MANUAL TÉCNICO

API DE MIGRACIÓN DE LA RED HIVEBRITE

DIRECCION DE TECNOLOGIA Y SISTEMAS DE INFORMACION

KEVYN SEBASTIAN MORENO GOMEZ

2024

CONTENIDO

[INTRODUCCION 3](#_Toc184975766)

[ESTRUCTURA GENERAL DE LA API 3](#_Toc184975767)

[BASE DE DATOS 4](#_Toc184975768)

[CODIGO PHP 5](#_Toc184975769)

[Carpeta Config 6](#_Toc184975770)

[Carpeta html\public 6](#_Toc184975771)

[Carpeta src 7](#_Toc184975772)

[POSTMAN 12](#_Toc184975773)

[ERRORES 16](#_Toc184975774)

[RECOMENDACIONES 18](#_Toc184975775)

# INTRODUCCION

En este documento se presenta una guía técnica para correcta ejecución del software de migración para los datos que están presentes desde los endpoints de la red de egresados “Hivebrite” hacia una base de datos MySQL alojada en un servidor de ambiente de pruebas de la DTSI con la dirección IP 10.0.20.189, usando el puerto 3306 para conectarse al entorno MySQL y el servidor de pruebas localhost:82 para conectarse al servidor de apache.

A continuación, se va a profundizar en cada apartado que conforman el ecosistema de la API de migración, las particularidades de cada área y, por último, una serie de errores y recomendaciones que se deben tener presente al momento de ejecutar este software.

# ESTRUCTURA GENERAL DE LA API

La API de migración se compone de la interacción de 3 entornos distribuidos en las siguientes herramientas de trabajo: Visual Studio Code, Postman y MySQL Workbench

En Visual Studio Code se desarrolló toda la lógica detrás de la migración de los datos, gracias al trabajo realizado, este se encarga de extraer la información, clasificarla según su contenido e insertarla a la base de datos.

Postman se utilizó para el consumo de las peticiones tanto de consulta de la información como para realizar la inserción de los datos y era quien ejecutaba el código escrito en php.

Finalmente, para almacenar la información se opto por crear una base de datos en MySQL usando Workbench como editor de esta ya que el volumen de los datos se acomodaba mejor a este motor de base de datos.

A continuación, se profundizará más en cada uno de estos entornos de trabajo que conforman la API.

# BASE DE DATOS

Empezando por la base de datos llamada” bdegresados3”, esta fue creada específicamente para albergar los datos presentes en los 12 endpoints suministrados que conforman la red de egresados, tras un análisis previo, la información se distribuyó de la siguiente forma:

Cuenta con 38 tablas las cuales se dividen en 21 tablas independientes, 13 tablas dependientes y 4 tablas intermedias.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

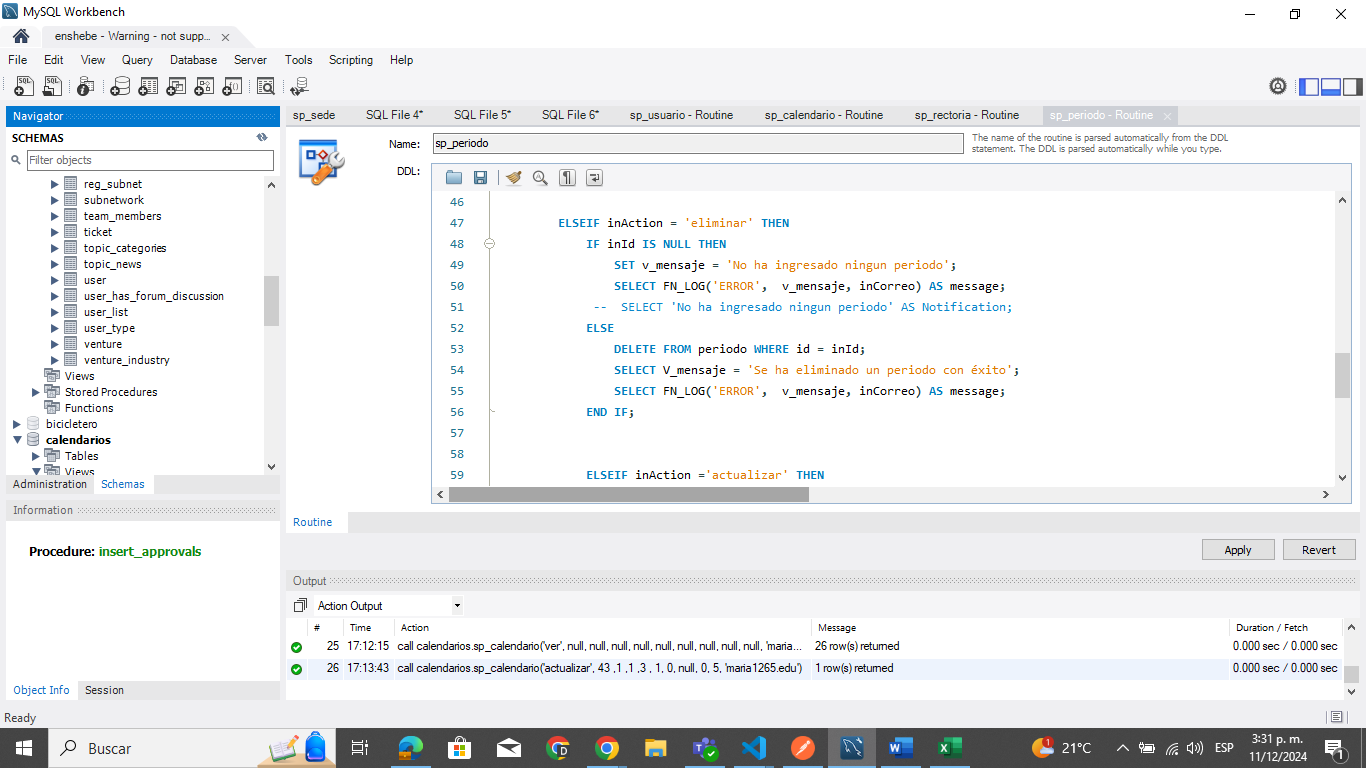
Descripción generada automáticamente

Imagen 1. Las 38 tablas que conforman la base de datos en el servidor de pruebas

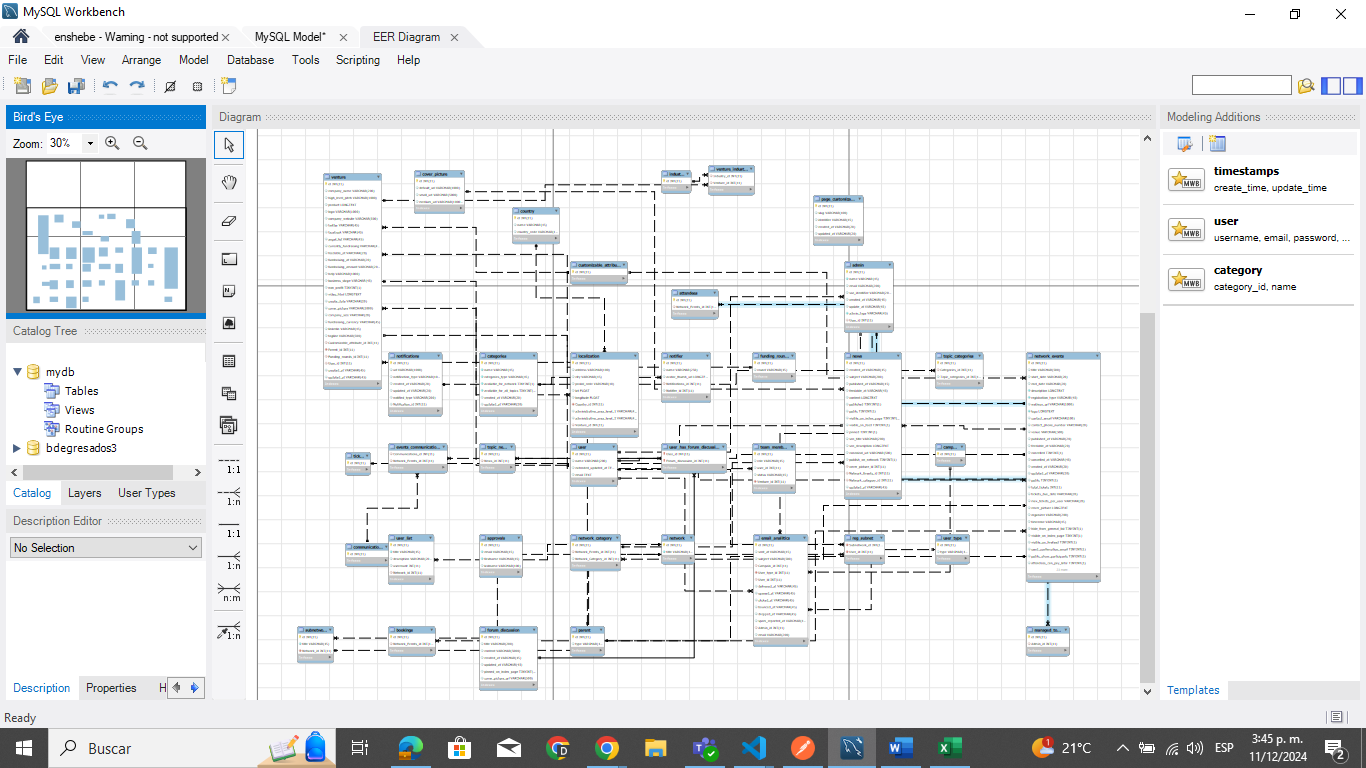


Imagen 2. Diagrama MER de la base de datos

# CODIGO PHP

Pasando al código que conforma el software, como se mencionó antes, este fue desarrollado en el editor de Visual Studio Code, usando el lenguaje de programación php (versión 8.2.12). La carpeta llamada APIEGRESADOS se conforma principalmente de 3 carpetas:

IMPORTANTE: Para que el código y el postman funcionen hay que levantar el servidor 82 con el comando php -S 0.0.0.0:82 -t . \html\public en la terminal de comandos, donde el ultimo directorio es en donde estará el archivo index.php

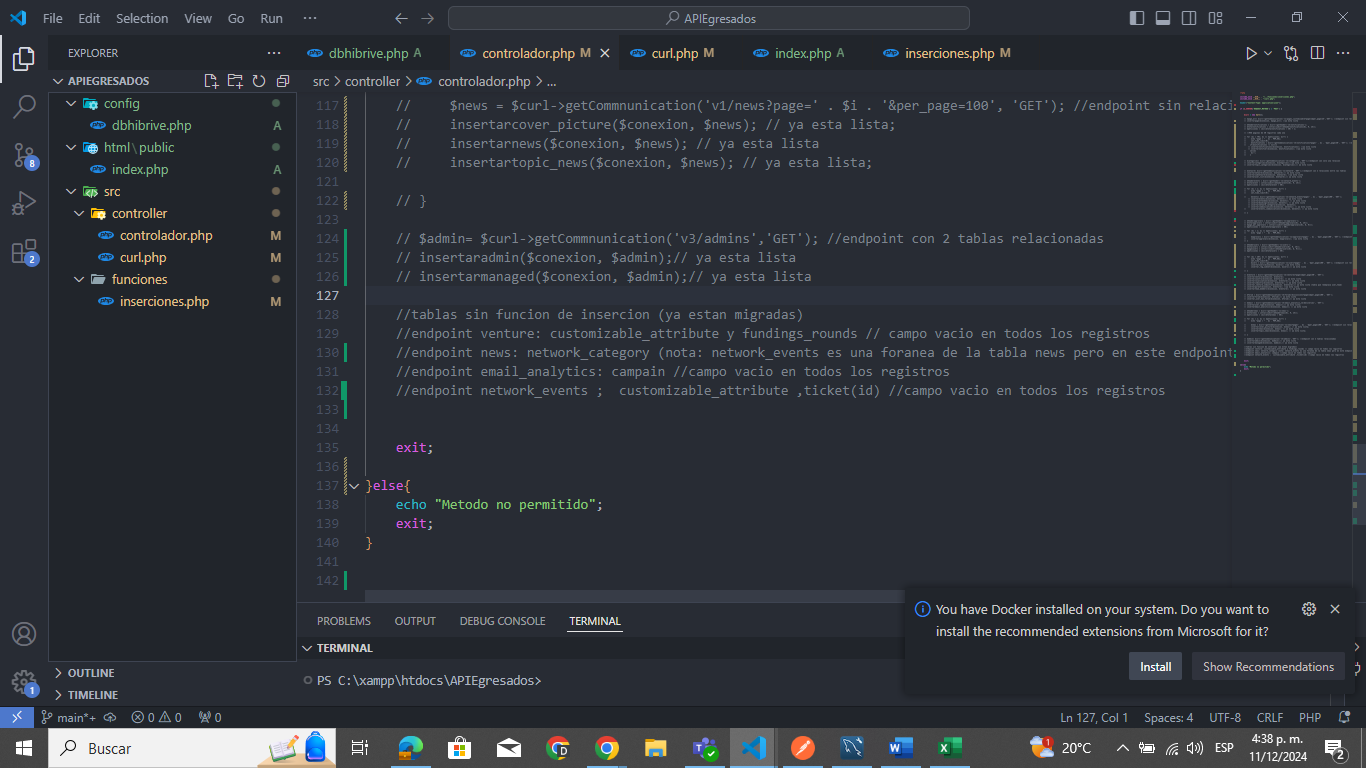


Imagen 3. Contenido de la carpeta fuente del proyecto APIEgresados

Carpeta Config

**Archivo dbhivebrite.php:** Este es el archivo donde están los parámetros de conexión al servidor 10.0.20.189 en donde se encuentra alojada la base de datos usando la librería MySQL para conectarse con la base de datos.

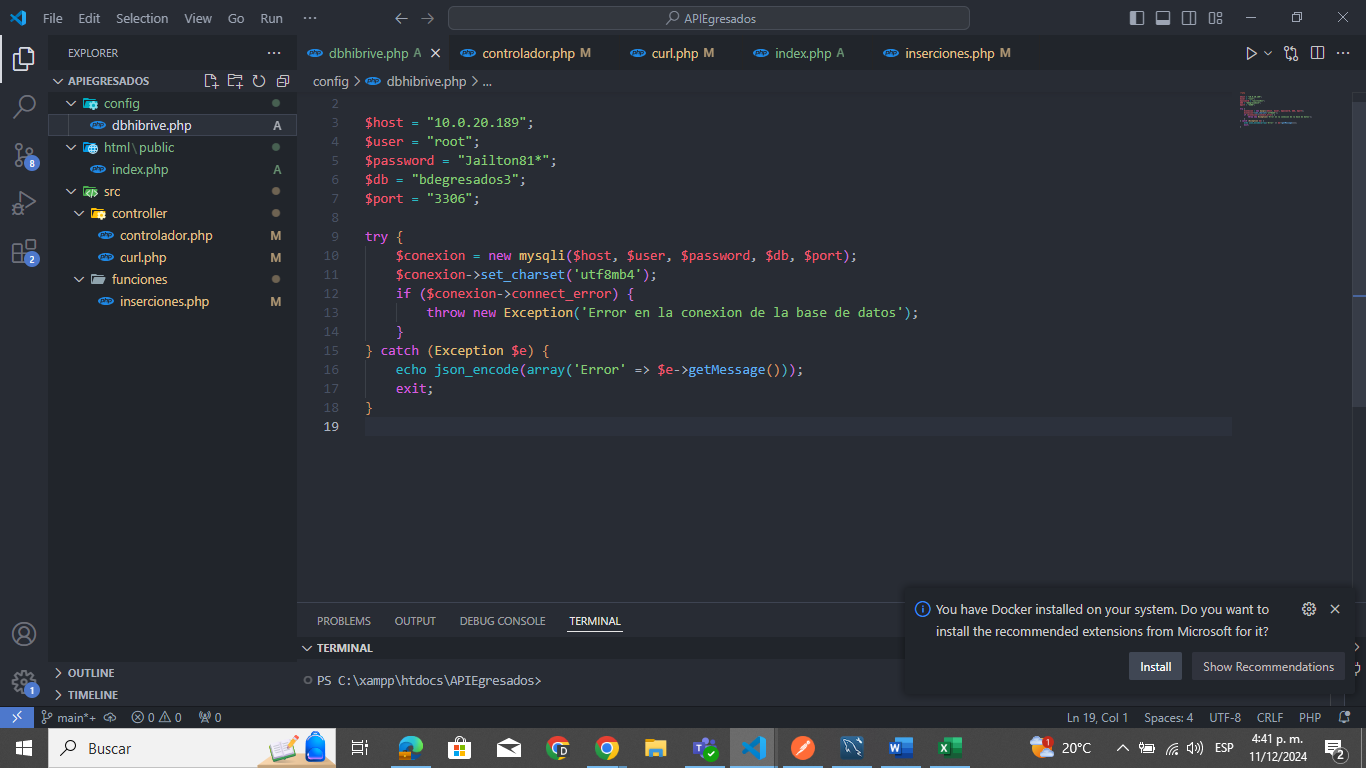


Imagen 4. Script del archivo dbhivebrite.php

Carpeta html\public

**Archivo index.php:** Este archivo alojado dentro de la carpeta public, que a su vez está alojado en la carpeta html; Es el archivo que postman usa para acceder a la url del proyecto, este archivo redirecciona los ficheros dbhivebrite.php (conexión a la db) y a controlador.php (administra las peticiones).

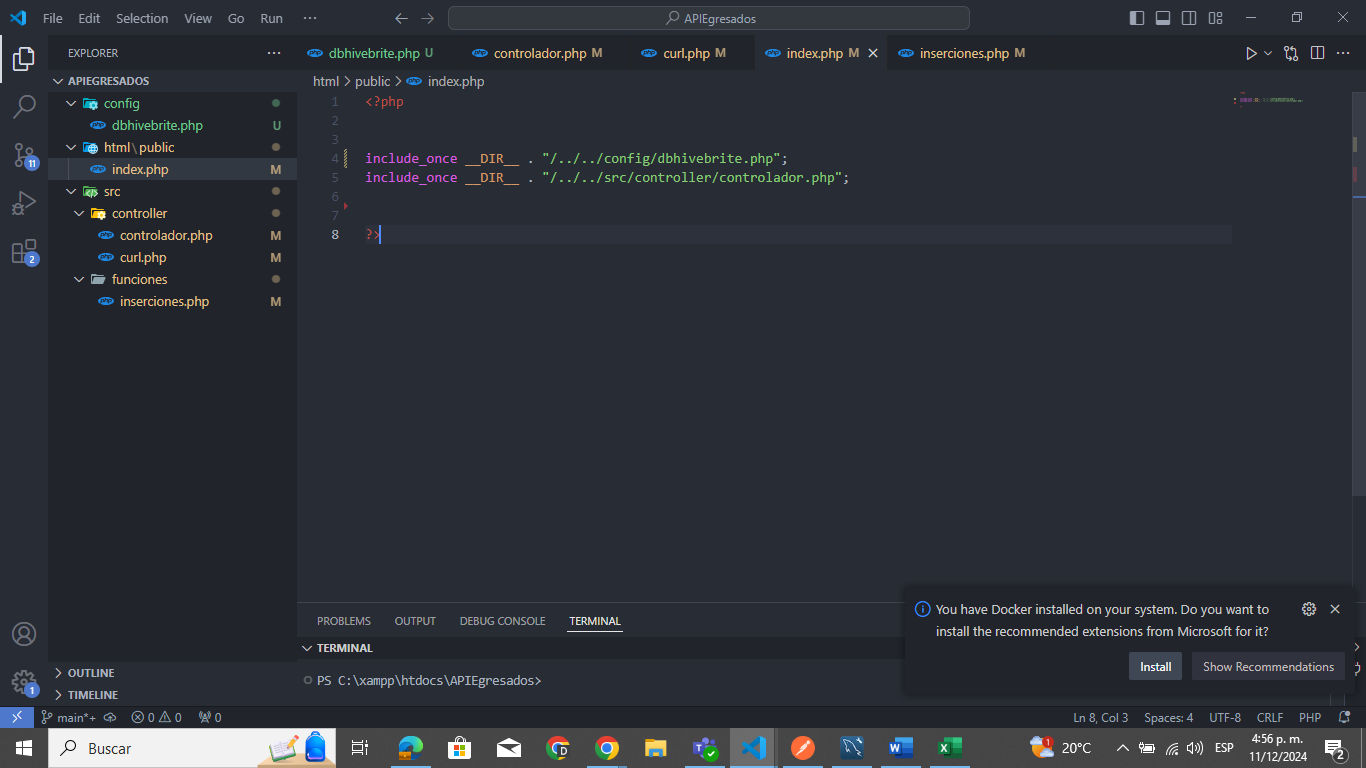


Imagen 5. Script del archivo index.php

Carpeta src

**Src/controller:** En esta carpeta se encuentran los archivos controlador.php y curl.php

**Controlador.php:** En este archivo se encuentra la administración de las funciones responsables de insertar los datos en cada tabla, cada grupo de funciones esta agrupada según al endpoint en donde está alojada la información, por ejemplo, el endpoint de venture, contiene la información con la que se van a llenar 7 tablas de la base de datos.

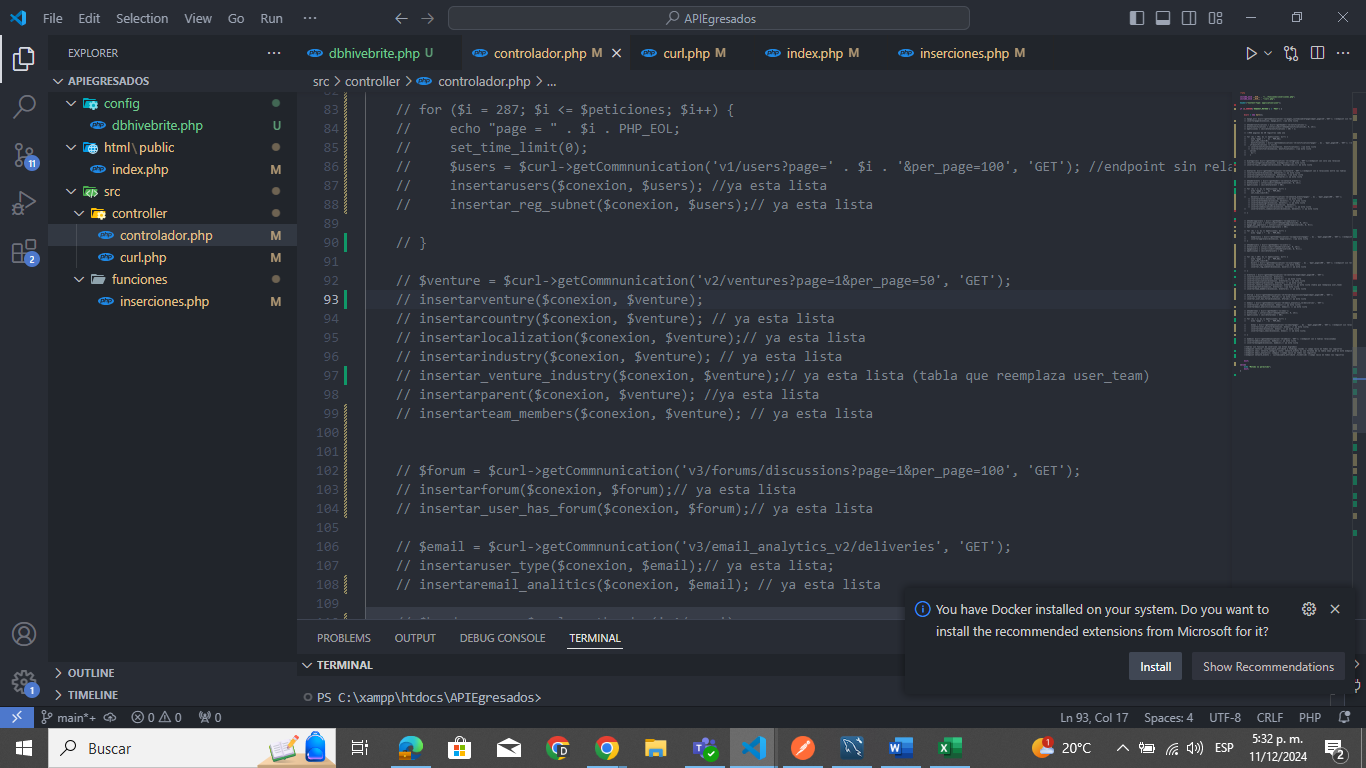


Imagen 6. Grupo de funciones del endpoint de venture

También existen un grupo de funciones en donde para poder insertarlas se hace uso de un bucle “For”, debido a que los endpoint están configurados para mostrar una cantidad de datos por página (más información en el apartado de postman), entonces obteniendo el número total de registros divido el número máximo de registros mostrados por página, se obtiene el iterador del bucle, el cual inserta los datos según los registros existentes presentes en cada página(iteración), si al pasar a la siguiente pagina no hay mas registros o no existen registros la inserción finaliza a menos de que se presente el error de la petición vacía(más información en el apartado de errores).

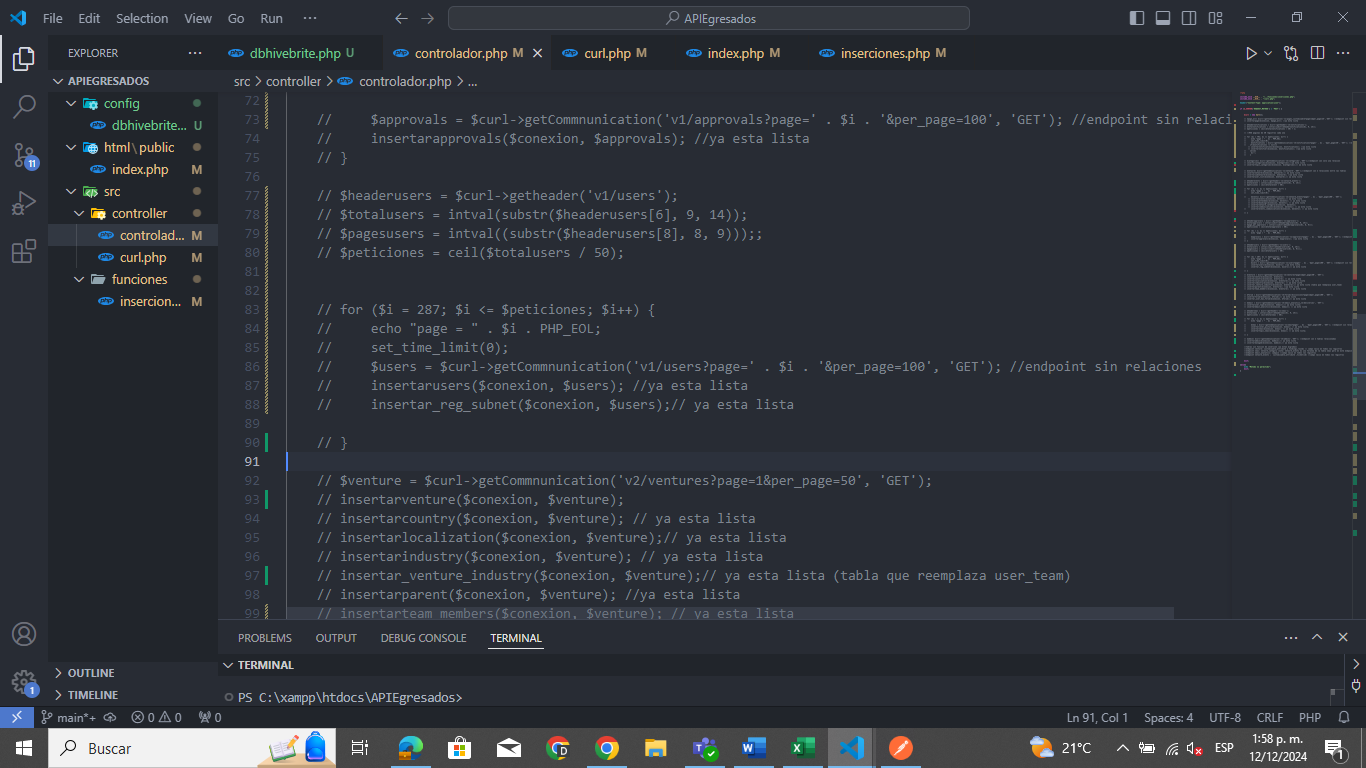
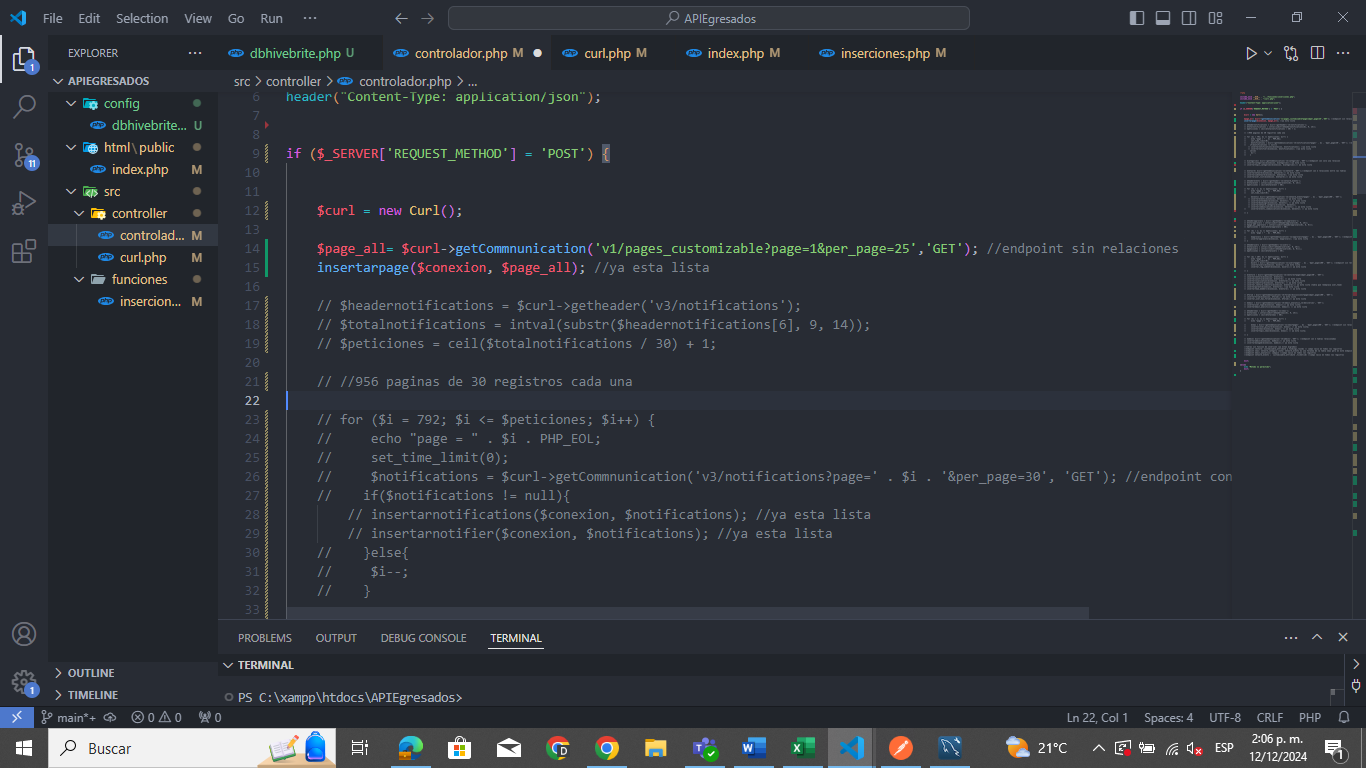


Imagen7. Bucle para insertar las tablas users y reg\_subnet

La forma de habilitar la inserción de una o varias tablas es simplemente descomentando la instancia de la clase curl del endpoint que contiene la información junto con la función a la que hace referencia a la tabla a la que se le va a insertar información, asegurándose de que dicha tabla este vacía.



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Además de ello existen algunas tablas cuya información en el endpoint está vacía, por ello se optó por no realizar una función de inserción, en donde dichas tablas están comentadas en la parte de abajo del script.

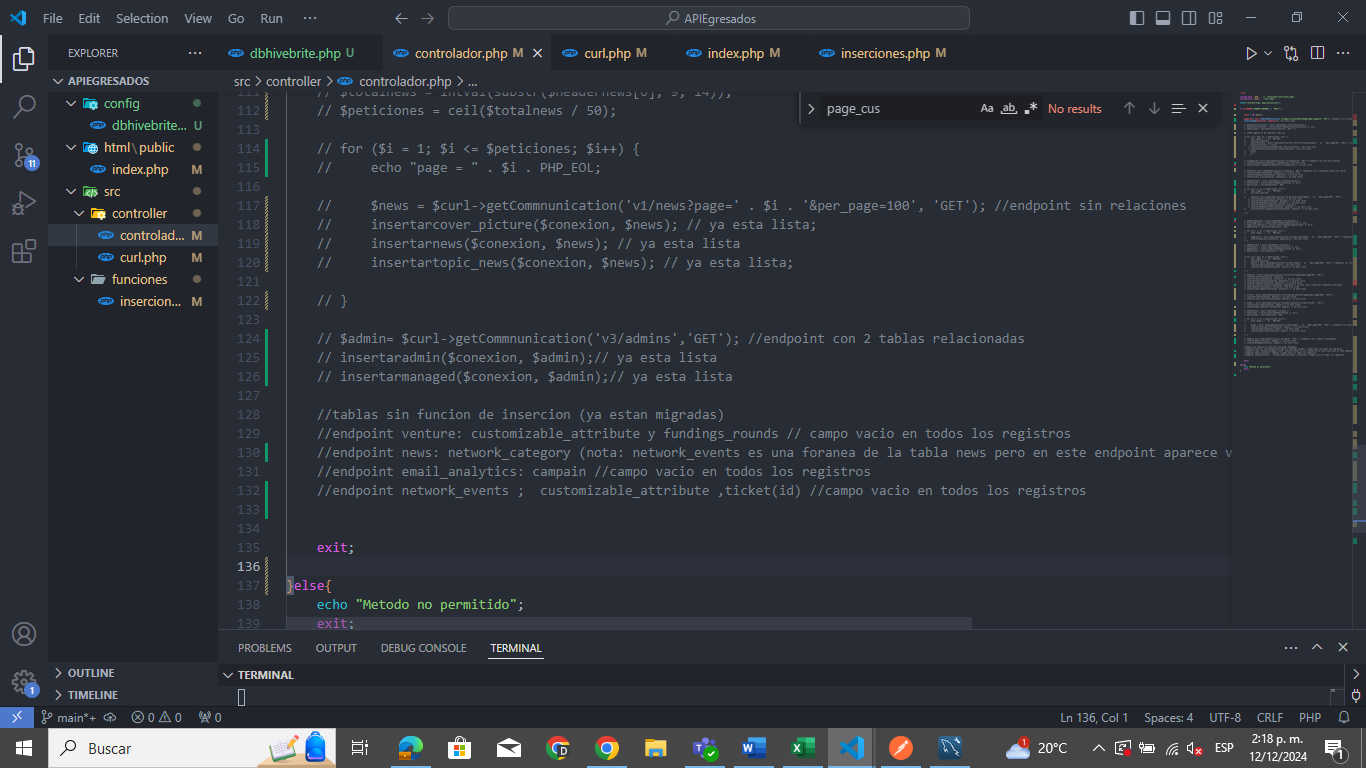


Imagen 8. Tablas que están vacías en los registros de los endpoints

Por último, el orden de cada grupo de funciones de un endpoint está distribuido jerárquicamente de la siguiente forma:

Captura de pantalla de computadora

Descripción generada automáticamente

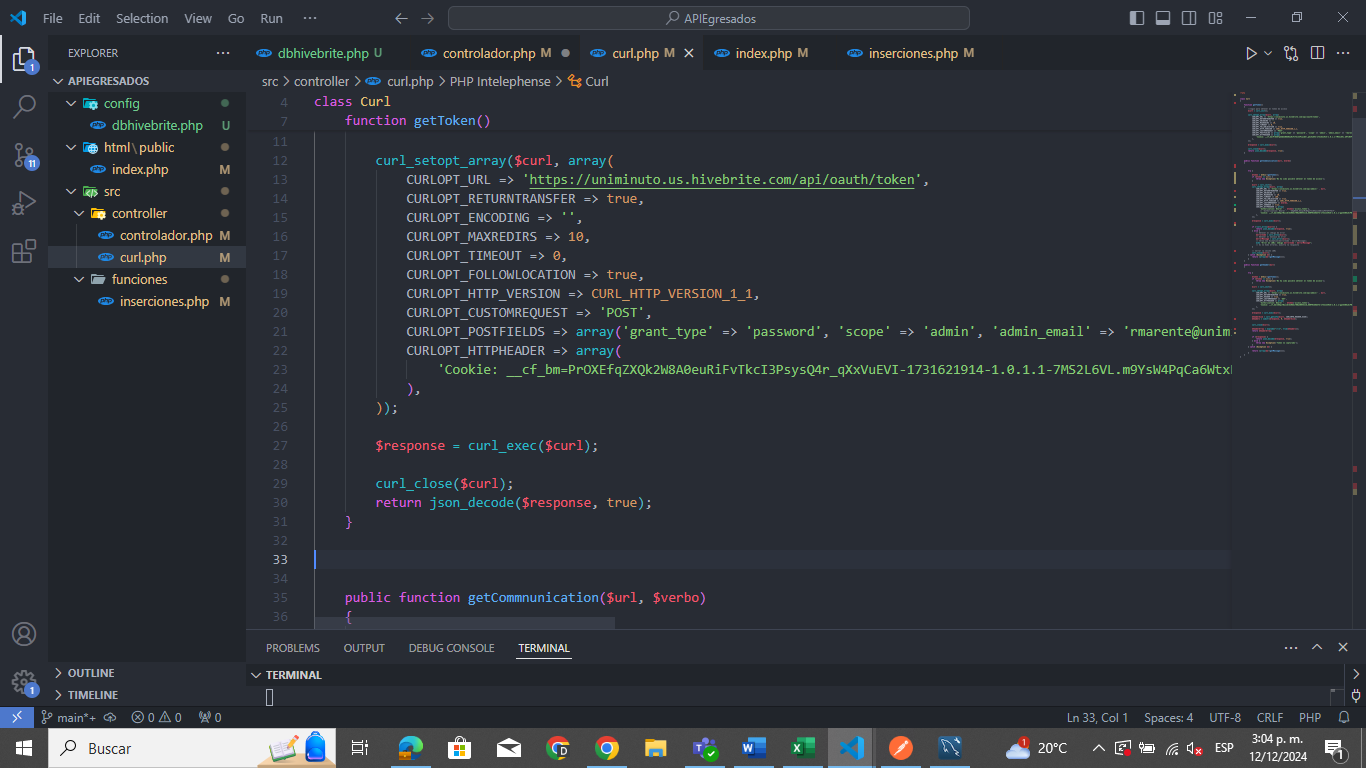
1. variable que almacena toda la información del endpoint (para este ejemplo el nombre de la variable será venture), usando la instancia de la clase curl que utiliza como parámetros la parte de final de la url del endpoint (versionendpoint/nombredelendpoint?page=N&per\_page=N) y el tipo de petición que se realiza al consultar la información (GET).
2. llamado de la función de la tabla a la cual se va a insertar, usando como parámetros la variable global declarada arriba junto con la variable conexión de dbhivebrite.php (tabla independiente)
3. llamado de la función de la tabla a la cual se va a insertar, usando como parámetros la variable global declarada arriba junto con la variable conexión de dbhivebrite.php (tabla dependiente).
4. llamado de la función de la tabla a la cual se va a insertar, usando como parámetros la variable global declarada arriba junto con la variable conexión de dbhivebrite.php (tabla intermedia).

Nota 1: Solo acepta solicitudes POST, si se pide otro tipo de solicitud no funcionara (más información en el apartado de errores)

Nota 2: En algunos endpoints, debido a la cantidad masiva de registros se superaba el tiempo estándar máximo de ejecución (120 segundos) por lo que se opto por configurar el bucle con un tiempo de ejecución 0 para que pudiera insertar todos los registros sin tener que preocuparse por el tiempo de ejecución (más información en el apartado de postman) .

**Curl.php:** Este archivo se compone de 3 funciones las cuales se encargan de consultar la información de los endpoints de hivebrite para poder almacenar la información en un array de datos.

La primera función se llama getToken,esta se encarga de realizar una solicitud post al servidor de hivebrite usando como parámetros un perfil de administrador, para que la respuesta sea un token de acceso, sin el token de acceso la información de los demás endpoints no puede ser consultada.



La segunda función se llama getCommunication y es la mas importante de las 3, debido a que esta es la que consulta la información almacenada en cada endpoint de tipo GET de hivebrite y la convierte en un gran array asociativo. Para operar correctamente usa el token que se obtiene con la función getToken como uno de sus parámetros para realizar la petición, aunque alternativamente se puede copiar y pegar directamente el token que responde el endpoint login de hivebrite en postman, aunque como este token no es una variable dinámica, caducará dentro de 7 dias a partir del momento en que se haya hecho la solicitud post. Esta función es llamada en controlador.php para que por medio de los parámetros url y verbo, almacene el array asociativo en una variable global del endpoint en cuestión.

Captura de pantalla de computadora

Descripción generada automáticamente

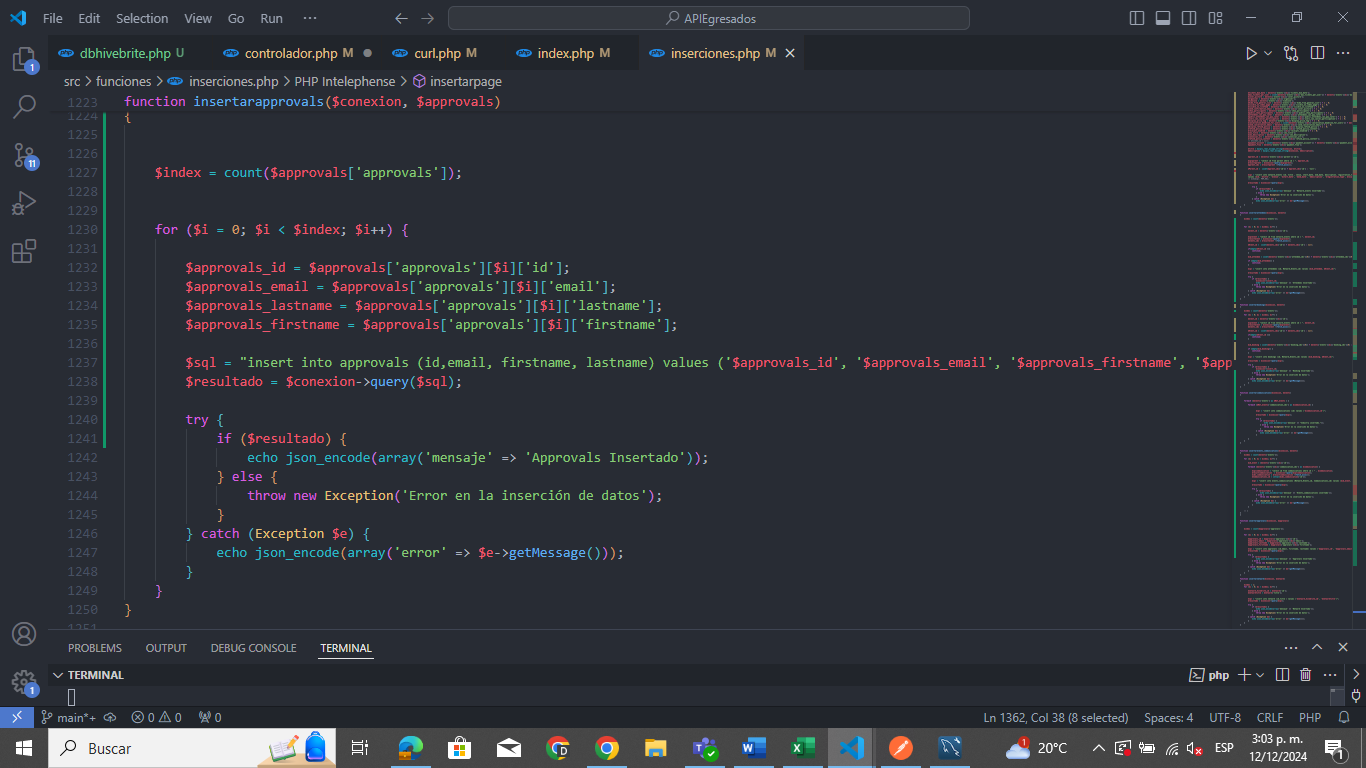
Por último, está la función getHeader, la cual se encarga de usar la misma estructura de petición que getCommunication solo que esta vez guarda los headers del endpoint indicado por el parámetro de la url, en donde en controlador.php, se extrae la variable que contiene el numero total de registros, la cual es necesaria para cuando se requiere realizar una inserción masiva de registros.

Captura de pantalla de computadora

Descripción generada automáticamente

Src/funciones: En esta carpeta se encuentra el archivo inserciones.php

Inserciones.php: Este archivo contiene todas las funciones que se encargan de clasificar, separar, organizar e insertar la información que proviene de la variable que contiene getCommunication de controlador.php, la cual usan como parámetro, junto con la variable conexión de dbhivebrite.php. Cada tabla de la base de datos que contenga información tiene una función correspondiente, son en total 35 funciones las cuales cada una esta configurada para insertar los datos según las características propias de cada tabla, y el manejo de llaves foráneas.



# POSTMAN

En postman es en donde se consultan los endpoint de hivebrite usando un protocolo curl. Los endpoints de hivebrite están divididos de la siguiente forma: 12 endpoint de tipo GET en donde se encuentra almacenada la información de la red hivebrite y 1 endpoint de tipo POST “login” el cual es el que se consulta al momento de obtener el token (más información en código php – curl.php), una vez se consume el endpoint login, no se necesita consumirlo nuevamente para acceder a la información de los demás endpoints, salvo que el token expire, sea invalido o haya una petición vacía (mas información en el apartado de errores).

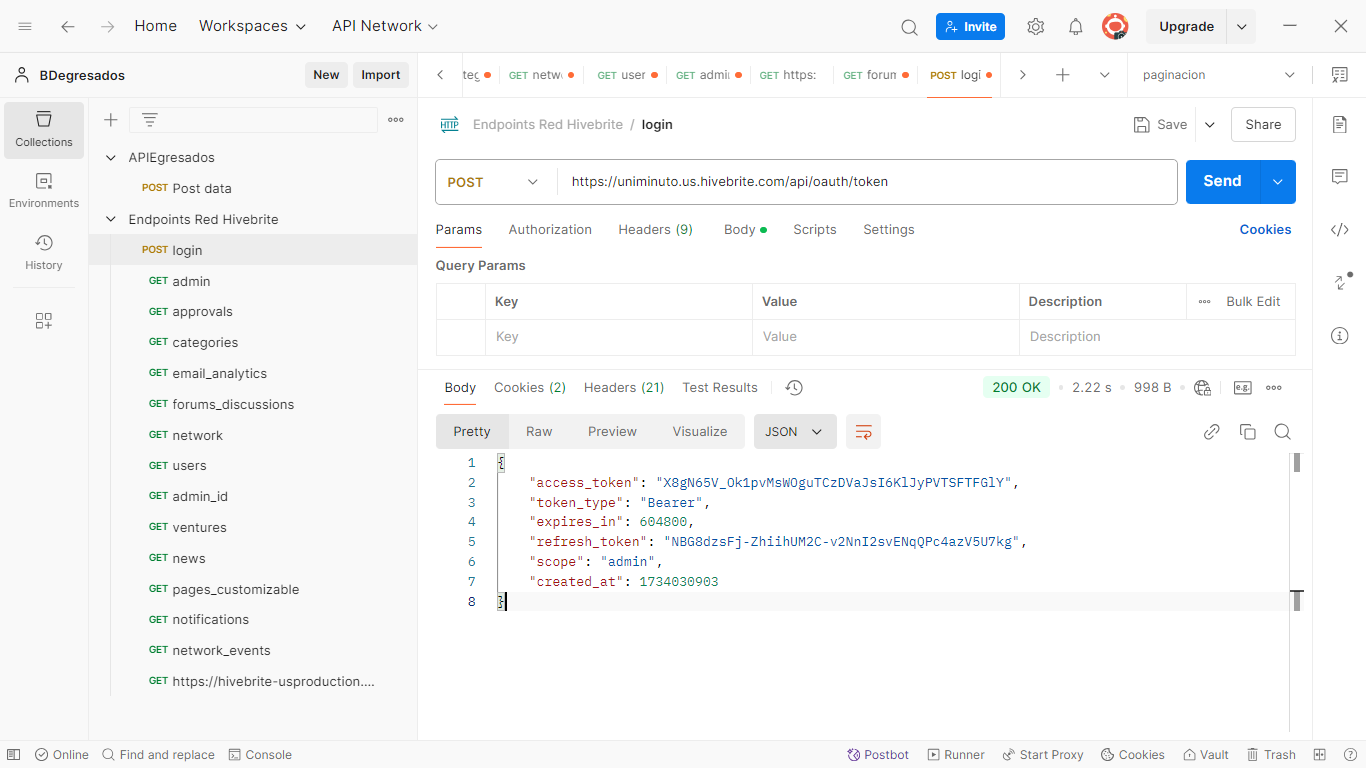


Imagen 9. Endpoints del proyecto, para consultar la información de hivebrite primero se consume el endpoint de login

Por otro lado para ejecutar la inserción de los datos en la base de datos existe un endpoint de tipo post el cual usa la url <http://localhost:82> para conectarse tanto al servidor apache, como al puerto MySQL, simplemente al oprimir “send” , se comienza a consultar la información, a procesarla y a insertarla en su respectiva tabla, dependiendo del endpoint, la cantidad de páginas, la conexión a internet y la cantidad de tablas insertadas en la misma petición, el tiempo puede variar significativamente.

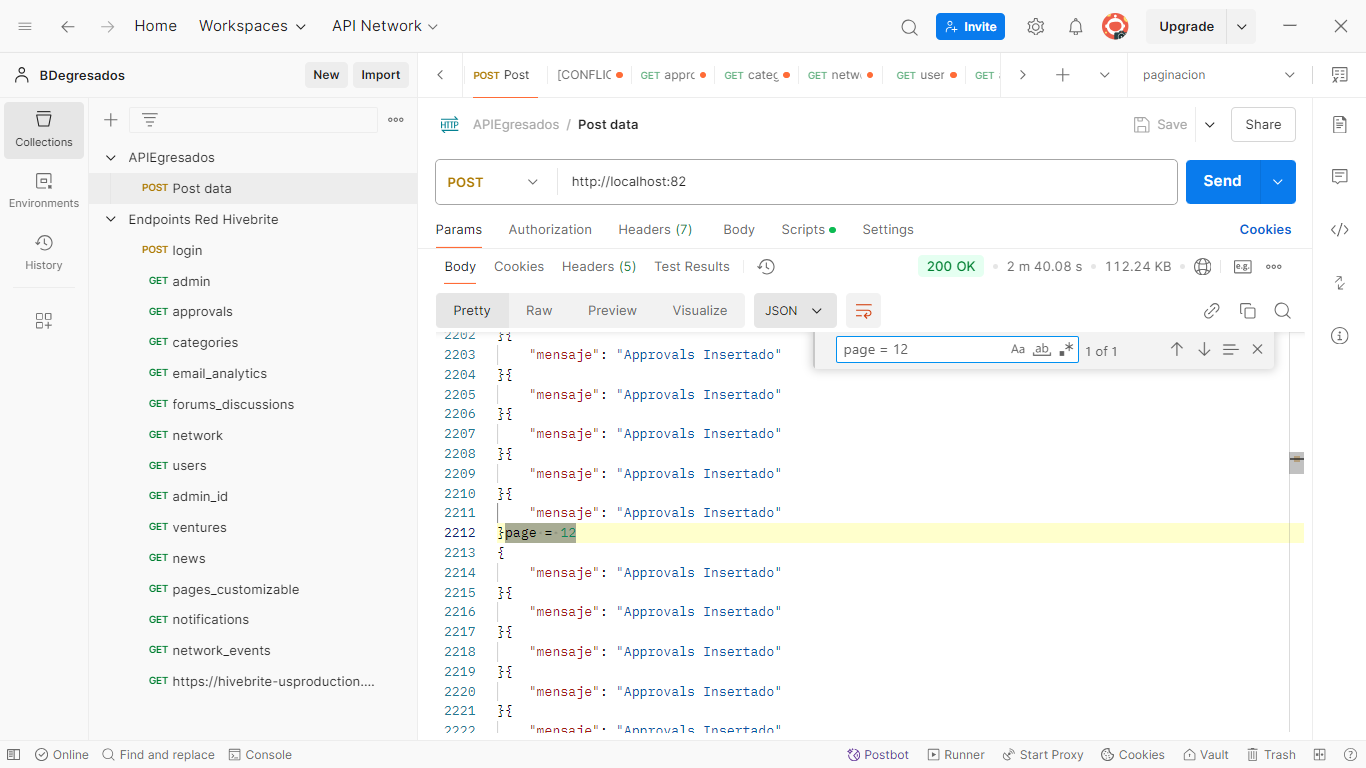
Si la inserción se realizado exitosamente se mostrara un mensaje en donde dirá “(nombre de la tabla) insertado(a)” según la cantidad de registros no vacíos (y en algunos casos únicos) que existan almacenados en el array de su respectivo endpoint de hivebrite

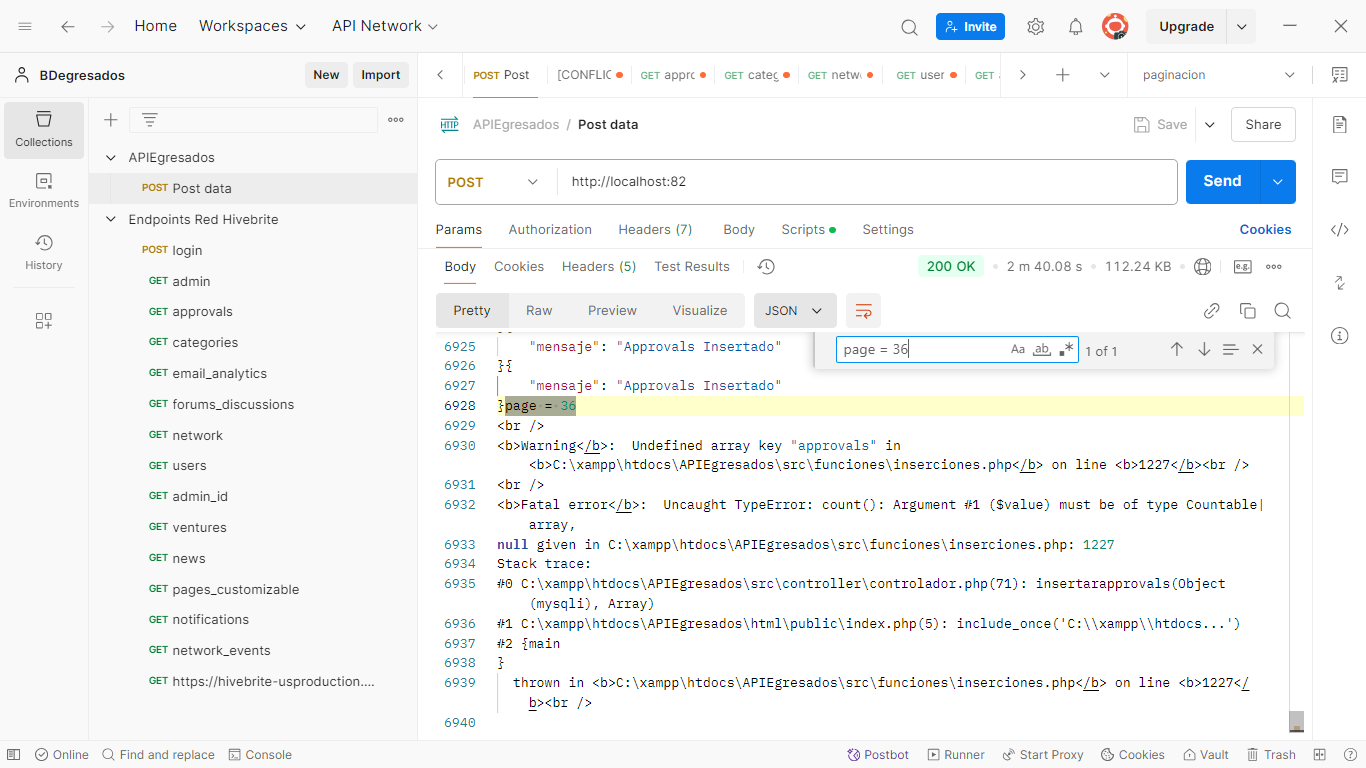
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Imagen 10. Resultado de una inserción exitosa

Si la inserción se realiza a un endpoint que contenga múltiples páginas, se usa el ciclo for que esta junto a las funciones de ese endpoint, por cada pagina insertada se mostrara un pequeño mensaje que dirá “page = (numero de la página actual)” que indicara en que pagina del endpoint va por si se llega a presentar el error de la petición vacía (más información en el apartado de errores) junto con al final terminar en un error de arreglo nulo que indica que en la siguiente pagina ya no existen más registros.





El tiempo estimado para la inserción de las tablas según su endpoint es el siguiente:

* Endpoint Notifications: 3 horas y 40 minutos aproximadamente
* Endpoint Users: 20 minutos aproximadamente
* Endpoint Approvals: 2 minutos y 4 segundos aproximandamente
* Endpoint News: 1 minuto y 30 segundos aproximadamente
* Resto de Endpoints: menos de 1 minuto aproximadamente

# ERRORES

A continuación, se va a profundizar en los errores mas comunes que se pueden encontrar al momento de insertar los datos en postman.

* Error de la petición vacía: Este error se hace presenta aleatoriamente al momento de pedir una nueva solicitud al servidor de hivebrite, la forma en la que funciona este error es haciendo que la función de curl.php llamada getCommunations reciba una respuesta vacía, lo cual provoca que la variable declarada en controlador.php guarde un objeto vacio, el cual hace que no se pueda realizar la inserción de datos al no haber un array fuente para insertar dichos datos.

Se desconoce la causa directa de este error, aunque se teoriza que puede ser provocado por la configuración de la forma en que los servidores de la red de egresados administran sus peticiones simultaneas. Este error es el principal culpable de romper el bucle de inserción masiva de los datos, si eso ocurre, simplemente se reemplaza el valor del iterador por el valor de la pagina hasta donde pudo llegar y se vuelve a enviar la solicitud.

* Error de duplicados por llave foránea: Este error sucede cuando se trata de insertar registros en una tabla cuando ya existen registros en la base de datos en esa misma tabla, para evitar este error hay que asegurarse de que no existan datos registrados previamente en la base de datos.

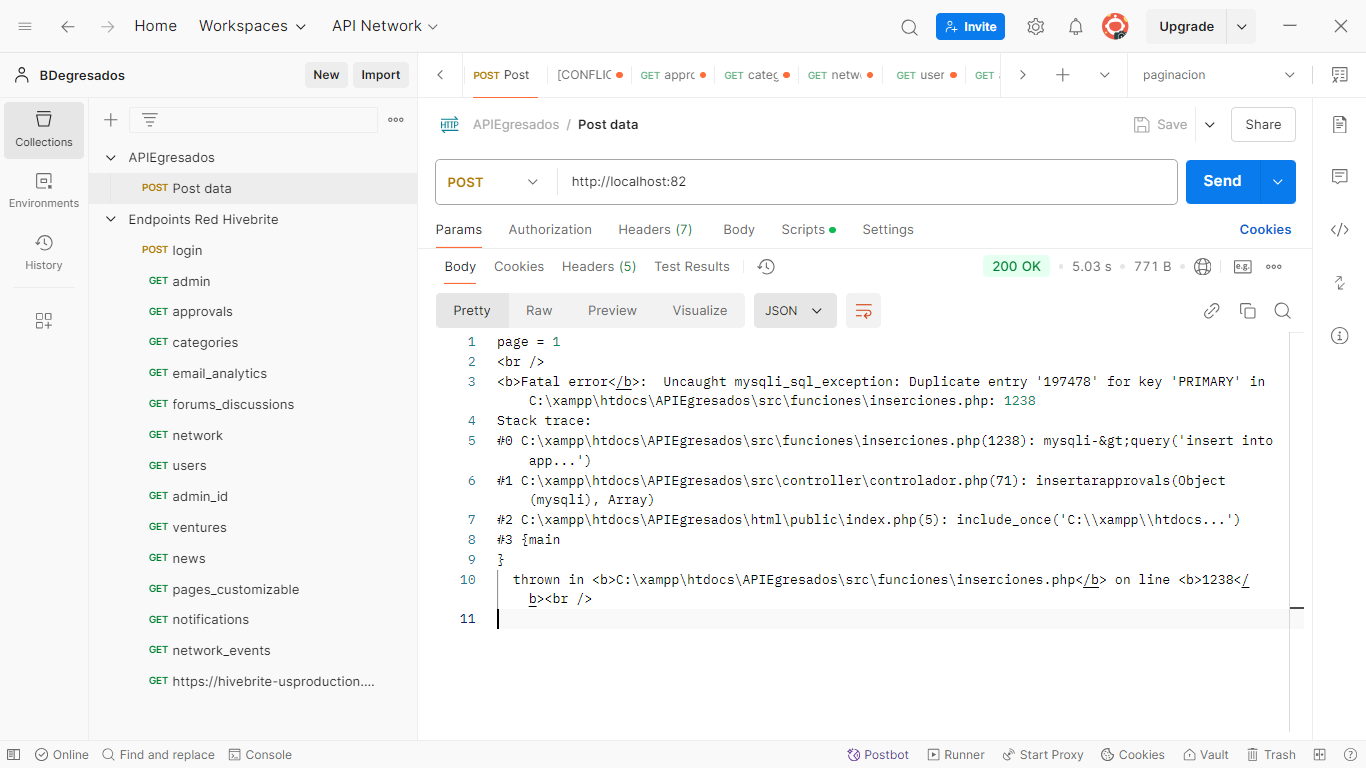


Imagen 11. Error de registros duplicados por la llave foránea de la base de datos

* Error de arreglo nulo en bucle: Similar al error de la peticion vacia, este error ocurre cuando al pasar a la siguiente pagina no existen registros, solo que a diferencia el anterior error este es intencional cuando el numero de pagina es mayor a la cantidad de páginas de contenidas en el endpoint, indicando que ya termino su proceso de inserción.

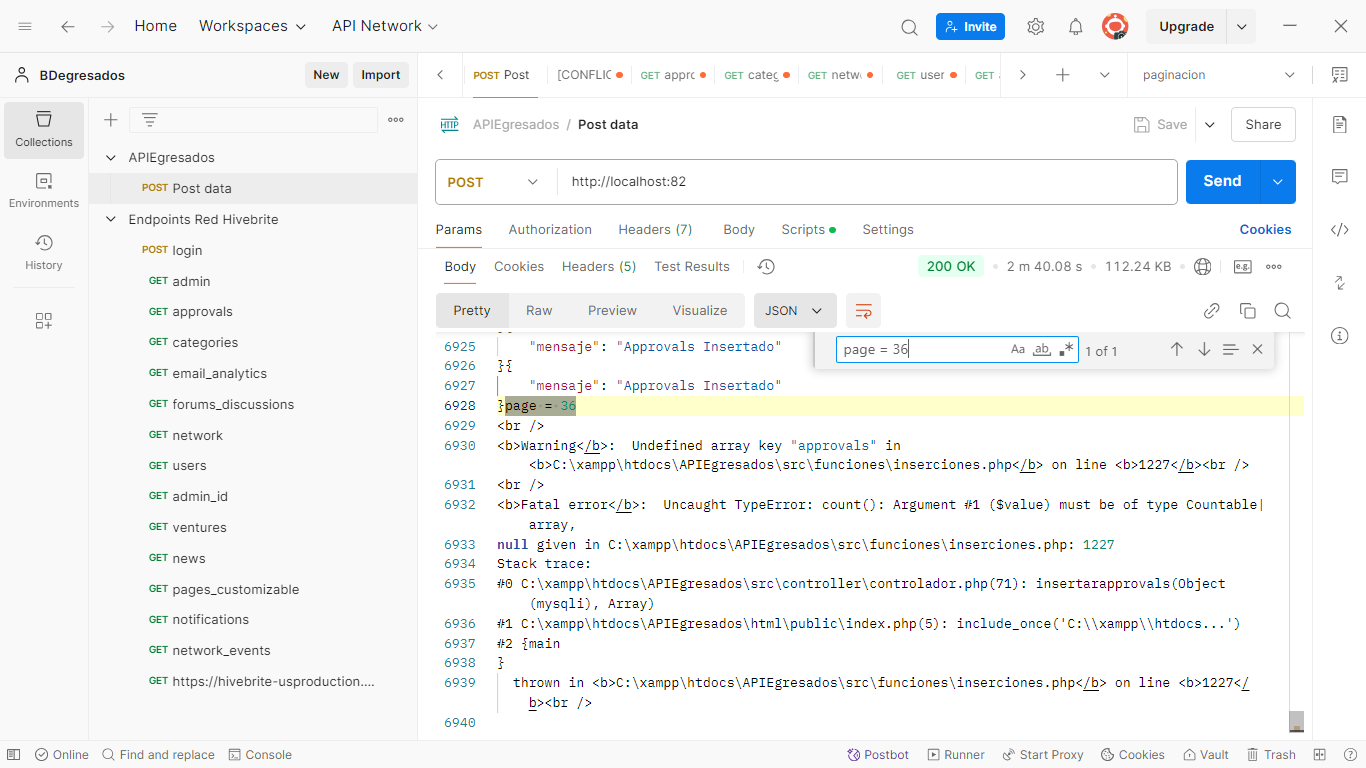


Imagen 12. Error de arreglo nulo en un bucle

* Error de sintaxis: Los errores de sintaxis se suelen presentar cuando la información de los endpoints supera el espacio de memoria permitido para cada atributo de la tabla a la que se le esta insertando datos o cuando el tipo de dato no corresponda con el campo solicitado. Para evitar este error consultar el diccionario de datos o la estructura de las tablas de la base de datos.

RECOMENDACIONES

A continuación, unas recomendaciones para tener en cuenta al momento de insertar los datos

1. Asegurarse de que la tabla a la que se le va a insertar la información esta vacia en la base de datos.
2. Si la tabla tiene una relación, asegurarse de que primero se inserte la información en la tabla padre (independiente) y luego en las tablas hijas (dependiente), guiándose con el diccionario de datos o según el orden descendente que esta en cada grupo de funciones por endpoint.
3. Las tablas intermedias se llenan a lo último ya que estas, a diferencia de las otras tablas, son las únicas que cuya información no se encuentra en los endpoints de hivebrite sino en la base de datos.
4. Si bien algunas tablas y algunos endpoints permiten llenarse en simultaneo, no se recomienda llenar un grupo de tablas de un endpoint al tiempo y mucho menos varios grupos de tablas de diferentes endpoints por el conflicto que puede haber con las llaves foráneas de la base de datos.