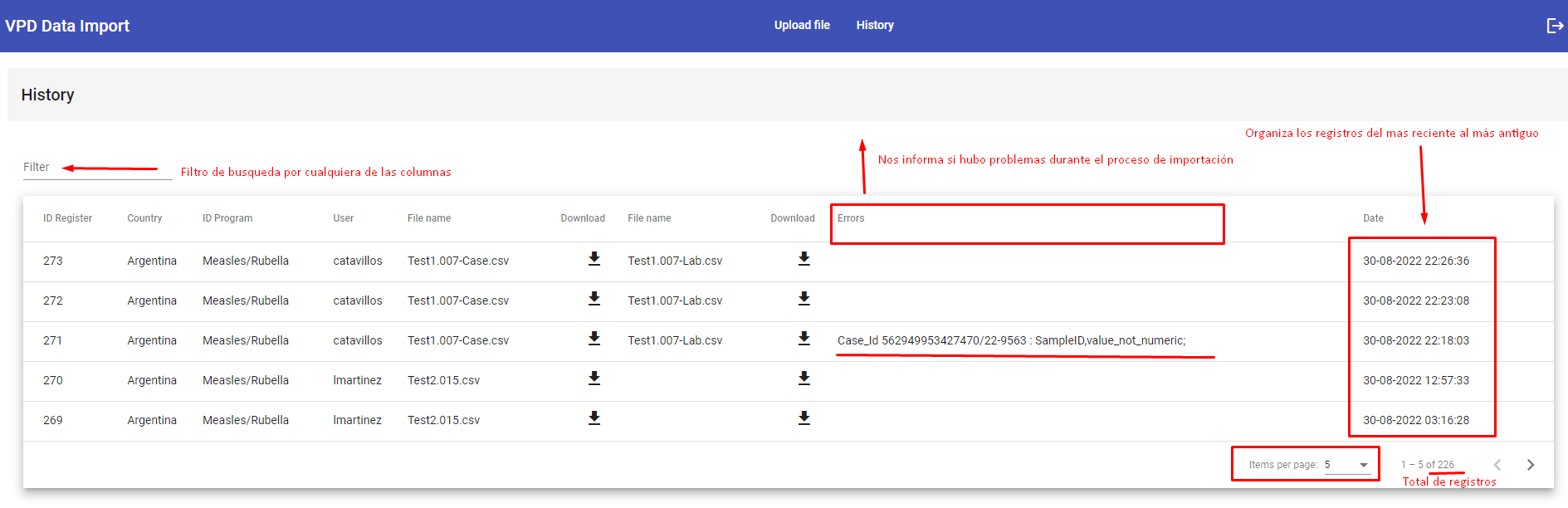
Descripción generada automáticamente con confianza media

Las columnas que se pueden consultar en este historial de importaciones son las siguientes:

1. ID Register
2. Country (País al cual se importó la data)
3. ID Program (Programa al cual se importó la data)
4. User (Usuario que subió el o los archivos con la data importada)
5. File name (Nombre del 1er archivo)
6. Download (Para descargar el 1er archivo)
7. File name (Nombre del 2do archivo)
8. Download (Para descargar el 2do archivo)
9. Erros (Errores en el proceso de importación)
10. Date (Fecha y hora de la importación)



Cuando carga la data, se puede visualizar los registros relacionados al historial de las diferentes importaciones, si es un usuario con rol de super usuario, puede visualizar todas las importaciones, incluyendo las de otros usuarios, de no tener ese rol (super - usuario) solo podrá ver el historial de importaciones del usuario con el que esta logueado en la aplicación.



El filtro funciona, registrando la búsqueda, de la siguiente manera (solo un ejemplo por usuario)

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

1. **Uso de los Endpoints: Swagger**

Para esta opción, se hace uso de Swagger, donde se tiene el acceso a los endpoints que tiene la aplicación, a continuación, como se visualizan:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para poder hacer uso de los endpoints, hay que tener en claro es que necesario tener el token (puesto que la mayoría de los métodos lo requiere), entonces se puede obtener, por esta misma opción de Swagger, hacia el final está la opción de login, y se da clic

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Y se despliega más información para ese endpoint, y se selecciona la opción ***Try it out***, para poder habilitar la edición de los parámetros y registrar la información para consumir el endpoint

Patrón de fondo

Descripción generada automáticamente

Para efectos de prueba, registro este usuario y contraseña que son válidos para la aplicación y posterior a esto, clic en execute:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Y así de esta manera se muestra la respuesta del endpoint, con los parámetros anteriormente registrados, hace la generación del TOKEN, el cual se genera por oauth2, por DHIS2 (con el cual se va a seguir trabajando para revisar los demás endpoints):

Interfaz de usuario gráfica, Texto, Aplicación

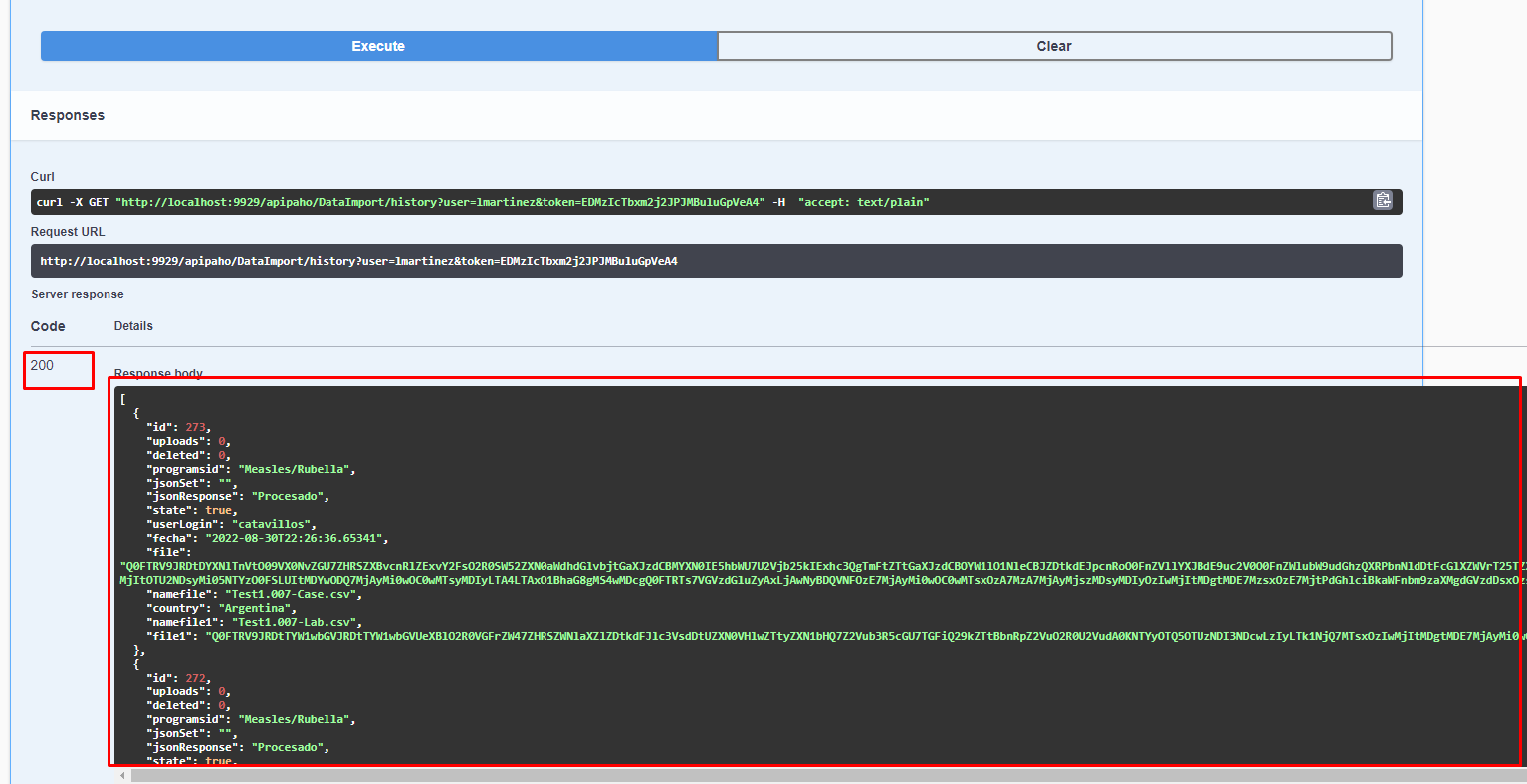
Descripción generada automáticamente

**Token generado para las siguientes pruebas:** EDMzIcTbxm2j2JPJMBuluGpVeA4

Del 1er apartado de endpoints, ya con el Token generado, se puede consultar el historial de las importaciones, así de esta manera:Interfaz de usuario gráfica, Aplicación

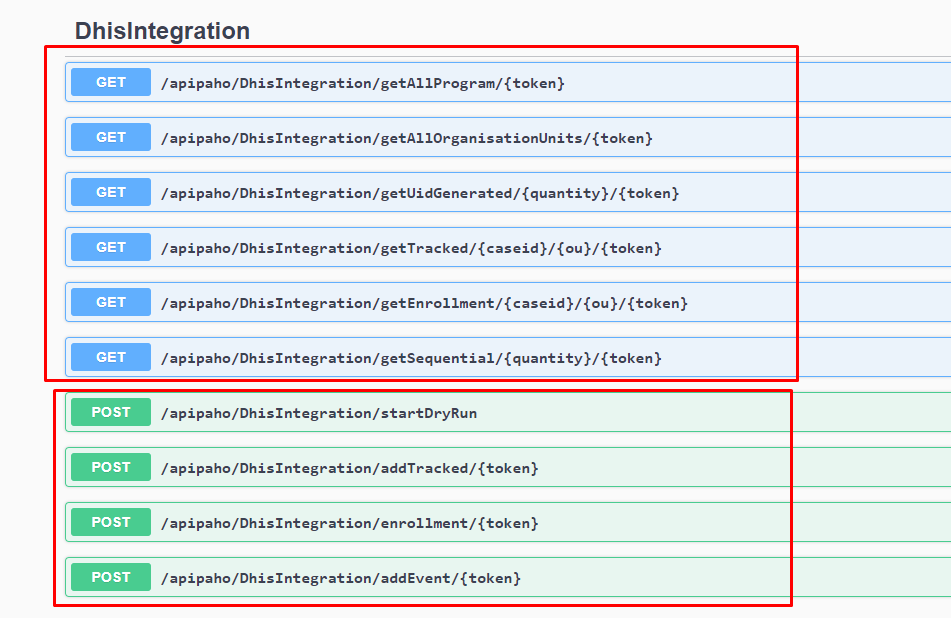
Descripción generada automáticamente

Aquí la respuesta:



En este apartado de Endpoints, tenemos varios métodos de tipo GET y POST, que se usan dentro de la aplicación, a continuación, los de tipo GET:

1. **GetAllProgram:** Método que permite traer todos los programas mapeados.
2. **GetAllOrganizationUnits:** Método que permite traer todas las unidades organizativas.
3. **GetUidGenerated:** Método que permite traer cual es uid del generado por el tracked.
4. **GetTracked:** Método que permite identificar si ya existe o no un tracked asociado cuando se está creando uno nuevo.
5. **GetEnrollment**: Método que permite traer los Enrollments.
6. **GetSequencial:** Método que genera la secuencias para el dato el campo number en DHIS2.



Aquí están los métodos POST:

{

"trackedEntityInstances": [

{

"trackedEntityInstance": "fqSPTQTAIUr",

"trackedEntityType": "Hb3HvqaWl2u",

"orgUnit": "WjvSpxMImAo",

"attributes": [{

"attribute": "WelU0kd868C",

"value": "4"

}, {

"attribute": "uiI2pR0ia32",

"value": "6"

}, {

"attribute": "kR6TpjXjMP7",

"value": "21-NVA00004"

}, {

"attribute": "ACeWiOj15cU",

"value": "2014-09-06"

}, {

"attribute": "P9D9Kno5h5M",

"value": "2021-01-03"

}, {

"attribute": "x48FoeDRMV8",

"value": "2"

}, {

"attribute": "BJOIp2AWzaD",

"value": "1"

}, {

"attribute": "ozzQiwrDfYC",

"value": "2021"

}, {

"attribute": "MPZ1QsldiKH",

"value": "2021"

}, {

"attribute": "cpu4Y9A1KZp",

"value": "2021-01-14"

}, {

"attribute": "I1bgG0PWXQw",

"value": "true"

}, {

"attribute": "VaG0ZzCCHsR",

"value": ""

}, {

"attribute": "gE9Z5IijE5J",

"value": "OLIVEROS "

}, {

"attribute": "sLfTm5aHAAy",

"value": "REBECA DEL VALLE"

}, {

"attribute": "zk2BaGzXyC4",

"value": "2"

}

]

}

}

]

1. **StartDryRun**: Método que se encarga de todo el proceso de la pre-validación del o los archivos a importar.
2. **AddTracked**: Método que se encarga de guardar toda la información relacionada con el tracked (registro de la persona), se puede hacer de manera individual o masiva (solo de los tracked o de la información completa (tracked, enrollments y events).
3. **Enrollment:** Método que guarda todo lo relacionado con el enrollment (que corresponde a enrolar a la persona o caso al programa) de manera individual o masiva.
4. **AddEvent:** Método que guarda toda la información relacionada los eventos de la persona o caso relacionado a las etapas (vacunación, investigación y laboratorio).

A continuación, algunos ejemplos de JSON para poder consumirlos:

|  |  |
| --- | --- |
| "enrollments": [{  "trackedEntityInstance": "QAeFDZvzzEW",  "program": "Hh5bklmdAw7",  "status": "ACTIVE",  "orgUnit": "ME59zsycp2B",  "enrollmentDate": "2019-02-18",  "incidentDate": "2019-02-18"  }, {  "trackedEntityInstance": "vPmYG6U2J9U",  "program": "Hh5bklmdAw7",  "status": "ACTIVE",  "orgUnit": "ME59zsycp2B",  "enrollmentDate": "2019-02-18",  "incidentDate": "2019-02-18"  }, {  "trackedEntityInstance": "E4HnH5meCIe",  "program": "Hh5bklmdAw7",  "status": "ACTIVE",  "orgUnit": "ME59zsycp2B",  "enrollmentDate": "2019-02-18",  "incidentDate": "2019-02-18"  }  ]  } | {  "events": [{  "events": "RBRYIay3lW1",  "trackedEntityInstance": "QAeFDZvzzEW",  "program": "Hh5bklmdAw7",  "orgUnit": "ME59zsycp2B",  "programStage": "sNQqHHN5gi3",  "eventDate": "2019-02-18",  "status": "COMPLETED",  "storedBy": "lcobo",  "coordinate": null,  "dataValues": [{  "dataElement": "RNaBky7SY1S",  "value": "1"  }, {  "dataElement": "M2VqHQyb2mY",  "value": "1"  }, {  "dataElement": "ugmyjBJvuSF",  "value": "2019-02-15"  }, {  "dataElement": "RtEQBiKrxAP",  "value": "2019-02-15"  }, {  "dataElement": "aufOFqfJej4",  "value": "2019-02-15"  }, {  "dataElement": "JLMjJQ4hyQQ",  "value": "1"  }, {  "dataElement": "PxyQ76H4gco",  "value": "0"  }, {  "dataElement": "yNpp8uXzAGP",  "value": ""  }, {  "dataElement": "CyZyXazs9OD",  "value": "RSF"  }, {  "dataElement": "UrnhPkMRx3H",  "value": "1"  }, {  "dataElement": "MmA8DCPxwLz",  "value": "2019-02-15"  }  ]  }  ]  } |

Para hacer el consumo de estos es muy similar que con los métodos GET, se debe tener en cuenta el token y los JSON: (para efectos de este ejemplo el anterior token expiro)

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Este método es asíncrono, y se crea una tarea, la opción de **relativeNotifierEndPoint**

Interfaz de usuario gráfica

Descripción generada automáticamente

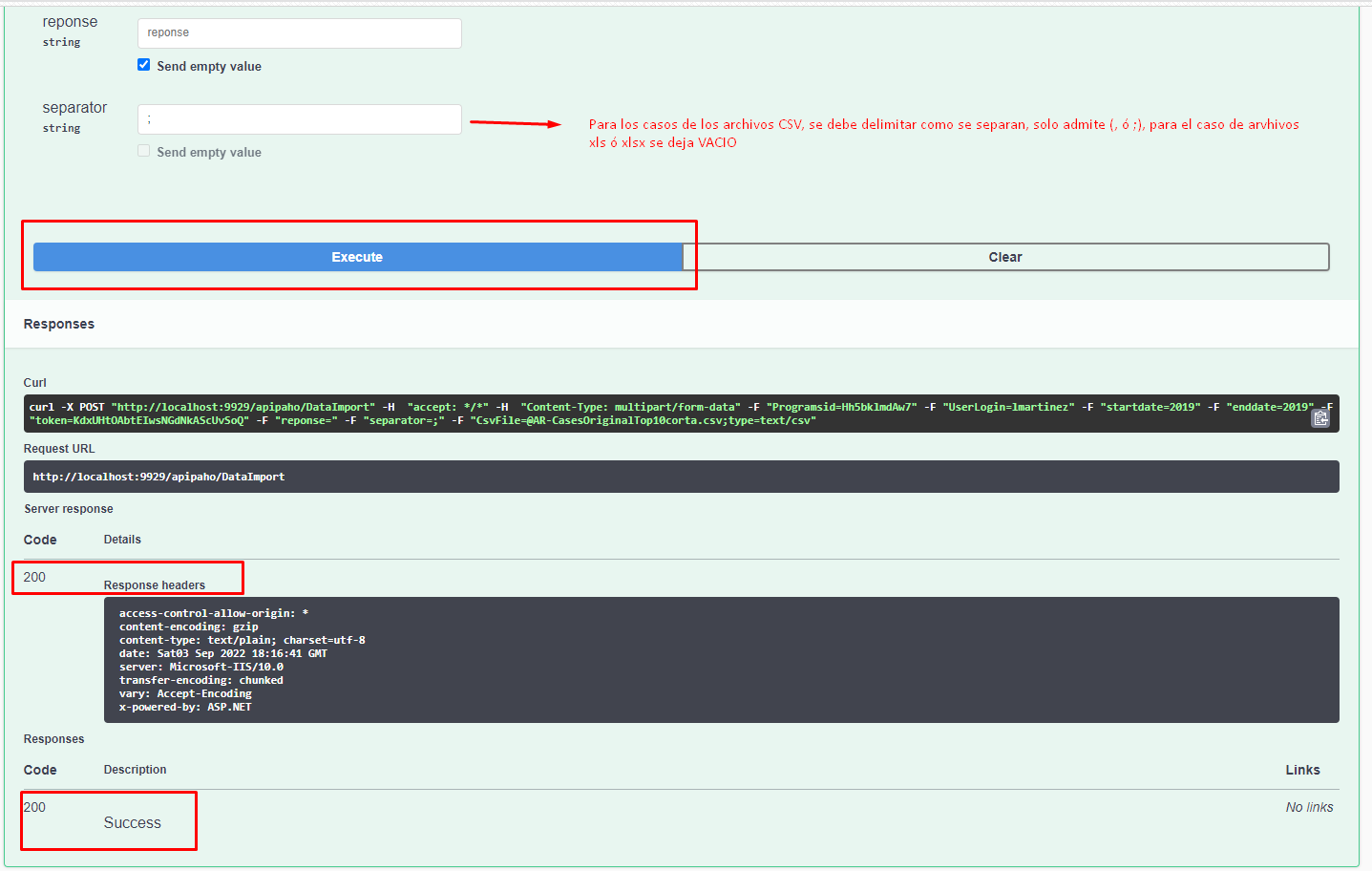
Dicha tarea se puede validar con postman, así:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

Descripción generada automáticamente

Imagen que contiene Gráfico

Descripción generada automáticamente



La información registrada en el archivo seleccionado está relacionada con esta unidad organizativa y aquí la evidencia que funciona correctamente:

Una captura de pantalla de una red social

Descripción generada automáticamente

También se puede, consumir este Endpoint desde postman, teniendo en cuenta los mismos parámetros y recomendaciones:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

También llegan los correos, del usuario con lo que ejecute el consumo del endpoint

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

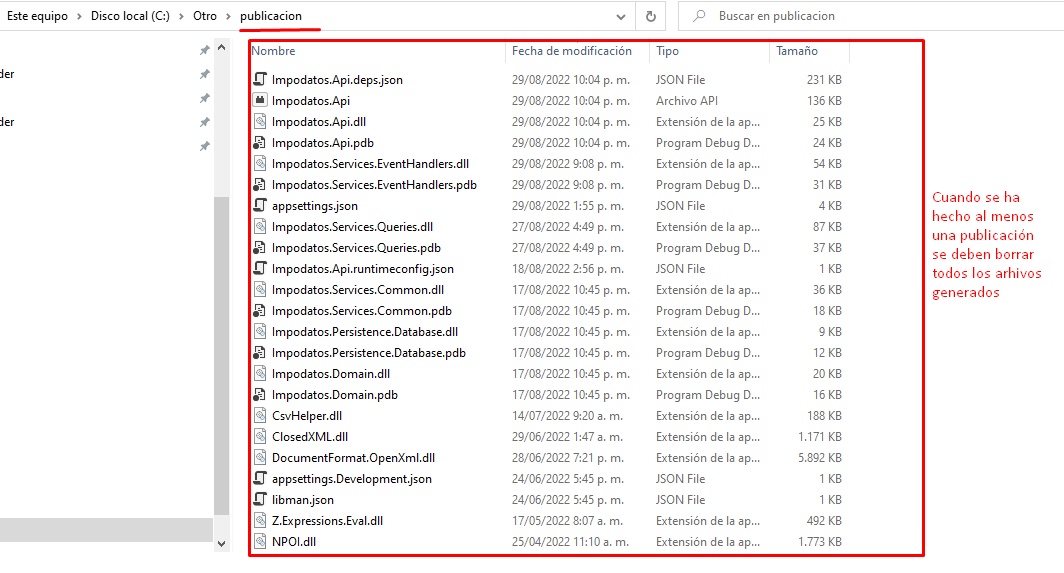
Descripción generada automáticamente

1. **Publicación**

Para esta parte de la publicación hay que tener en cuenta que el proyecto de VPD Data Import, cuenta con 2 proyectos uno para el front hecho en angular y la parte del back hecho en c#, para ambos proyectos se debe generar la publicación.

* 1. **Publicación en el Back**

Para el back, se recomienda crear una carpeta solo para este proceso y cada que se haga una publicación nueva borrar su contenido, debido a que en ocasiones los archivos NO se actualizan correctamente. Para este ejemplo en la unidad c, se ha creado una carpeta llamada publicación

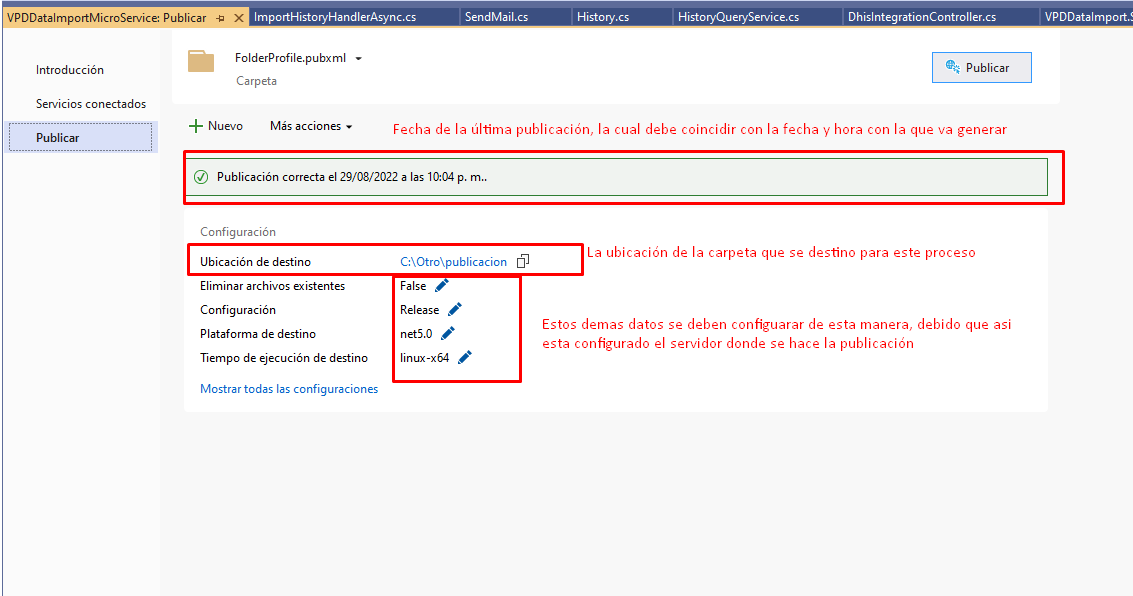


Ahora se dirige a la solución, con los todos cambios al día, es decir actualizado, y se genera la publicación, de esta manera:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Genera la siguiente pantalla dentro de la solución en visual studio



Una vez se tenga todo listo en la configuración de la publicación, se procede a publicar en el botón que tiene el mismo nombre, este proceso puede tardar unos minutos:

Interfaz de usuario gráfica, Aplicación, Teams

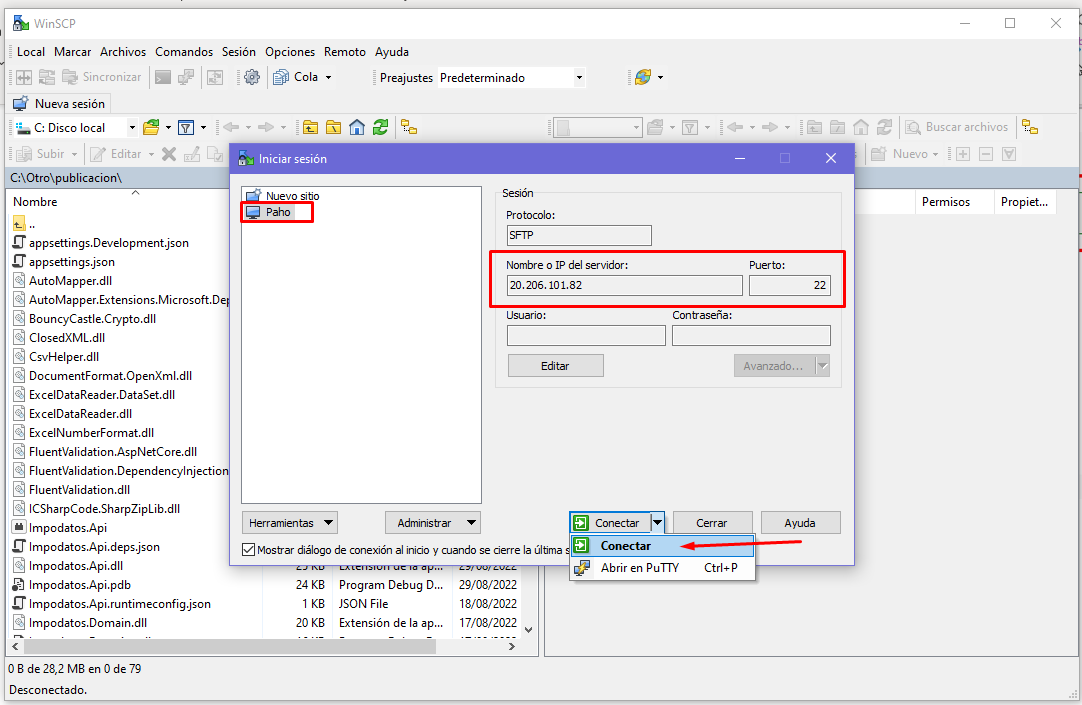
Descripción generada automáticamente

Una vez termine el proceso correctamente, nos debe quedar con la fecha hora actual, y correcta, así:

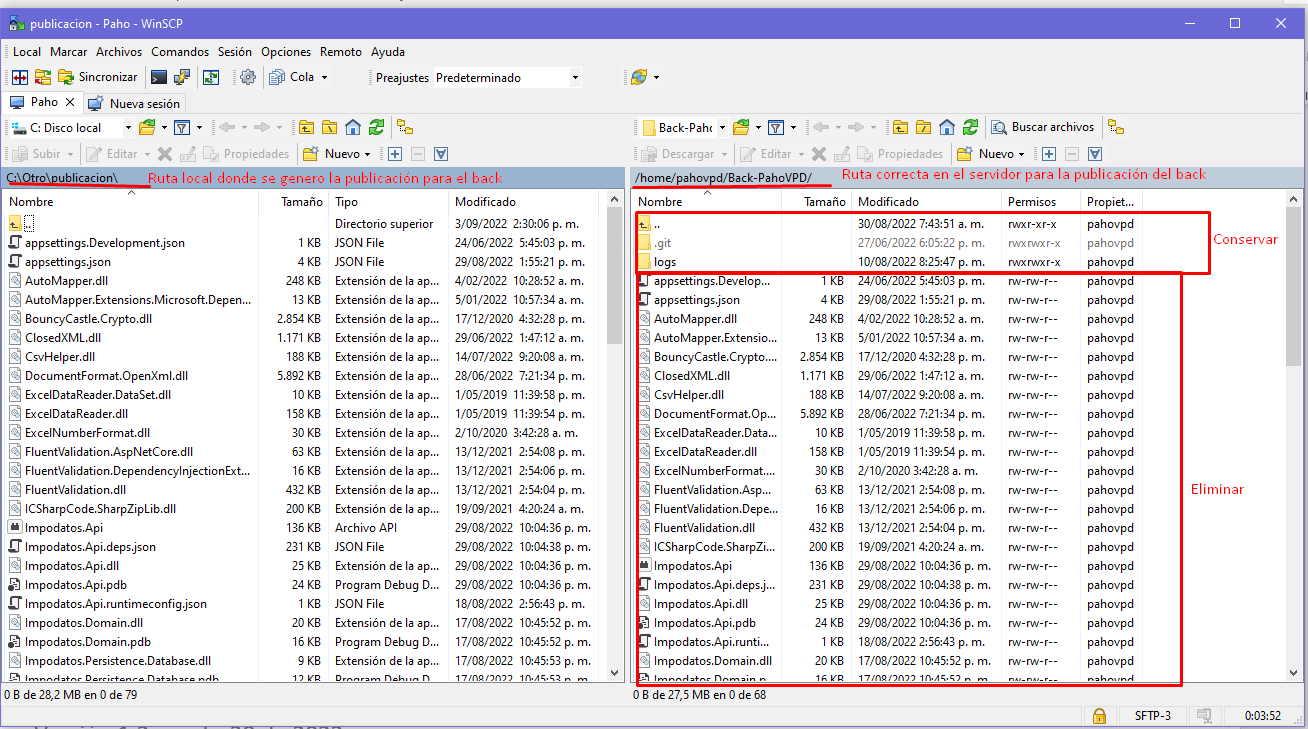
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Posterior a esto con la ayuda de la herramienta WinSCP, se hace el login para ingresar al servidor



Una vez este conectado, nos muestra 2 pantallas, la primera corresponde al equipo local donde se hizo la publicación y la segunda al servidor donde se deben transferir todos los archivos generados, para el back se tiene una carpeta individual solo para back.



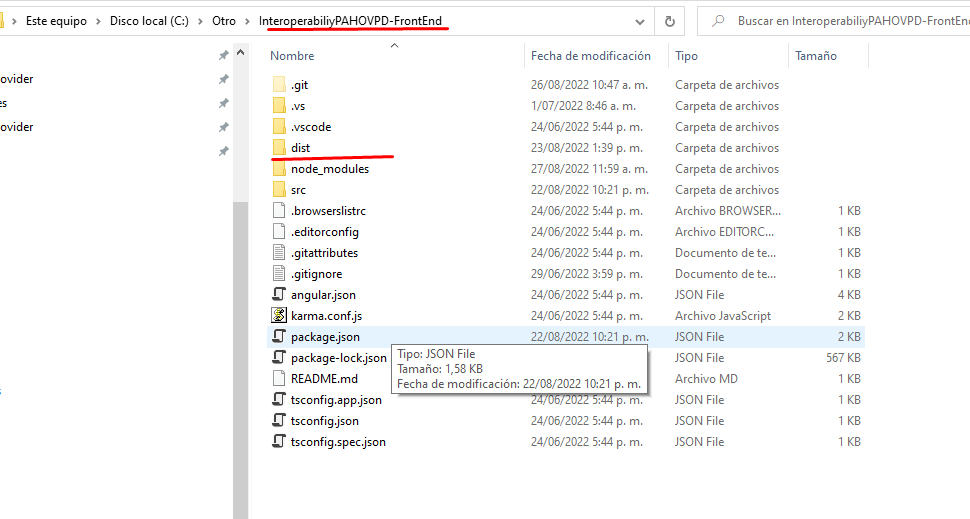
En caso de que, por equivocación, se borre la carpeta logs del servidor, se debe crear una carpeta con el mismo nombre y se deja vacía, ya que si no la encuentra genera conflicto.

Una vez se eliminen los registros, se selecciona de la 1era pantalla con ayuda de control + a, y se arrastran hacia la 2da pantalla, y se espera a que termine la transferencia, y así de esta manera se publica el back.

Pte imagen

* 1. **Publicación en el Front**

Para el front, se va a ubicación del proyecto local, para efectos de este ejemplo se encuentra en esta ubicación (la cual puede variar), la publicación o compilación del front, se genera dentro de la misma ubicación, en una carpeta llamada **dist**:



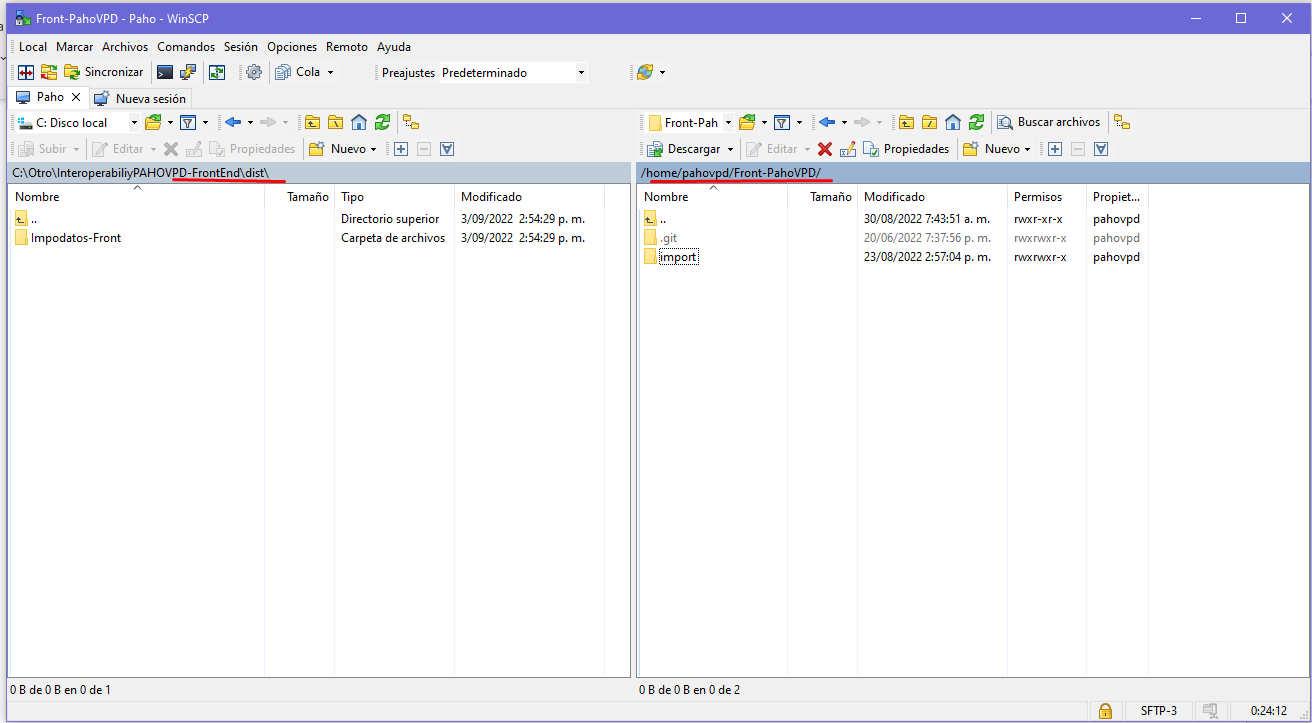
Para genera dicha publicación se debe ingresar por visual studio code y abrir el proyecto que corresponde al front, y se ejecuta el siguiente comando: **npm run buid**

Este proceso al igual que el del back, puede tardar unos minutos

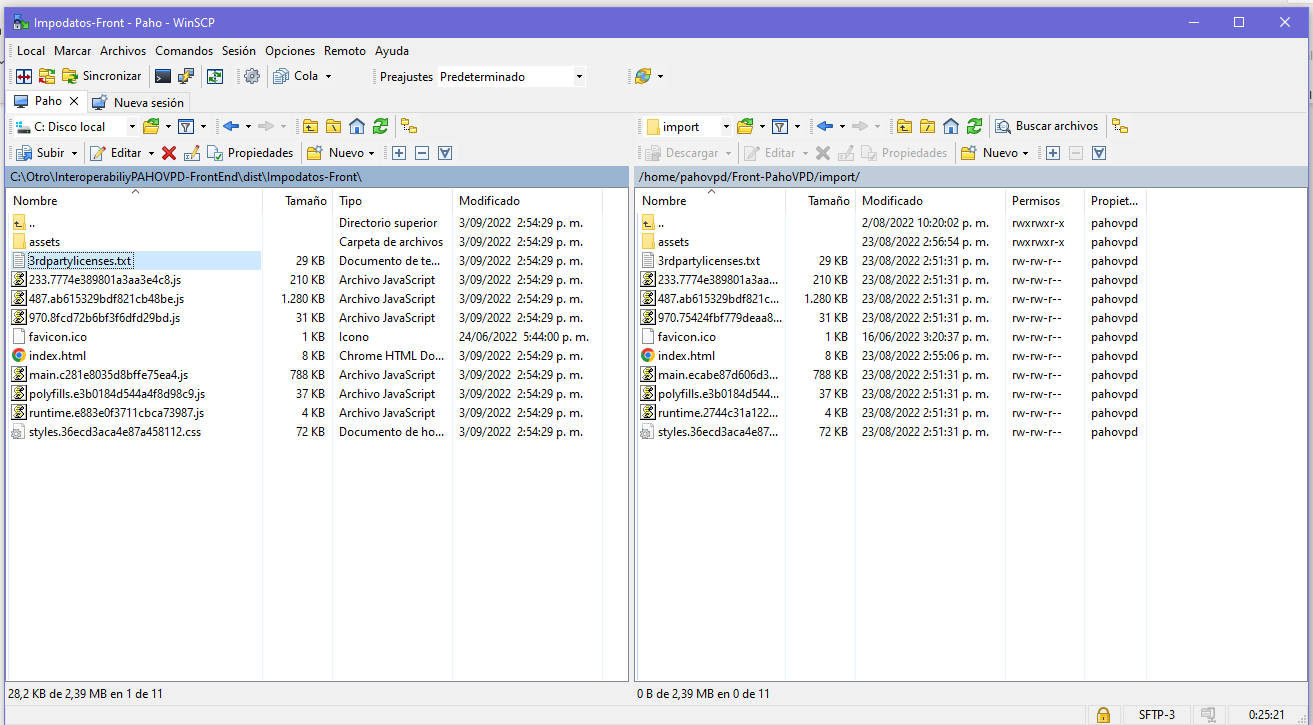
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Una vez termine el proceso, es muy similar al anterior, con la ayuda de WinSCP, se hace el login para ingresar al servidor, y se ubica de forma local donde esta el proyecto del front, en la carpeta dist y en el servidor en la carpeta de front:



Aquí también se debe borrar el contenido de la carpeta import del servidor y el contenido de la capeta local transferirlo (se seleccionan y se arrastran hacia la carpeta del servidor):



1. **Código fuente, Arquitectura, patrones y flujo de trabajo de la aplicación**

El proyecto de integración de datos VDP a DHIS2 tiene un diseño inicial como el presentado a continuación.

Diagrama, Esquemático

Descripción generada automáticamente

En este diseño se observa como los componentes actuales utilizados en PAHO interactúan a través del ETL para consumir/enviar datos a DHIS2. Para el caso de la importación de datos a través de archivos estándar se hace con el nuevo servicio construido y que corresponde a los componentes presentados a continuación.

Diagrama

Descripción generada automáticamente

El componente Import RawData, corresponde al servicio construido para importar datos desde archivos estándar, este servicio expone una REST API que puede ser consumida por otros servicios como el ETL (SSIS), también el ETL podría consumir directamente la API de DHIS2 en caso que lo requiera.

Como recomendación se describen dos componentes handler apipaho y apiDHIS2, estos serían eventualmente requeridos para comunicarse con las respectivas API.

* 1. Patrón arquitectónico CQRS.

Para el desarrollo del servicio se utilizó .netcore 5 y se adoptó el patrón arquitectónico CQRS([CQRS pattern - Azure Architecture Center | Microsoft Learn](https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs)), es un patrón que separa las operaciones de lectura y actualización de un almacén de datos, con el fin de no compartir los modelos que habitualmente se utilizan en operaciones básicas CRUD. En el proyecto desarrollado se evidenció que los modelos requeridos en los archivos estándar son distintos a los requeridos por DHIS2, por tanto, este patrón permitió manejar distintos modelos y manejarlos en la capa requerida.

El proyecto de desarrollo se distribuye en carpetas de acuerdo con las responsabilidades dentro del patrón adoptado, en la imagen a continuación se puede observar la distribución del proyecto.

|  |  |
| --- | --- |
| Texto  Descripción generada automáticamente | * VPDDataImportMicroservices   Contiene todos los controladores que exponen en rutas a los servicios Rest construidos,como ejemplo en Endpoint DataImport expuesto en el controlller DataImportController.   * VPDDataImport.Services.Commons   Contiene el mecanismo los clases que se encargar de hacer las peticiones Request y los DTOs que definen el modelo de respuesta de los endpoints. Consultados.   * VPDDataImport.Services.EventHandlers   Contiene los comandos de creación, actualización, eliminación requeridos en la solución. En este proyecto se maneja toda la lógica principal de la aplicación de importación.   * VPDDataImport.Services.Queries   Contiene todos los métodos de consulta que retorna los endpoint de consulta e incluye los DTO que se manejan en las respuestas, la lógica de armar el historial de importación se encuentra aquí.   * VPDDataImport.Persistence.Database   Contiene el acceso a la BD donde se almacena el historial incluyendo el modelo manejado para su creación.   * VPDDataImport.Domain     Contiene los modelos utilizados para manejar las respuestas de los EndPoint consumidos de DHIS2. |

* 1. Archivo de configuración

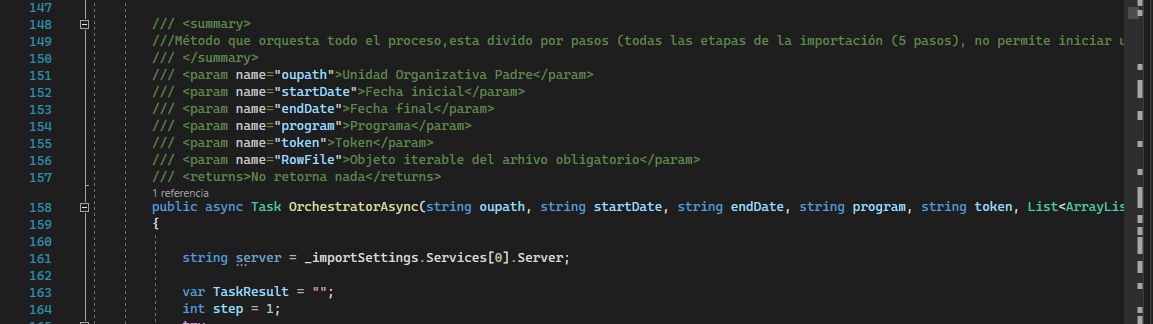
En esta carpeta del proyecto VPDDataImportMicroService se encuentra el archivo de configuración appsetting.json en el cual se puede configurar el tamaño del bloque de importación, la estrategia de importación, los mensajes que se enviarán como notificación, url servidor DHIS2 entre otras configuraciones.

* ConnectionStrings : Configuración de acceso a la base de datos
* Services.url : dirección url de la Api de DHIS2
* Services.url.Authentication: se configura el token de acceso externo a DHIS2 para uso de Oauth2
* ImportSettings : Configuración del servidor de correo, mensajes de respuesta a notificaciones, tamaño del bloque de importación y estrategia de importación etc.
  + async: importación síncrona o asíncrona, poner atención ya que es un cambio crítico, se recomienda dejar en true.
  + sizeUpload: el tamaño del bloque de importación se recomienda dejar en 100
  + individual: determina si se desea que la importación se realice de uno en uno, se recomienda dejar en false
  + server: dirección del servidor de correo smtp
  + emailFrom: usuario del servidor de correo smpt
  + pass: contraseña del usuario del servidor de correo smpt
  + port: puerto del servidor de correo smpt

las otras key corresponden al mensaje de respuesta.

* 1. Patrón orquestador.

Para el adecuado manejo de la asincronidad y la espera de respuestas de alta duración asociadas a la comunicación entre el servicio y DHIS2, se implementó el patrón de comportamiento orquestador ([Patrones de orquestación: Aplicaciones sin servidor | Microsoft Learn](https://learn.microsoft.com/es-es/dotnet/architecture/serverless/orchestration-patterns)) el cual permite ejecutar con base en los resultados de cada método la ejecución de un paso siguiente. La lógica implementada está construida en la clase ImportHistoyHandlerAsync separada en 5 pasos y además fue necesario programación concurrente para evitar bloqueos del servidor.



* 1. El código fuente se encuentra publicado en:
* [HISPColombia/InteroperabiliyPAHOVPD-BackEnd (github.com)](https://github.com/HISPColombia/InteroperabiliyPAHOVPD-BackEnd)
* [HISPColombia/InteroperabiliyPAHOVPD-FrontEnd (github.com)](https://github.com/HISPColombia/InteroperabiliyPAHOVPD-FrontEnd)

El código fuente se encuentra documentado en cada uno de los métodos desarrollados.

* 1. Componentesdel código fuente y flujo de trabajo

La solución está distribuido en dos componentes generales, que son el microfrontend que se presenta como aplicación web y el microservicio creado en c# para el backend. El microservicio consume los endpoint del servicio REST utilizando la misma autenticación de DHIS2.

Por otro lado, el microservicio está constituido por varios proyectos como se ilustra a continuación.

Diagrama

Descripción generada automáticamente

Y la forma en que interactúan está definido por la orquestación creada para importar hacia DHIS2. A continuación se entrega el diagrama de secuencia indicando como interactúan las clases entre proyectos y sus relaciones que corresponden a los métodos. Además, el componente UI (microfrontend) que consume los endpoint reflejados en métodos del microservicio.

Diagrama

Descripción generada automáticamente

Adjunto a este documento se entrega el diseño de la solución hecha en Visual paradigm para ver detalladamente el diagrama presentado.