**Solución Sistema de Reportes Escaw para la Empresa Polybarq**

**Descripción de la solución**

La solución consistió en actualizar el framework del sistema de reportes Escaw desde Reportico 4 (Año 2014) a Reportico 6 (Año 2018).

Reportico 6 provee soluciones que no estaban implementadas en Reportico 4 como …

El reto fue empezar la estructuración del sistema Escaw bajo el nuevo paradigma que se está imponiendo en el mundo del desarrollo: Los microservicios y contenedores

La implementación de Reportico 6 consistió en:

* Instalación de un VM Linux que servirá como plataforma para los contenedores a usarse
* Uso de la plataforma de Gitlab para soporte en la nube para el desarrollo, manteniéndose respaldos constantes del código y de la progresión del desarrollo
* Se crearon 2 contenedores con Docker: uno con Linux, apache y otro con SQL Server (este último para pruebas)

La implementación se hizo de manera transparente

Qué queda pendiente por hacer en el equipo de desarrollo Polybarq:

Mantener la progresión en cuanto al avance hacia la arquitectura de microservicios

**Detalles técnicos de la implementación**

1) Equipo Linux (Contenedor de Contenedores)

Tipo Virtual Machine

Nombre de la VM: Linux-mint-Containers

Sistema Operativo: Linux Mint 18.3

Software Instalado: Docker v. , Git, VisualStudio Code

Ip Address v.4: 192.168.1.186

Password de acceso: 040418

2) Contenedor del sistema Reportico:

Tipo: Docker Container

Nombre: reportico:latest

Sistema Operativo: Linux basado en debian, imagen oficial php (php:7.3.3-apache)

Software instalado: composer, zip, unzip, libpng-dev, zlib1g-dev, libbz2-dev, curl, apt- transport-https, gnupg, msodbcsql17, mssql-tools, unixodbc-dev, pdo, gd, bz2, sqlsrv, pdo\_sqlsrv, Reportico

Ip Address v.4: 192.168.1.186

Volumen de datos: ./reportico Desde la carpeta donde se ejecute el contenedor

Puerto de Acceso: 8010 map to 80

Git: https://gitlab.com/sistemaspolybarq/reportico-polybarq.git

3) Contenedor de MSSQL (para entorno de desarrollo):

Tipo: Docker Container

Nombre: mssql-polybarq:latest

Sistema Operativo: Ubuntu 16.04  
Git: https://gitlab.com/sistemaspolybarq/mssql-polybarq.git

4) Archivo plantilla de Macros en Excel

Clave para acceso a macros: polyB2012.

5) Repositorio en Gitlab (https://gitlab.com)

Usuario: sistemaspolybarq  
email: sistemas@polybarq.com

Contraseña: polyB2012.

**Software y utilidades empleados**

* Visual Studio Code

Extensiones: GitLens

* Microsoft Azure Data Studio
* Git
* Docker – Docker Compose

**Procedimiento para la creación del Entorno de Desarrollo y Testing**

Los pasos a seguir son los siguientes:

1. Creación de cuenta en Gitlab

* Creación de proyectos (reportico y mssql)

Imagen base Reportico (Apache + PHP + Reportico + configuraciones)

Imagen base MSSQL (SQL Server Express + configuraciones)

Entorno de desarrollo

2. Creación y upload de imágenes y contenedores docker al registry de gitlab

3. Despliegue de los contenedores del registry

**Creación de cuenta en Gitlab**

Creada la cuenta en gitlab, se crean 3 nuevos proyectos:

El primer proyecto (Imagen base Reportico): Contendrá el Apache, PHP, el sistema base de Reportico y configuraciones adicionales.

a) Crear un directorio local llamado “dockerfile” con el siguiente contenido, (explicación en los comentarios):

**# Imagen base de Apache + PHP. Link oficial (https://hub.docker.com/\_/php)**

**FROM php:7.3.3-apache**

**# Instalación de dependencias y paquetes necesarios para composer y reportico**

**RUN apt-get update \**

**&& apt-get install -y zip unzip libpng-dev zlib1g-dev libbz2-dev curl apt-transport-https gnupg**

**# Instalación de los drivers sql server (msodbcsql17 mssql-tools unixodbc-dev)**

**# https://docs.microsoft.com/en-us/sql/connect/odbc/linux-mac/installing-the-microsoft-odbc-driver-for-sql-server?view=sql-server-ver15#debian**

**RUN curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add - \**

**&& curl https://packages.microsoft.com/config/debian/9/prod.list > /etc/apt/sources.list.d/mssql-release.list \**

**&& apt-get -yqq update \**

**&& apt-get update \**

**&& ACCEPT\_EULA=Y apt-get install -y msodbcsql17 mssql-tools unixodbc-dev**

**# Habilitar extensiones necesarias en PHP para reportico y sql server**

**# Como especifica en la sección de How to install more PHP extensions en el siguiente link:**

**# https://hub.docker.com/\_/php**

**RUN docker-php-ext-install pdo gd bz2 \**

**&& pecl install sqlsrv pdo\_sqlsrv \**

**&& docker-php-ext-enable sqlsrv pdo\_sqlsrv**

**# Instalación de Composer para PHP**

**# https://getcomposer.org/download/**

**RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer**

**# Instalación de Reportico**

**RUN composer create-project reportico-web/reportico:6.0.11 \**

**&& chmod -R 777 .**

**# Copiamos archivo de configuraciones de reportico a reportico en la imagen para**

**COPY ./config.php reportico/projects/admin/**

b) Descargar e instalar (Reportico 6.0.11) localmente y copiar al directorio de trabajo el archivo “**config.php**” que está en la ruta “reportico/projects/admin/.

c) Crear la imagen con el comando docker build de acuerdo a la siguiente sintaxis:

**docker build -t [NOMBRE-DE-LA-IMAGEN-PROVISTO-POR-GITLAB] [CARPETACONDOCKERFILE]**

En nuestro caso:

**docker build -t registry.gitlab.com/sistemaspolybarq/reportico .**

d) Subir la imagen al registry de gitlab del repositorio antes creado con el comando docker push de acuerdo a la siguiente sintaxis:

**docker push [NOMBRE-DE-LA-IMAGEN-PROVISTO-POR-GITLAB]**

En nuestro caso:

**docker push registry.gitlab.com/sistemaspolybarq/reportico**

El segundo proyecto (Imagen base mssql): Contendrá mssql con los datos de la base de datos andisacos**.**

a) Crear un directorio local llamado “dockerfile” con el siguiente contenido, (explicación en los comentarios):

**# https://hub.docker.com/\_/microsoft-mssql-server**

**FROM mcr.microsoft.com/mssql/server:2019-CU1-ubuntu-16.04**

**USER root**

**# Variables de entorno para Contraseña de mssql y Aceptar licencia**

**# Tal como se especifica en la documentación de la imagen base**

**# https://hub.docker.com/\_/microsoft-mssql-server**

**ENV SA\_PASSWORD=123456Ps ACCEPT\_EULA=Y MSSQL\_PID=Express**

**# Copiar backup File y entrypoint para el contenedor**

**COPY ./andisacos.bak /var/opt/mssql/backup/**

**COPY ./data-mssql.sh /**

**COPY ./entrypoint.sh /**

**# Otorgar permisos del archivo**

**RUN chmod +x /data-mssql.sh**

**# Configurar el punto de entrada para el contenedor**

**CMD /bin/bash /entrypoint.sh**

El archivo dockerfile hace referencia 3 archivos mas, he aquí el contenido de cada uno:

**entrypoint.sh:**

**/data-mssql.sh & /opt/mssql/bin/sqlservr**

**data-mssql.sh:**

**sleep 90s**

**# Restore de la base de datos andisacos en la imagen**

**# https://docs.microsoft.com/en-us/sql/Linux/tutorial-restore-backup-in-sql-server-container?view=sql-server-ver15**

**/opt/mssql-tools/bin/sqlcmd \**

**-S localhost -U SA -P '123456Ps' \**

**-Q 'RESTORE DATABASE B00010006 FROM DISK = "/var/opt/mssql/backup/andisacos.bak" WITH MOVE "B00010006" TO "/var/opt/mssql/data/B00010006.mdf", MOVE "B00010006\_log" TO "/var/opt/mssql/data/B00010006\_log.ldf"'**

**- andisacos.bak:**

Este archivo contiene un backup generado por sqlserver de la base de datos andisacos.

c) Crear la imagen con el comando docker build de acuerdo a la siguiente sintaxis:

**docker build -t [NOMBRE-DE-LA-IMAGEN-PROVISTO-POR-GITLAB] [CARPETACONDOCKERFILE]**

En nuestro caso:

**docker build -t registry.gitlab.com/sistemaspolybarq/mssql-polybarq .**

d) Subir la imagen al registry de gitlab del repositorio antes creado con el comando docker push de acuerdo a la siguiente sintaxis:

**docker push [NOMBRE-DE-LA-IMAGEN-PROVISTO-POR-GITLAB]**

En nuestro caso:

**docker push registry.gitlab.com/sistemaspolybarq/mssql-polybarq**

El tercer proyecto (Entorno de desarrollo):

a) Crear un directorio local “reportico-polybarq” y dentro de este directorio un archivo llamado docker-compose.yml con el siguiente contenido  
  
versión**: '3'**

**services:**

**# Servicio de php-apache con Reportico**

**web:**

**image: registry.gitlab.com/sistemaspolybarq/reportico**

**ports:**

**- '8000:80'**

**# Volumen para el desarrollo reactivo**

**volumes:**

**- ./reportico:/var/www/html/reportico**

**# Crea una conexion con el servicio de la base de datos**

**links:**

**- datab**

**# Variables para asignación de datos variables de Conexion a la base de datos**

**environment:**

**- USR=sa**

**- DB=B00010006**

**- HOST=datab**

**- PASSWD=123456Ps**

**- USR\_ANDISACOS=sa**

**- DB\_ANDISACOS=B00010006**

**- HOST\_ANDISACOS=datab**

**- PASSWD\_ANDISACOS=123456Ps**

**- USR\_CORDELERIA=sa**

**- DB\_CORDELERIA=B00010006**

**- HOST\_CORDELERIA=datab**

**- PASSWD\_CORDELERIA=123456Ps**

**- USR\_POLYBARQ=sa**

**- DB\_POLYBARQ=B00010006**

**- HOST\_POLYBARQ=datab**

**- PASSWD\_POLYBARQ=123456Ps**

**# Servicio de base de datos SQL Server**

**datab:**

**image: registry.gitlab.com/sistemaspolybarq/mssql-polybarq**

**ports:**

**- '1433:1433'**

b) A nivel del archivo docker-compose.yml se descargo una carpeta con reportico, para ser usado en el desarrollo de manera reactiva con el contenedor a través de los volúmenes de docker

c) Para poner en funcionamiento el entorno de desarrollo usar el comando **docker-compose up -d** sobre el directorio donde se encuentre el archivo docker-compose.yml

**Nota:** esperar unos minutos que los contenedores carguen correctamente

**Resumen de Cambios en Código Base**

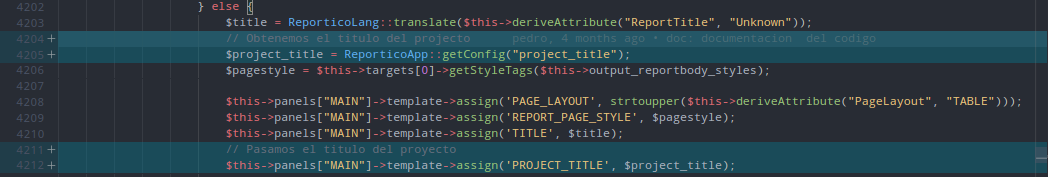
Comparar el commit 73f8a76 con el HEAD

La documentación esta realizada en cada archivo de código en forma de comentarios. Aquí se explicarán mas detalladamente alguunoas cambios.

**1) /reportico/src**

**1.1) reportico.php**

* Se agregaron estas lineas al archivo base de reportico ( reportico.php ) para poder obtener el titulo en los reportes

****

**1.2) reporticoCsv.php**

* Se cambio el separador del excel de “,” a “!” en cada una de las ocasiones, para evitar errores de columnas mezcladas con números decimales en la opción de exportar a excel

****

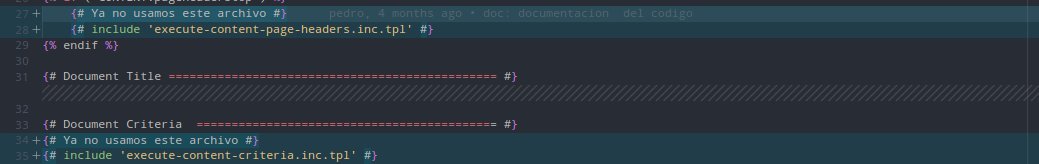
**2) /reportico/themes/default**

**2.1) execute-content-page-title.inc.tpl**

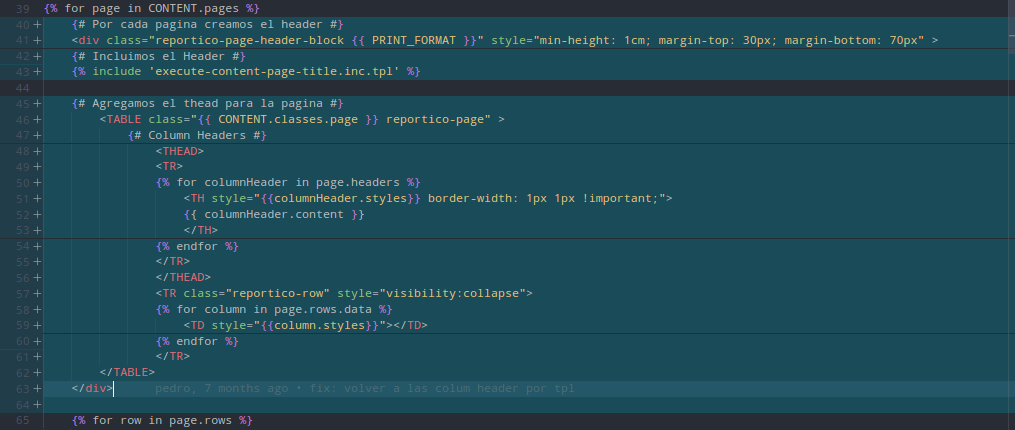
* Creamos la cabecera a usar en todas las paginas. Note el uso de PROJECT\_TITLE creado en el archivo reportico.php

**2.2) execute-content.inc.tpl**

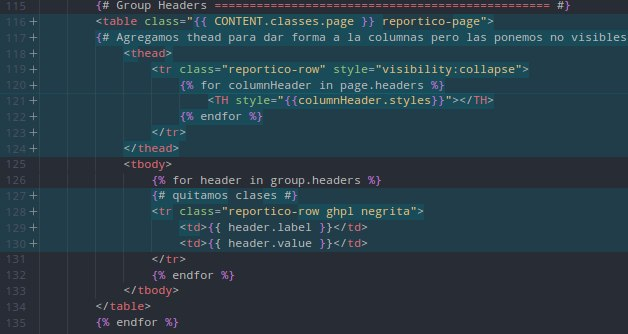
* Se comentaron estas líneas debido a que ya no usamos estos archivos

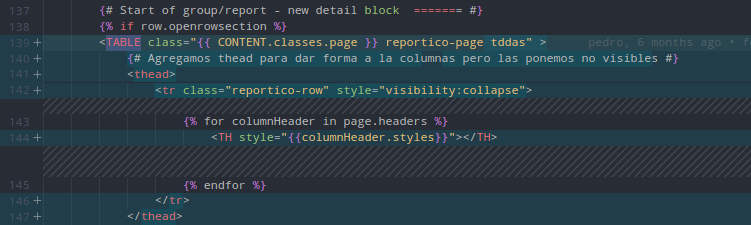
****

* Dentro del loop de cada pagina agregamos el header incluyendo el archivo **execute-content-page-title.inc.tpl**
* Se agrego aquí también los thead (encabezados de tabla o títulos de columna) en cada página

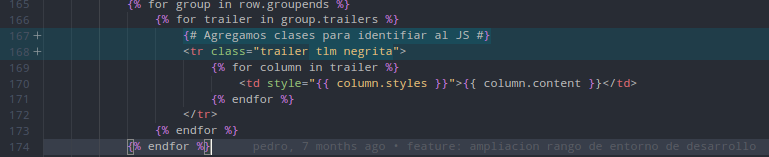
****

* Agregamos los thead a los GroupHeaders para que no se deformen las filas dentro del cuerpo del reporte. Esto se hacve tanto con el Group Header (encabezado de la data) como con la data de las tablas en si.
* Al header le agregamos una clase para identificarlo por javascript y otra clase para agregarle la letra negrita

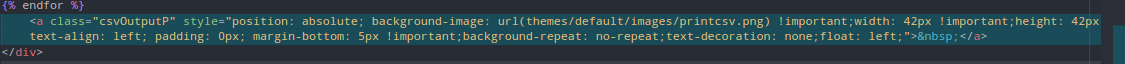
****

****

* Al footer le agregamos una clase para identificarlo por javascript y otra clase para agregarle la letra negrita



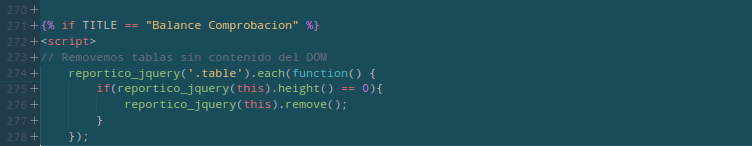
**2.3) execute.tpl**

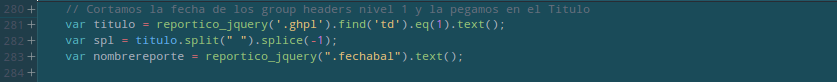
* Se agregó el botón para el output excel
* Para los ajustes específicos de cada reporte se hicieron scripts y se asignan con un condicional y el nombre del reporte

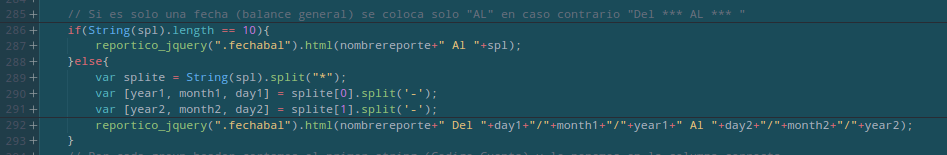


**Balance de comprobación:**

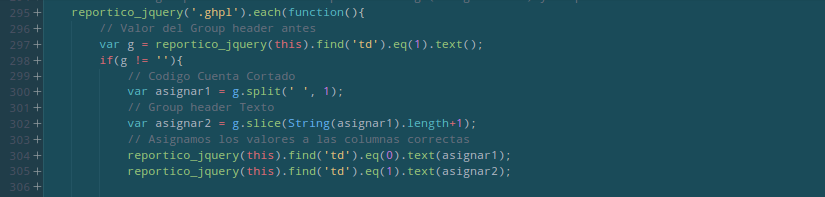
* Eliminamos tablas que no tienen contenido



* Cortamos la fecha del Group Header. Nivel 1
* Dependiendo si tiene fecha desde y hasta se coloca “del” y “al”



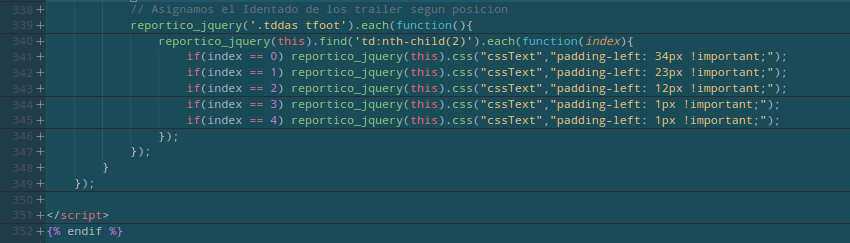
* Debido a que se está obteniendo el código cuenta a través de los group headers se cortan de estos y se colocan en la columna correcta



* Luego asignamos los indentados según el nivel de group header, colocando un padding

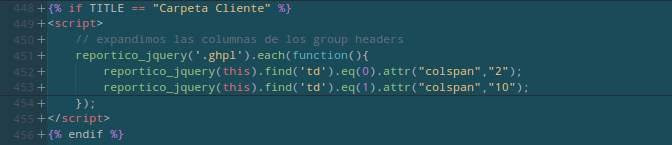


* De igual forma a los footers le añadimos indentado colocándole un padding



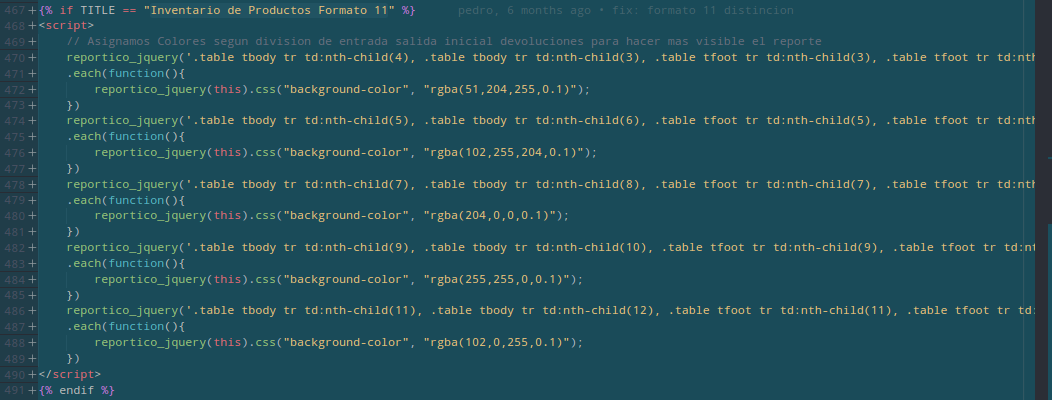
**Para el reporte carpeta cliente:**

* Expandimos algunas columnas agregando el atributo colspan



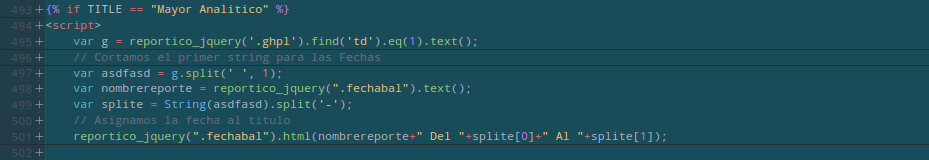
**Para el reporte de Inventario de Productos Formato 11:**

* Se añadió color a la columna por entrada, salida, devolución e inicial

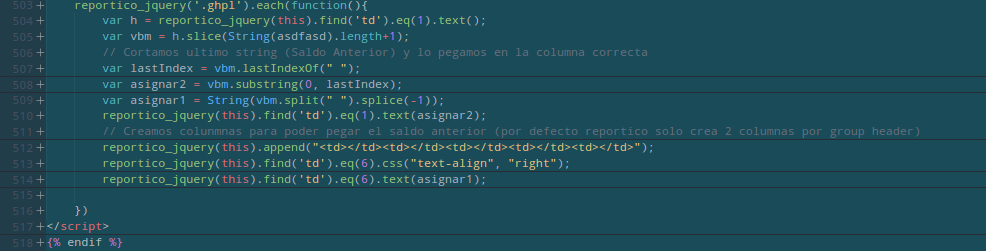


**Para el reporte mayor analítico:**

* Cortamos la fecha del group header y lo pegamos en el titulo

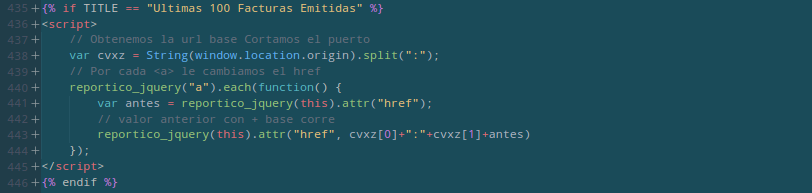


* Cortamos el saldo anterior del group header y lo pegamos en la columna correcta



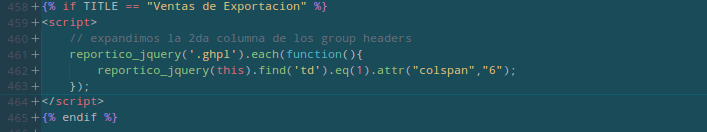
**Para el reporte de ultimas 100 facturas emitidas:**

* Obtenemos el url base sin puertos y luego le pegamos lo demás

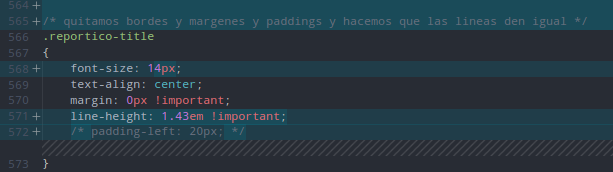


**Para el reporte de Ventas de exportación:**

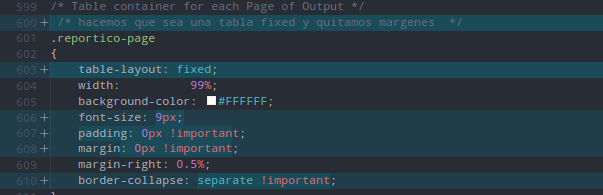
* Expandimos la 2da columna de los group headers



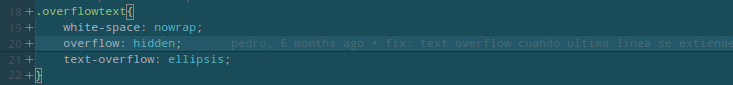
**2.4) reportico.css:**

Modificamos la clase .reportico-title para hacer que cada linea mida lo mismo y evitar malos saltos de pagina

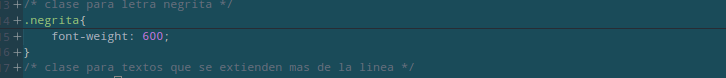
igual con esta clase

****

creamos clase para evitar que el texto se extienda mas de la linea



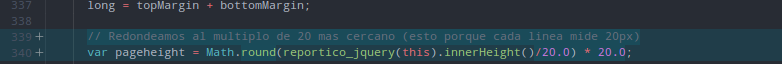
creamos clase para letra negrita



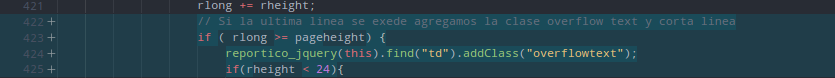
**3) /reportico**

**3.1) reportico.js:**

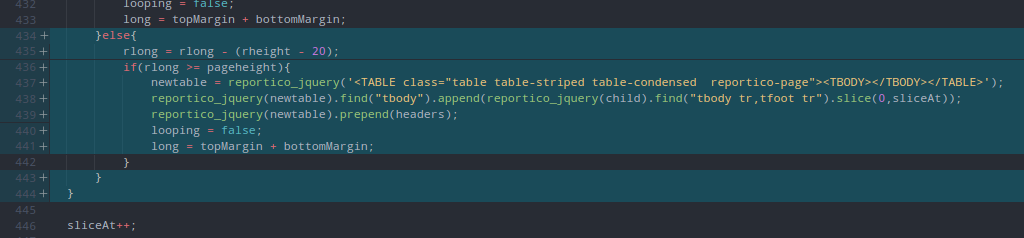
* Redondeamos el alto de la pagina a un múltiplo de 20 para hacer el corte de pagina iguales todos



* Si la ultima linea se excede agregamos la clase overflow text y corta linea



* Si no la excede corta la linea solamente



**4) /reportico/projects/[NOMBRE-PROYECTO]/**

**4.1) config.php**

* Al crear un proyecto en la interfaz de reportico se crea una carpeta con el nombre dado en el directorio reportico/projects/ esa carpeta creada contiene un archivo de configuración php al cual cambiamos la conexión a la base de datos para que sea por variable de entorno usando la función getenv(“”)



estas variables de entorno se asignan en el docker-compose.yml