# DBB

Cuenta con el archivo SQL con las instrucciones para la creación de la base de datos con los parámetros requeridos para el proyecto y los nombres de Tablas y Columnas utilizados en la carpeta Entregable

También cuenta con rutinas que serán utilizadas, a excepción de las que tienen un componente tipo DATE, al utilizar las rutinas para registros de tipo DATE no reconoce la fecha y registra 0000-00-00, las rutinas con componentes de tipo DATE son solo ejemplos para su uso que pueden moverse directamente al código.

Las variables de tipo DATE del SQL tienen el formato YYYY-MM-DD

# Entregable

## key

El archivo Json de las llaves esta con variables de prueba para la inicialización de una base de datos en los archivos php.

## Carga Masiva REM y ejemplo

El archivo xlsx esta para la prueba de lectura y registro desde un Excel a la base de datos con un archivo php como intermediario, el archivo cuenta con 2 libros y el área de las pruebas es el 2do libro, por lo tanto en RegitroRemuneraciones.php la variable $sheet toma el 2do libro del archivo Excel previamente cargado en la variable $objPHPExcel con la función getSheet(1) (1 porque la contabilización de variables empieza desde el 0, por lo que getSheet(0) leería el 1er libro).

## PHP

En los archivos php en generalidad se inicia con un set\_time\_limit que expande el tiempo límite en que se puede cargar la página, de 30 a 700 segundos; se incluye en los programas que tardan mucho en ejecutarse completamente o sino fallara con un **Fatal Error: Maximum execution time of 30 seconds exceded in…**

Los programas que buscan un Json desde la API de CymaSuite utilizan la variable $opts con un array con el método y la API-KEY para crear el contexto para acceder a los datos en la variable $data.

Todos los programas que buscan un Json lo almacenan en un $json\_ugly que luego es decodificado y codificado en $json\_pretty para una mejor visualización en caso de algún error; el enlace de la API cambia de caso en caso y se especificaran sus variables en el archivo correspondiente.

Todos los programas se conectan a la base de datos con la información de key.json para establecer la conexión.

Todos los archivos tienen la variable $PKempresa que es proporcionada por CymaSuite, se usa para parte del enlace de la API y se guardó como una variable Usuario en la base de datos.

Los registros en la tabla libro cuentan con 2 variables extras para la clasificación con la PK del elemento registrado y una PK de la tabla origen del elemento.

Por último, no hay ningún tipo de método que evite la repetición aun porque está en periodo de pruebas y los datos repetidos se utilizan haciendo pruebas de funcionamiento, fallas y pruebas de datos relacionados entre funciones. (como RegistrosHonorarios y RegistroHonorarioLibro)

### RegistroBanco

La API solicita la información bancaria general (sin movimientos) de las empresas registradas en CymaSuite únicamente con la variable $PKempresa previamente registradas.

Se revisa si la variable institution\_country dentro de la variable bank dentro el primer array ([0]) del json ya está registrada en la tabla bancos, si es así la llave primaria del registro se guarda para un futuro registro, si no se registra el nuevo tipo de banco.

Se revisa si la variable name dentro del primer array ([0]) del json ya está registrada en la tabla tipocuenta, si es así la llave primaria del registro se guarda para un futuro registro, si no se registra el nuevo tipo de cuenta.

Teniendo todo eso en cuenta el programa guarda las 5 variables de la tabla banco.

### RegistroCompraVenta

La API para la solicitud de Compra y Venta utiliza 3 variables extras, mes, año y tipo de operación (COMPRA o VENTA) en las variables $mes, $yr y $operation; se utiliza un while para que se ejecute 2 veces, uno con venta y uno con compra, a esto se le suma la variable $archivo que inicia como 1 y se utiliza para buscar que tipo de código de documento era de tipo 1 IEV (referente a ventas) o 2 IEC (referente a compra), así al terminar el 1er ciclo while a $archivo se le agrega una unidad y $operation se cambia por COMPRA.

Al inicio se hace una búsqueda en la tabla codigosdocumentos donde los elementos de la columna Archivo son iguales a la variable $archivo.

Se registra la primary key y la columna Codigo en las variables $pk\_docid y $pk\_doc, la 1ra variable es necesaria para la consulta de la API porque regresa únicamente los registros de compra o venta que posean ese código. La 2da variable es para el registro porque la tabla de compraventa esta relacionada con la tabla codigosdocumentos con la primary key.

En la devolución del Json de la API tiene una variable llamada total\_items que señala cuantos registros se encontraron, si el resultado de esta variable es mayor a 0.

Se revisa si las variables de rut del json ya está registrada en la tabla empresa, si es así la llave primaria del registro se guarda para un futuro registro, si no se registra el nuevo rut y razón social con un tipo de empresa null.

Se revisa si la variable descTipoTransaccion del json ya está registrada en la tabla tipomovimiento, si es así la llave primaria del registro se guarda para un futuro registro, si no se registra el nuevo tipo de movimiento.

Finalmente se hace un análisis de cada variable de compra y venta que pueda ser NULL y se coloca en una variable independiente, esto es necesario porque para registrar, por ejemplo, la variable 25, en el código SQL debe registrarse como ‘25’ entre comillas simples. En caso de esta variable ser null, debe escribirse sin comillas o habrá un error. Así este análisis de variables pregunta si cada variable es null, si lo es se queda como null, si no lo es se toma la variable y se guarda entre comillas simples para el registro.

Aparte de eso, se separan las variables tipo DATE, ya que el json envia los datos en formato DD/MM/YYYY y el formato del SQL es YYYY-MM-DD y el formato de fecha de PHP es MM/DD/YYYY por lo que el cambio de formato debe hacerse manualmente. Además de tener cuidado que la fecha de recepción tiene un formato que agrega la hora.

Teniendo todo eso en cuenta el programa guarda las 68 variables de la tabla compraventa.

### RegistroEmpresa

Sirve para registrar una empresa registrada en CymaSuite, la única variable que solicita es $PKempresa que fue previamente registrado.

Se revisa si la variable glosaActividad del json ya está registrada en la tabla tipoempresa, si es así la llave primaria del registro se guarda para un futuro registro, si no se registra el nuevo tipo de empresa.

Teniendo todo eso en cuenta el programa guarda las 4 variables de la tabla empresa.

### RegistroHonorarioLibro

Para las pruebas se hace el registro haciendo una búsqueda de todos los honorarios que tengan la misma PK\_Usuario que la variable $PKempresa.

El honorario debe hacer 3 registros en libro, 1ro el Bruto que debe registrarse como tipo 2 (traspaso) y cuenta 125 (HONORARIOS PROFESIONALES) además de registrar la columna Bruto en el DEBE.

El 2do registra el Pagado, debe registrarse como tipo 2 y cuenta 66 (HONORARIOS PAGADOS) además de registrar la columna Pagado en el HABER.

El ultimo registra el Retenido, debe registrarse como tipo 2 y cuenta 74 (RETENCION PROFESIONALES) además de registrar la columna Retenido en el HABER.

Los 3 registran su tabla de origen como 4 que corresponde a lo que viene de la tabla honorarios.

### RegistroHonorarios

La API solicita el año para devolver un json con todos los honorarios del año separados por mes, el año se le entrega desde la variable $year.

Se hace un while de 12 para buscar entre los 12 meses, en cada mes se revisa la variable current que muestra cuantos honorarios hay por mes. Si es mayor a 0 se hace un while la cantidad de veces de la cantidad de honorarios que hay por mes.

Se revisa si la variable state del json ya está registrada en la tabla estadohonorario, si es así la llave primaria del registro se guarda para un futuro registro, si no se registra el nuevo tipo de estado.

Se revisa si la variable rut del json ya está registrada en la tabla emisor, si es así la llave primaria del registro se guarda para un futuro registro, si no se registra el nuevo emisor. Para registrar un nuevo emisor se debe verificar su SocProf, es verdadero o falso, 0 es falso y 1 es verdadero.

Antes de registrar se separan las variables tipo DATE, ya que el json envía los datos en formato DD/MM/YYYY y el formato del SQL es YYYY-MM-DD y el formato de fecha de PHP es MM/DD/YYYY por lo que el cambio de formato debe hacerse manualmente.

Teniendo todo eso en cuenta el programa guarda las 9 variables de la tabla honorarios.

### RegistroMovimientos

La API solicita la información bancaria de los movimientos con una variable bancaria de acceso registrada previamente.

El json resultante no tiene ningún tipo de medición que indique cuantos elementos contiene con seguridad, así que se ejecuta un while con las variables buscadas en el json para que se siga ejecutando cuando esta no este vacía.

El registro del banco se encuentra en el formato YYYY-MM-DDTHH:MM:SS.mmmZ por lo tanto para equipararse al formato del SQL (YYYY-MM-DD) se hace una separación por la letra ‘T’ y que registra la primera mitad.

Teniendo todo eso en cuenta el programa guarda las 7 variables de la tabla de movimientos.

### RegistroRemunerariones

Para que el programa pueda leer un archivo Excel se necesita hacer la descarga del enlace comentado en la 1ra línea de código. La carpeta debe colocarse en la misma carpeta de este programa.

La variable $highestRow dice cuántas filas están ocupadas, se utiliza un for que empiece desde el 2 porque la 1ra fila tiene los títulos y no información de los registros.

Se revisa si la variable departamento del Excel ya está registrado en la tabla departamento, si es así la llave primaria del registro se guarda para un futuro registro, si no se registra el nuevo tipo de estado.

Se revisa si la variable departamento del Excel ya está registrado en la tabla trabajadores, si es así la llave primaria del registro se guarda para un futuro registro, si no se registra el nuevo trabajador.

Como los trabajadores aun no es algo que se pueda registrar y por la cantidad de variables se esta dejando en una variable auxiliar 1 previamente registrada.

Teniendo todo eso en cuenta el programa guarda las 40 variables de la tabla remuneraciones.

### RegistroRemuneracionLibro

Para las pruebas se hace el registro haciendo una búsqueda de todos los registros de remuneraciones que tengan la misma PK\_Usuario que la variable $PKempresa

Los registros de remuneraciones que se registran al HABER tipo 2 (traspaso) y cuentas 77 (AFP), cuentas 78 (ISAPRE), cuentas 73 (Impuesto Unico), cuentas 168 (SEGURO CESANTIA EMPLEADOR) y cuentas 165 (MUTUAL DE SEGURIDAD), registrando las columnas PREVISION, SALUD, IMPUNICO, SEGCES y MUTUAL en el HABER y registra la tabla de origen como 4 (Honorarios).

Los registros de remuneraciones que se registran al DEBE tipo 2 (traspaso) y cuentas 158 (SueldoBase), cuentas 159 (Gratificaciones), cuentas 163 (ASIGNACION DE TELEFONO), cuentas 162 (MOVILIZACION) y cuentas 161 (COLACION), registrando las columnas SBASE, GRATLEGAL, CONECT, MOVI y COLACION en el HABER y registra la tabla de origen como 4 (Honorarios).

### RegistroVentaLibro

Para las pruebas se hace el registro haciendo una búsqueda de todos los honorarios que tengan la misma PK\_Usuario que la variable $PKempresa y el Archivo del código de documento asociado sea = 1 (IEV).

Los registros de ventas se registran en libro 2 veces, 1ro el registro de la venta que debe registrarse como tipo 2 (traspaso) y cuenta 198 (VENTAS AFECTAS) además de registrar la columna detMntTotalen el HABER y registra la tabla de origen como 1 (Venta).

El 2do registra el reflejo, debe registrarse como tipo 2 y cuenta 5 (DEUDORES CLIENTES) además de registrar la columna detMntTotalen en el DEBE y registra la tabla de origen como 2 (Cliente) a esta fila se le agrega una columna extra Auxiliar, que aplica como el número de días antes del vencimiento de una deuda (por defecto se asigna como 30, refiriéndose a 30 días).