

## **Título del tema:**

ETL con Node.js

## **Integrantes:**

Jesus Andres Zini

## **Descripción sobre la Situación Actual:**

Durante la cursada se vio el proceso de ETL desde el rol de un DBA utilizando por ejemplo herramientas como 'Pentaho data integration'. En este trabajo se propone mostrar un proceso de ETL desde el rol de un desarrollador web, haciendo uso del entorno de ejecución Node.js, la librería [Sheet.js](#) para manipulación de documentos de Excel, el lenguaje de programación y scripting TypeScript, y de una base de datos relacional PostgreSQL.

## **Planteo del Problema:**

Durante el proceso de desarrollo de una aplicación web para un estudio contable de la ciudad de Corrientes, nos encontramos con el problema de que este estudio se servía de un documento de Excel para anotar los impuestos que pagaban sus clientes y los vencimientos de estos impuestos de acuerdo a la terminación del CUIT de cada contribuyente.

Surgió la necesidad de disponer de esa información también desde la aplicación web, pero sin dejar de utilizar el documento Excel original donde los empleados cargan los impuestos con sus vencimientos.

Una forma de conseguir esto, podría ser creando una nueva interfaz en la aplicación front-end para ingresar esos datos manualmente. Esta propuesta no era la más óptima ya que los empleados del estudio contable iban a tener que ingresar los mismos datos dos veces, una en el documento de Excel y otra en la aplicación web.

Se propuso entonces encontrar una forma de extraer los datos de ese documento Excel original y cargarlos automáticamente a la base de datos principal de la aplicación web; o sea realizar un proceso de ETL.

El documento esta constituida de la siguiente forma:

Cada hoja es un impuesto y debe llevar el nombre del impuesto como título.

En cada hoja de impuesto va a existir una tabla donde las filas (i) indican cada mes del año, y las columnas (j) indican la terminación de CUIT que puede tener un contribuyente.

El elemento  $[i,j]$  es el día de vencimiento de ese impuesto, en ese mes para determinado contribuyente dependiendo de la terminación de su numero de CUIT.

	A	B	C	D	E	F	G	H	I	J	K
1	MES	CUIT_0	CUIT_1	CUIT_2	CUIT_3	CUIT_4	CUIT_5	CUIT_6	CUIT_7	CUIT_8	CUIT_9
2	Enero	13	13	13	13	14	14	14	15	15	15
3	Febrero	17	17	17	17	18	18	18	19	19	19
4	Marzo	15	15	15	15	16	16	16	17	17	17
5	Abril	13	13	13	13	14	14	14	15	15	15
6	Mayo	13	13	13	13	14	14	14	17	17	17
7	Junio	14	14	14	14	15	15	15	16	16	16
8	Julio	13	13	13	13	14	14	14	15	15	15
9	Agosto	13	13	13	13	17	17	17	18	18	18
10	Septiembre	13	13	13	13	14	14	14	15	15	15
11	Octubre	13	13	13	13	14	14	14	15	15	15
12	Noviembre	15	15	15	15	16	16	16	17	17	17
13	Diciembre	13	13	13	13	14	14	14	15	15	15
14											
15											
16											
17											
18											
19											
20											
21											
22											
23											
24											
25											
26											
27											

Tabla de vencimiento de impuesto.

También hay casos en donde la nación todavía no define un día de vencimiento para determinados meses de un impuesto. Por lo que nos podemos encontrar con filas sin rellenar.

	A	B	C	D	E	F	G	H	I	J	K
1	MES	CUIT_0	CUIT_1	CUIT_2	CUIT_3	CUIT_4	CUIT_5	CUIT_6	CUIT_7	CUIT_8	CUIT_9
2	Enero	13	13	13	13	14	14	14	15	15	15
3	Febrero										
4	Marzo										
5	Abril										
6	Mayo										
7	Junio										
8	Julio										
9	Agosto										
10	Septiembre	13	13	13	13	14	14	14	15	15	15
11	Octubre	13	13	13	13	14	14	14	15	15	15
12	Noviembre	15	15	15	15	16	16	16	17	17	17
13	Diciembre	13	13	13	13	14	14	14	15	15	15

Tabla de vencimiento de impuesto donde todavía no se publicaron los vencimientos de todos los meses.

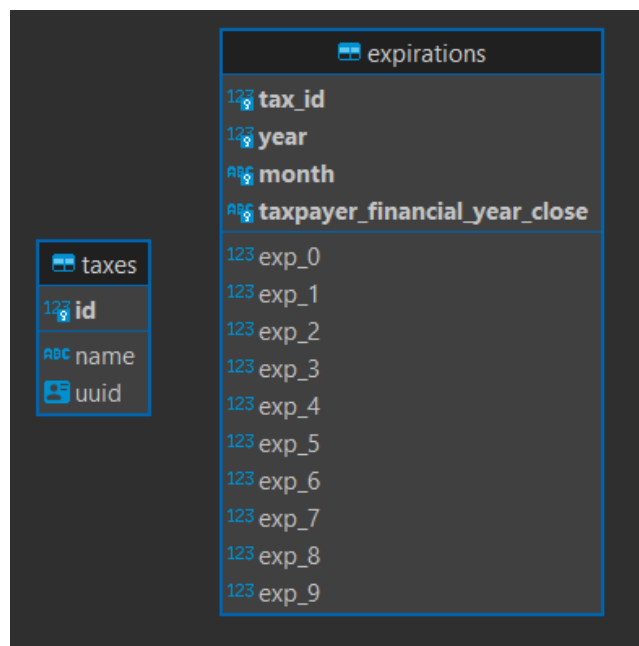
Existen también impuestos donde, para determinar el día de vencimiento de un impuesto para un contribuyente, además de los factores anteriores, también depende el mes de cierre del ejercicio económico que tiene cada contribuyente, entonces la matriz queda como la siguiente:

	A	B	C	D	E	F	G	H	I	J	K	L
1	CIERRE_EJER	MES	CUIT_0	CUIT_1	CUIT_2	CUIT_3	CUIT_4	CUIT_5	CUIT_6	CUIT_7	CUIT_8	CUIT_9
2		Enero	13	13	13	13	14	14	14	15	15	15
3		Febrero	17	17	17	17	18	18	18	19	19	19
4		Marzo	15	15	15	15	16	16	16	17	17	17
5		Abril	13	13	13	13	14	14	14	15	15	15
6		Mayo	13	13	13	13	14	14	14	14	17	17
7		Junio	14	14	14	14	15	15	15	16	16	16
8		Julio	13	13	13	13	14	14	14	15	15	15
9		Agosto	13	13	13	13	17	17	17	18	18	18
10		Septiembre	13	13	13	13	14	14	14	15	15	15
11		Octubre	13	13	13	13	14	14	14	15	15	15
12		Noviembre	15	15	15	15	16	16	16	17	17	17
13	Enero	Diciembre	13	13	13	13	14	14	14	15	15	15
14		Enero	13	13	13	13	14	14	14	15	15	15
15		Febrero	17	17	17	17	18	18	18	19	19	19
16		Marzo	15	15	15	15	16	16	16	17	17	17
17		Abril	13	13	13	13	14	14	14	15	15	15
18		Mayo	13	13	13	13	14	14	14	17	17	17
19		Junio	14	14	14	14	15	15	15	16	16	16
20		Julio	13	13	13	13	14	14	14	15	15	15
21		Agosto	13	13	13	13	17	17	17	18	18	18
22		Septiembre	13	13	13	13	14	14	14	15	15	15
23		Octubre	13	13	13	13	14	14	14	15	15	15
24		Noviembre	15	15	15	15	16	16	16	17	17	17
25	Febrero	Diciembre	13	13	13	13	14	14	14	15	15	15
26		Enero	13	13	13	13	14	14	14	15	15	15
27		Febrero	17	17	17	17	18	18	18	19	19	19

Tabla de vencimiento de impuestos contemplando el mes de cierre de ejercicio de cada contribuyente.

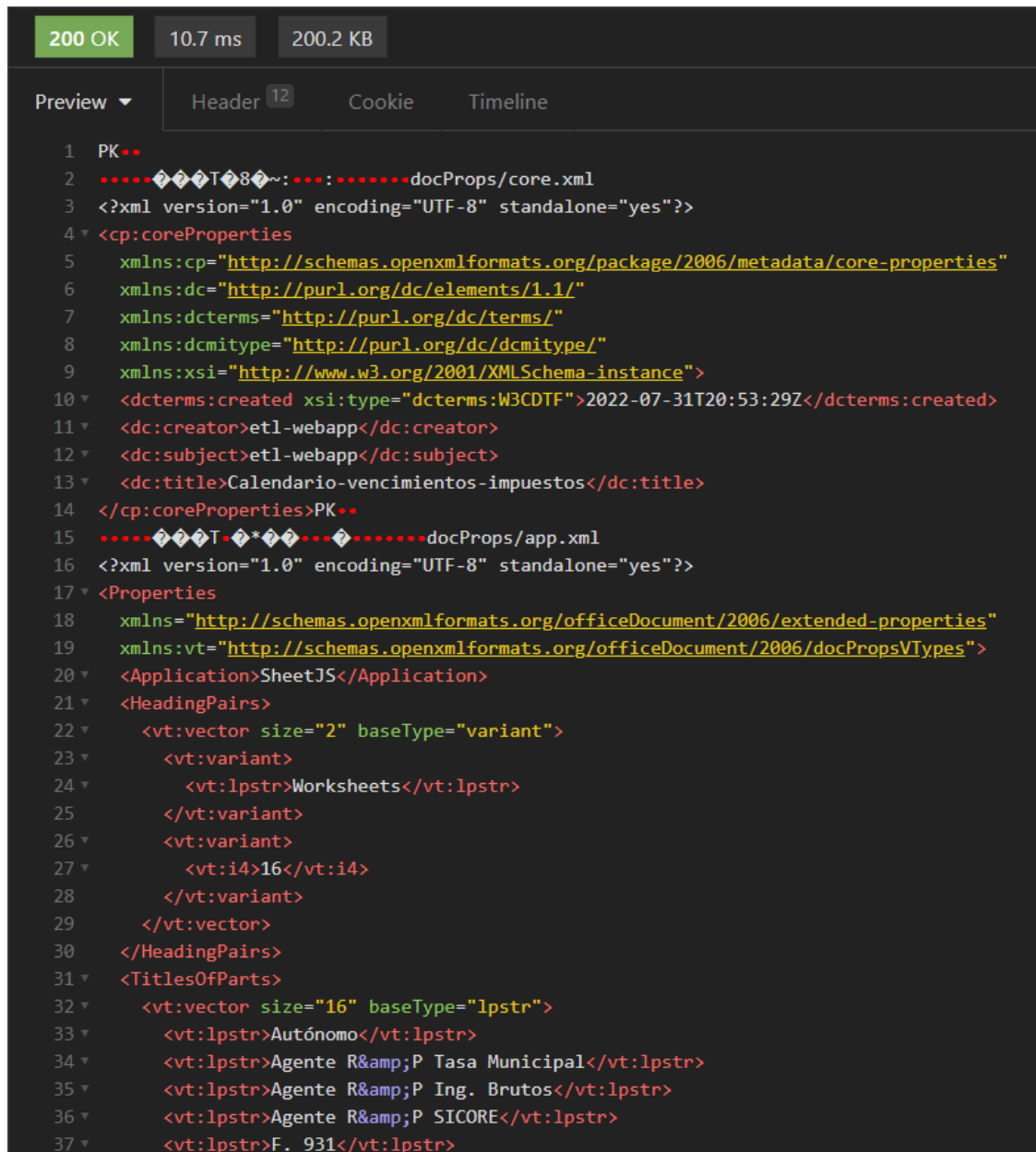
## Desarrollo de la Solución:

Se propuso como solución utilizar la librería Sheet.js para extraer los datos del documento Excel. Luego utilizar el paradigma funcional del lenguaje de programación JavaScript para aplicar las transformaciones necesarias a esos datos, y finalmente cargar los datos en la base de dato PostgreSQL de la aplicación web.



Estructura de las tablas creadas para almacenar la información relacionada a los impuestos y vencimientos.

Los datos del documento de Excel están codificados en formato XML, por lo que una vez extraídos, se deben convertir a formato JSON (PostgreSQL posee buen manejo de objetos en formato JSON <https://www.postgresql.org/docs/9.3/functions-json.html>), y aplicar las transformaciones a los datos según las necesidades de carga.



```
200 OK 10.7 ms 200.2 KB
Preview ▾ Header 12 Cookie Timeline
1 PK...
2 .....T8~:.....docProps/core.xml
3 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
4 <cp:coreProperties
5   xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
6   xmlns:dc="http://purl.org/dc/elements/1.1/"
7   xmlns:dcterms="http://purl.org/dc/terms/"
8   xmlns:dcmitype="http://purl.org/dc/dcmitype/"
9   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
10 <dcterms:created xsi:type="dcterms:W3CDTF">2022-07-31T20:53:29Z</dcterms:created>
11 <dc:creator>etl-webapp</dc:creator>
12 <dc:subject>etl-webapp</dc:subject>
13 <dc:title>Calendario-vencimientos-impuestos</dc:title>
14 </cp:coreProperties>PK...
15 .....T*~:.....docProps/app.xml
16 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
17 <Properties
18   xmlns="http://schemas.openxmlformats.org/officeDocument/2006/extended-properties"
19   xmlns:vt="http://schemas.openxmlformats.org/officeDocument/2006/docPropsVTypes">
20 <Application>SheetJS</Application>
21 <HeadingPairs>
22 <vt:vector size="2" baseType="variant">
23 <vt:variant>
24 <vt:lpstr>Worksheets</vt:lpstr>
25 </vt:variant>
26 <vt:variant>
27 <vt:i4>16</vt:i4>
28 </vt:variant>
29 </vt:vector>
30 </HeadingPairs>
31 <TitlesOfParts>
32 <vt:vector size="16" baseType="lpstr">
33 <vt:lpstr>Autónomo</vt:lpstr>
34 <vt:lpstr>Agente R&P Tasa Municipal</vt:lpstr>
35 <vt:lpstr>Agente R&P Ing. Brutos</vt:lpstr>
36 <vt:lpstr>Agente R&P SICORE</vt:lpstr>
37 <vt:lpstr>F. 931</vt:lpstr>
```

*Datos originales del documento Excel en formato XML.*

```

/**
 * upload EXCEL of taxes and expirations.
 */
public uploadExcel = async (
  req: Request,
  res: Response,
  next: NextFunction
): Promise<Response | void> => {
  try {
    const { year } = req.query;
    const excel = req.file?.buffer;
    const workbook = xlsx.read(excel);
    const worksheet = workbook.SheetNames;
    const taxes: string[] = worksheet;
    const expirations: IExpiration[] = [];

    // upload new taxes in the db table
    const uploadedTaxes = await excelSvc.uploadTaxes(taxes);

    if (uploadedTaxes) {
      // iterate through EXCEL sheets
      taxes.forEach((sheetName, i) => {
        const currentTax = uploadedTaxes.find(
          (row) => row.name === sheetName
        ); // find current tax

        if (currentTax) {
          const jsonSheet: IExcelRow[] = xlsx.utils.sheet_to_json(
            workbook.Sheets[sheetName]
          );

          let currentCloseMonth: string | null = null; // var to save CIERRE_EJERCICIO value

          // iterate through EXCEL rows
          jsonSheet.forEach((row: IExcelRow) => {
            // creating 'expiration' object
            const exp: IExpiration = {
              year: Number(year as string),
              tax_id: currentTax.id,
              month: getMonth(row.MES),
              taxpayer_financial_year_close: getMonth(row.CIERRE_EJERCICIO),
              exp_0: row.CUIT_0,
              exp_1: row.CUIT_1,
              exp_2: row.CUIT_2,
              exp_3: row.CUIT_3,
              exp_4: row.CUIT_4,
              exp_5: row.CUIT_5,
              exp_6: row.CUIT_6,
              exp_7: row.CUIT_7,
              exp_8: row.CUIT_8,
              exp_9: row.CUIT_9,
            };

            // if row is the one with CIERRE_EJERCICIO, then save that value
            if (row.CIERRE_EJERCICIO) {
              currentCloseMonth = row.CIERRE_EJERCICIO;
            }

            // if next row has its CIERRE_EJERCICIO undefined (because of how it's parsed from excel to json),
            // then assign the saved value
            if (!row.CIERRE_EJERCICIO) {
              exp.taxpayer_financial_year_close = getMonth(currentCloseMonth);
            }

            // push formatted 'expiration' in the global array of expiration objects
            expirations.push(exp);
          });
        }
      });

      // upload new expirations
      const uploadedExps = await excelSvc.uploadExpirations(
        expirations,
        year as string
      );

      // save excel
      excelSvc.saveExcel(workbook);
    }
  } catch (err) {
    return next(err);
  }
};

```

*Funciones TypeScript para la transformación de datos.*

Además de las funciones escritas en código TypeScript, también son de gran importancia las funciones PLpgSQL escritas para realizar el proceso de carga de los datos.

```
CREATE FUNCTION config.upload_taxes(p_taxes text[]) RETURNS SETOF config.taxes
    LANGUAGE plpgsql
    AS $$
DECLARE
    v_tax text;
    v_tax_id int;
BEGIN
    IF (p_taxes IS NOT NULL) THEN
        --
        -- insert taxes or update if it comes with the same name but different case
        --
        FOREACH v_tax IN ARRAY p_taxes
        LOOP
            v_tax_id = (SELECT id FROM config.taxes WHERE LOWER(name) = LOWER(v_tax));

            IF (v_tax_id IS NOT NULL) THEN
                UPDATE config.taxes SET name = v_tax WHERE LOWER(name) = LOWER(v_tax);
            ELSIF (v_tax_id IS NULL) THEN
                INSERT INTO config.taxes(name) SELECT v_tax;
            END IF;
        END LOOP;

        --
        -- delete taxes that there are not in the new array of taxes
        --
        DELETE FROM config.taxes WHERE NOT (name ILIKE ANY(p_taxes));

        --
        -- return taxes
        --
        RETURN QUERY
        SELECT * FROM config.taxes;
    END IF;
END
$$;
```

*Funciones SQL para la carga de datos de impuestos.*

```

CREATE FUNCTION config.upload_expirations(p_expirations jsonb, p_year smallint) RETURNS
SETOF config.expirations
LANGUAGE plpgsql
AS $$
DECLARE
obj_exp json;
v_tfyc month_or_null_enum;
BEGIN
    IF (p_expirations IS NOT NULL) THEN

        -- remove all the old expirations from the year to upload
        DELETE FROM config.expirations WHERE year = p_year;

        -- insert new expirations
        INSERT INTO config.expirations
        SELECT
            (obj_exp."value" ->> 'tax_id')::int,
            (obj_exp."value" ->> 'year')::smallint,
            (obj_exp."value" ->> 'month')::month_enum,
            CASE
                WHEN obj_exp."value" ->> 'taxpayer_financial_year_close' IS NULL THEN ''
                ELSE (obj_exp."value" ->> 'taxpayer_financial_year_close')::month_or_null_enum
            END,
            (obj_exp."value" ->> 'exp_0')::smallint,
            (obj_exp."value" ->> 'exp_1')::smallint,
            (obj_exp."value" ->> 'exp_2')::smallint,
            (obj_exp."value" ->> 'exp_3')::smallint,
            (obj_exp."value" ->> 'exp_4')::smallint,
            (obj_exp."value" ->> 'exp_5')::smallint,
            (obj_exp."value" ->> 'exp_6')::smallint,
            (obj_exp."value" ->> 'exp_7')::smallint,
            (obj_exp."value" ->> 'exp_8')::smallint,
            (obj_exp."value" ->> 'exp_9')::smallint
        FROM jsonb_array_elements (p_expirations) obj_exp;

        -- return expirations
        RETURN QUERY
        SELECT * FROM config.expirations;
    END IF;
END
$$;

```

*Funciones SQL para la carga de datos de vencimientos.*

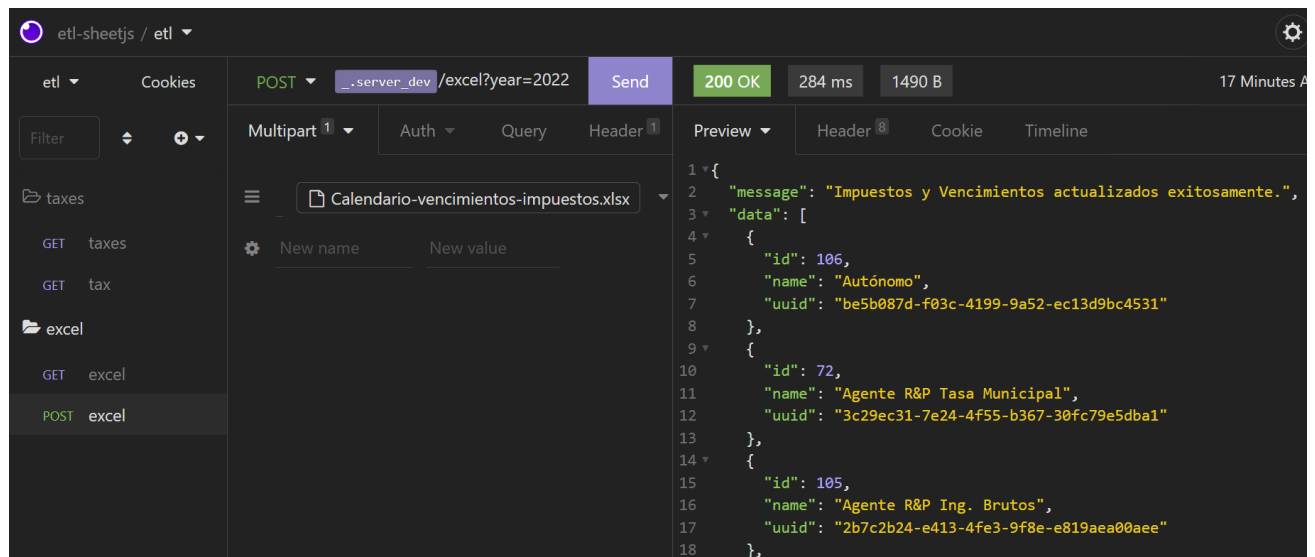
## Resultados:

Utilizando el Cliente REST Insomnia podemos ejecutar las funciones llamando a los endpoints de la aplicación back-end.

A la izquierda se puede apreciar la llamada al endpoint, donde se está enviando en el cuerpo de la petición, el documento de Excel llamado "Calendario-vencimientos-impuestos.xlsx". También se pasa como query-

string el año (year) al que corresponde estos impuestos y sus vencimientos, esto se hace para guardar un registro histórico en la base de datos.

A la derecha se visualiza la respuesta de esa llamada en formato JSON. Esa respuesta es el retorno de los datos después de haber sido extraídos del documento Excel gracias a librería Sheet.js, manipulados mediante las funciones de transformación escritas en TypeScript, y cargados mediante las funciones escritas en PLpgSQL.




*Ejecución del código de ETL desde un Cliente REST.*

## Impacto:

La solución propuesta fue satisfactoria para el cliente porque pudieron seguir utilizando su documento Excel original y cargar fácilmente esa información a la base de datos de la aplicación web. Para cerrar el desarrollo de esa funcionalidad, se creó una interfaz amigable para el usuario en la aplicación front-end.



Actualizar impuestos 

**CARGAR EXCEL**

Ningún archivo cargado

Año \*

**Descargar template**

**CONFIRMAR**

*Interfaz para carga del documento Excel en el front-end.*

### Conclusiones:

Se destaca la importancia de los procesos de ETL en la actualidad y cómo estos son necesarios en muchos ámbitos diarios de los negocios de cualquier tamaño y rubro.

También que estos procesos pueden ser aplicado de distintas formas dependiendo del problema y de los roles del profesional de sistemas que debe encarar esa problemática.

### Link del proyecto:

<https://github.com/jesusandres31/etl-sheetjs>