OPTIMIZACIÓN DE FUNCIONALIDADES DE VISUALIZACIÓN DE CURVAS DE PERFORACIÓN DEL APLICATIVO ECOAGE WEB. MANUAL TÉCNICO Y USUARIO

Edwin Blanco Guerrero

David Alfonso Barrera Castellanos

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS

BUCARAMANGA

2022

CONTENIDO

1. MANUAL TÉCNICO

1.1. INSTALACIÓN DEL PROTOTIPO VISUALIZADOR

El código fuente estará alojado en un repositorio de código para que de esta manera el desarrollador pueda acceder a este, una vez descargado o clonado el repositorio, se recomienda usar Visual Studio Code como editor de código, con este editor es posible acceder directamente a una consola, aunque el siguiente paso también se puede realizar ejecutando el CMD de Windows. Es importante contar con una versión estable de Node.js en la computadora para poder ejecutar e instalar los paquetes necesarios del proyecto. Luego, se ejecuta la consola de Cmd y se ubica en el directorio del proyecto como se muestra en el ejemplo a continuación:

```
PS C:\Users\Edwin Blanco\Desktop\Escritorio\PrototipoECOAGEvisualizador>
```

Una vez realizado este paso se debe ejecutar el siguiente comando `npm install` y se la tecla “Enter”:

```
PS C:\Users\Edwin Blanco\Desktop\Escritorio\PrototipoECOAGEvisualizador> npm install
```

Lo anterior hará que se instalen las dependencias necesarias para poder iniciar el proyecto. Por último, para ejecutar el proyecto se debe ejecutar el siguiente comando `npm start`:

```
PS C:\Users\Edwin Blanco\Desktop\Escritorio\PrototipoECOAGEvisualizador> npm start
```

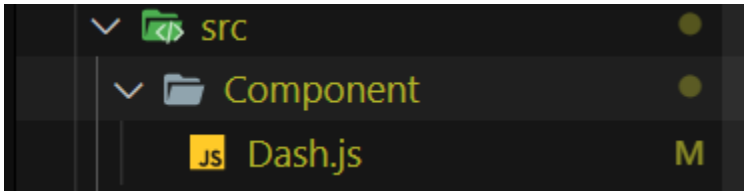
Lo anterior permitirá ejecutar el proyecto de forma local, por defecto en el puerto 3000 y se abrirá automáticamente el navegador predeterminado en donde se mostrará el visualizador.

```
You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.6:3000
```

Con los pasos anteriores se podrá ejecutar el proyecto y tener acceso al código fuente para cualquier modificación deseada.

1.1. DOCUMENTACIÓN CODIGO FUENTE

El código está basado en una aplicación de ReactJs pero básicamente el visualizador esta integrado en un solo componente del proyecto, el visualizador esta denominado “Dash.js” haciendo referencia a tablero:



En este componente se integra la gran mayoría del código del visualizador, inicialmente se importan lo módulos necesarios desde la librería de lightningChart y otras librerías los cuales después serán implementados:

```
import {
  lightningChart,
  Themes,
  AxisTickStrategies,
  synchronizeAxisIntervals,
  UILayoutBuilders,
  UIOrigins,
  UIElementBuilders,
  AutoCursorModes,
  translatePoint,
  UIDraggingModes,
  UIVisibilityModes,
  MarkerBuilders,
  UIBackgrounds,
  UIDirections,
  LegendBoxBuilders,
} from "@arction/lcjs";
import axios from "axios";
import React, { useRef, useEffect } from "react";
```

Luego se implementa la librería de “axios” para realizar peticiones al backend y poder obtener los datos que serán graficados posteriormente en el visualizador como se observa en el ejemplo a continuación:

```
axios
  .get("http://localhost:9000/api/datos_wits/wells/1/0", {
    responseType: "json",
  })
  .then(function (res) {
    if (res.status === 200) {
      console.log(res.data.length);
```

Después de realizada la petición se inicializan varias listas las cuales almacenaran los datos provenientes del backend como se observa en el ejemplo:

```
const dataV1ROPA = [];
const dataV2WOBA = [];
const dataV3TQA = [];
const dataV4RPMA = [];
```

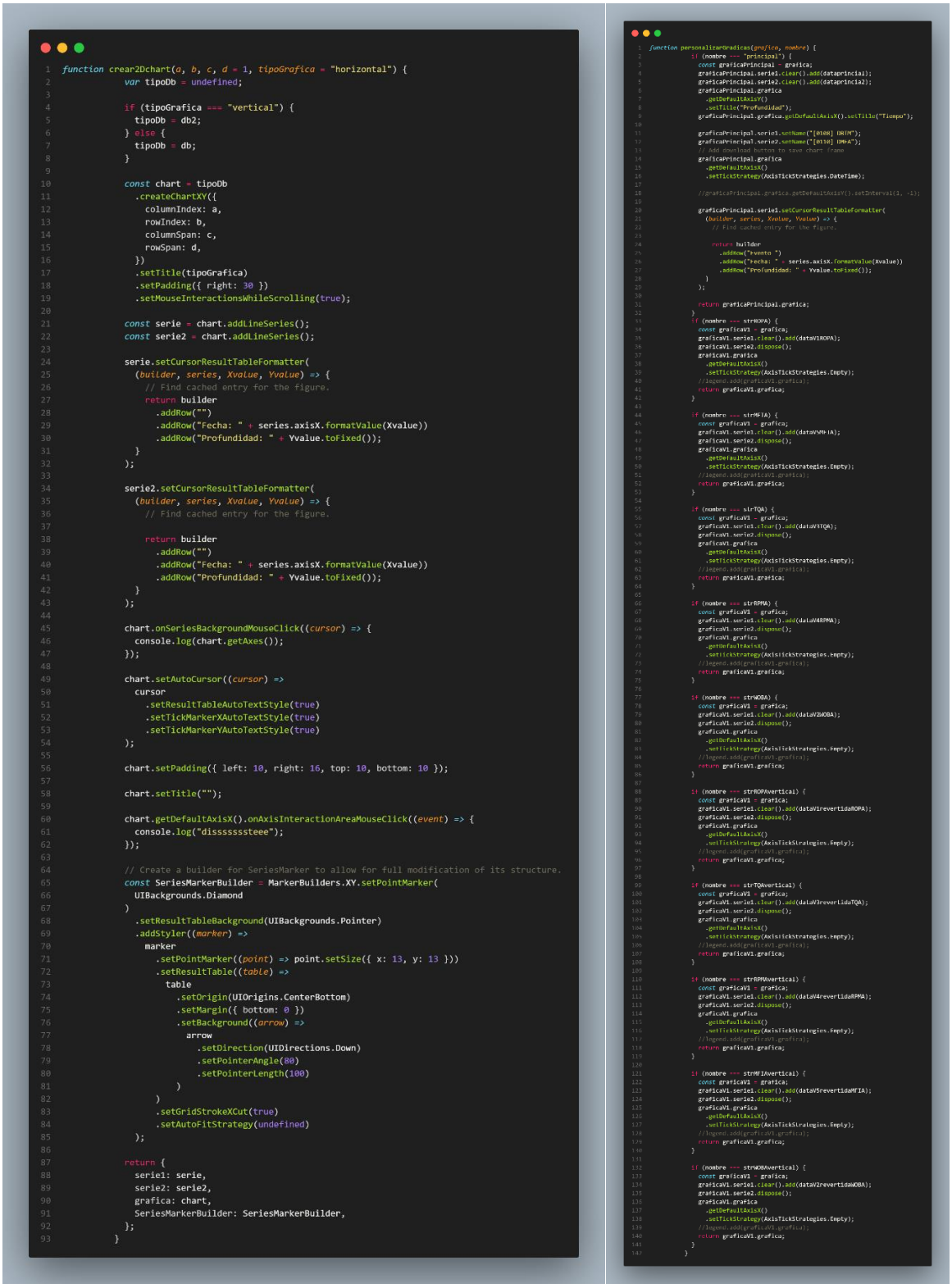
Se continua codificando los botones de navegación que permitirán agregar diferentes gráficas de manera dinámica al visualizador como se observa en la siguiente imagen.

```
const exampleContainer =
            document.getElementById("chart") || document.body;

          const mainDiv = document.createElement("div");
          exampleContainer.append(mainDiv);
          mainDiv.style.position = "absolute";
          mainDiv.style.width = "100%";
          mainDiv.style.height = "100%";
          mainDiv.style.display = "flex";
          mainDiv.style.flexDirection = "column";
          mainDiv.style.color = "black";

          const uiDiv = document.createElement("div");
          mainDiv.append(uiDiv);
          uiDiv.style.display = "flex";
          uiDiv.style.flexDirection = "row";
          uiDiv.style.padding = "10px";
          uiDiv.style.backgroundColor =
            exampleContainer.parentElement.parentElement &&
            window.getComputedStyle(
              exampleContainer.parentElement.parentElement
            ).backgroundColor;
          uiDiv.style.color =
            exampleContainer.parentElement.parentElement &&
            window.getComputedStyle(
              exampleContainer.parentElement.parentElement
            ).color;

          const uiDivTitle = document.createElement("span");
          uiDiv.append(uiDivTitle);
          uiDivTitle.innerHTML = "Click para agregar gráfica";
```

Para la gráfica 3D se realizó una pequeña API que proporcionaba los datos necesarios para la traficación y de igual manera de utilizó la librería de axios para la petición, los datos resultantes se almacenaron en un diccionario para posteriormente realizar la gráfica como se observa a continuación:

```
const getDatosFallas = async (id) => {
                axios
                  .get("http://localhost:3001/datos")
                  .then((response) => {
                    if (response.status === 200) {
                      console.log("OK fallas");
                      console.log(response.data.length);

                      const data3Dfallas = [];

                      for (Let i = 0; i < response.data.length; i++) {
                        data3Dfallas.push({
                          x: response.data[i]["X"],
                          y: response.data[i]["Y"],
                          z: response.data[i]["Z"],
                        });
                      }

                      chart3D.addLegendBox();
                      chart3D.addPointLineSeries().add(data3Dfallas);

                      chart3D.getDefaultAxisX().setTitle("Eje X");
                      chart3D.getDefaultAxisY().setTitle("Eje Y");
                      chart3D.getDefaultAxisZ().setTitle("Eje Z");
                    } else {
                      console.log(
                        "Ocurrió un error consultado los datos de fallas"
                      );
                      console.log(response.data);
                    }
                  })
                  .catch((error) => {
                    console.log(
                      "Ocurrió un error consultado los datos de fallas"
                    );
                    console.log(error.message);
                  });
              };
```

Por otra parte, se creó una función la cual crea una plantilla de grafica en 2D la cual después puede ser personalizada, haciendo uso de otra función que personaliza dicha plantilla de gráfica que ha sido creada:



```
function crear2Dchart(a, b, c, d = 1, tipoGrafica = "horizontal") {
            var tipoDb = undefined;

            if (tipoGrafica === "vertical") {
              tipoDb = db2;
            } else {
              tipoDb = db;
            }

            const chart = tipoDb
              .createChartXY({
                columnIndex: a,
                rowIndex: b,
                columnSpan: c,
                rowSpan: d,
              })
              .setTitle(tipoGrafica)
              .setPadding({ right: 30 })
              .setMouseInteractionsWhileScrolling(true);

            const serie = chart.addLineSeries();
            const serie2 = chart.addLineSeries();

            serie.setCursorResultTableFormatter(
              (builder, series, Xvalue, Yvalue) => {
                // Find cached entry for the figure.
                return builder
                  .addRow("")
                  .addRow("Fecha: " + series.axisX.formatValue(Xvalue))
                  .addRow("Profundidad: " + Yvalue.toFixed());
              }
            );

            serie2.setCursorResultTableFormatter(
              (builder, series, Xvalue, Yvalue) => {
                // Find cached entry for the figure.

                return builder
                  .addRow("")
                  .addRow("Fecha: " + series.axisX.formatValue(Xvalue))
                  .addRow("Profundidad: " + Yvalue.toFixed());
              }
            );

            chart.onSeriesBackgroundMouseClick((cursor) => {
              console.log(chart.getAxes());
            });

            chart.setAutoCursor((cursor) =>
              cursor
                .setResultTableAutoTextStyle(true)
                .setTickMarkerXAutoTextStyle(true)
                .setTickMarkerYAutoTextStyle(true)
            );

            chart.setPadding({ left: 10, right: 16, top: 10, bottom: 10 });

            chart.setTitle("");

            chart.getDefaultAxisX().onAxisInteractionAreaMouseClick((event) => {
              console.log("dissssssteee");
            });

            // Create a builder for SeriesMarker to allow for full modification of its structure.
            const SeriesMarkerBuilder = MarkerBuilders.XY.setPointMarker(
              UIBackgrounds.Diamond
            )
              .setResultTableBackground(UIBackgrounds.Pointer)
              .addStyler((marker) =>
                marker
                  .setPointMarker((point) => point.setSize({ x: 13, y: 13 }))
                  .setResultTable((table) =>
                    table
                      .setOrigin(UIOrigins.CenterBottom)
                      .setMargin({ bottom: 0 })
                      .setBackground((arrow) =>
                        arrow
                          .setDirection(UIDirections.Down)
                          .setPointerAngle(80)
                          .setPointerLength(100)
                      )
                  )
                  .setGridStrokeXCut(true)
                  .setAutoFitStrategy(undefined)
              );

            return {
              serie1: serie,
              serie2: serie2,
              grafica: chart,
              SeriesMarkerBuilder: SeriesMarkerBuilder,
            };
          }
```

```
function personalizarGraficas(grafica, nombre) {
          if (nombre === "principal") {
            const graficaPrincipal = grafica;
            graficaPrincipal.serie1.clear().add(dataprincipal);
            graficaPrincipal.serie2.clear().add(dataprincipal2);
            graficaPrincipal.grafica
              .getDefaultAxisY()
              .setTitle("Profundidad");
            graficaPrincipal.grafica.getDefaultAxisX().setTitle("Tiempo");

            graficaPrincipal.serie1.setName("[0108] 08TM");
            graficaPrincipal.serie2.setName("[0110] 10MA");
            // Add download button to save chart frame
            graficaPrincipal.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.DateTime);

            //graficaPrincipal.grafica.getDefaultAxisY().setInterval(1, -1);

            graficaPrincipal.serie1.setCursorResultTableFormatter(
              (builder, series, Xvalue, Yvalue) => {
                // Find cached entry for the figure.

                return builder
                  .addRow("viento ")
                  .addRow("Fecha: " + series.axisX.formatValue(Xvalue))
                  .addRow("Profundidad: " + Yvalue.toFixed());
              }
            );

            return graficaPrincipal.grafica;
          }
          if (nombre === strROPA) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataV1ROPA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }

          if (nombre === strMFIA) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataVMFIA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }

          if (nombre === strTQA) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataV4TQA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }

          if (nombre === strRPMA) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataV48PMA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }

          if (nombre === strKOBA) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataV2KOBA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }

          if (nombre === strROPAVvertical) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataV1revertidaROPA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }

          if (nombre === strTQAvertical) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataV3reverlidaTQA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }

          if (nombre === strRPMAvertical) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataV4revertidaRPMA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }

          if (nombre === strMFIAvertical) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataV5revertidaMFIA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }

          if (nombre === strKOBAVvertical) {
            const graficaV1 = grafica;
            graficaV1.serie1.clear().add(dataV2revertidaKOBA);
            graficaV1.serie2.dispose();
            graficaV1.grafica
              .getDefaultAxisX()
              .setTickStrategy(AxisTickStrategies.Empty);
            //legend.add(graficaV1.grafica);
            return graficaV1.grafica;
          }
        }
```

Para que el visualizador sea más dinámico se implementa la funcionalidad del tablero la cual permite integrar varias graficas y manipularlas al tiempo. Se crearon

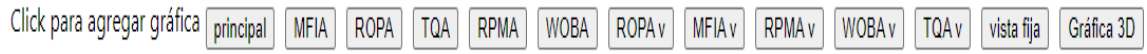
dos tableros, uno para alojar las graficas horizontales y otro para alojar las gráficas verticales:

```
const maxCellsCount = 6;
        const maxCellsCountV = 1;
        // Se cread el Dash
        const db = lightningChart().Dashboard({
          container: chartDiv,
          theme: Themes.glacier,
          numberOfRows: maxCellsCount,
          numberOfColumns: maxCellsCountV,
          disableAnimations: true,
        });

        const db2 = lightningChart().Dashboard({
          container: chartDiv2,
          theme: Themes.glacier,
          numberOfRows: 3,
          numberOfColumns: 5,
          disableAnimations: true,
        });
```

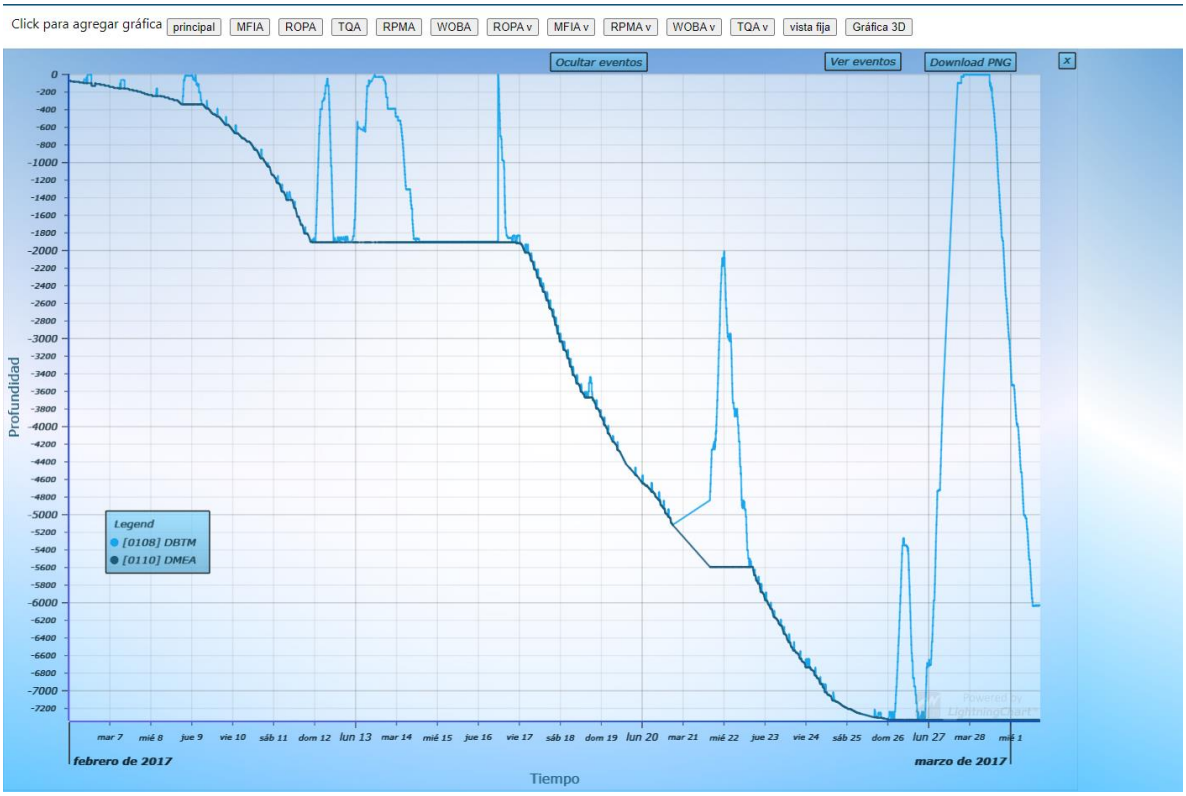
Estás son las principales funcionalidades en el código, de igual manera se pueden detallar mas funcionalidades desde el propio código, como lo fueron la programación de los botones, sincronizar ejes de las gráficas, entre otros.

2. MANUAL DE USUARIO

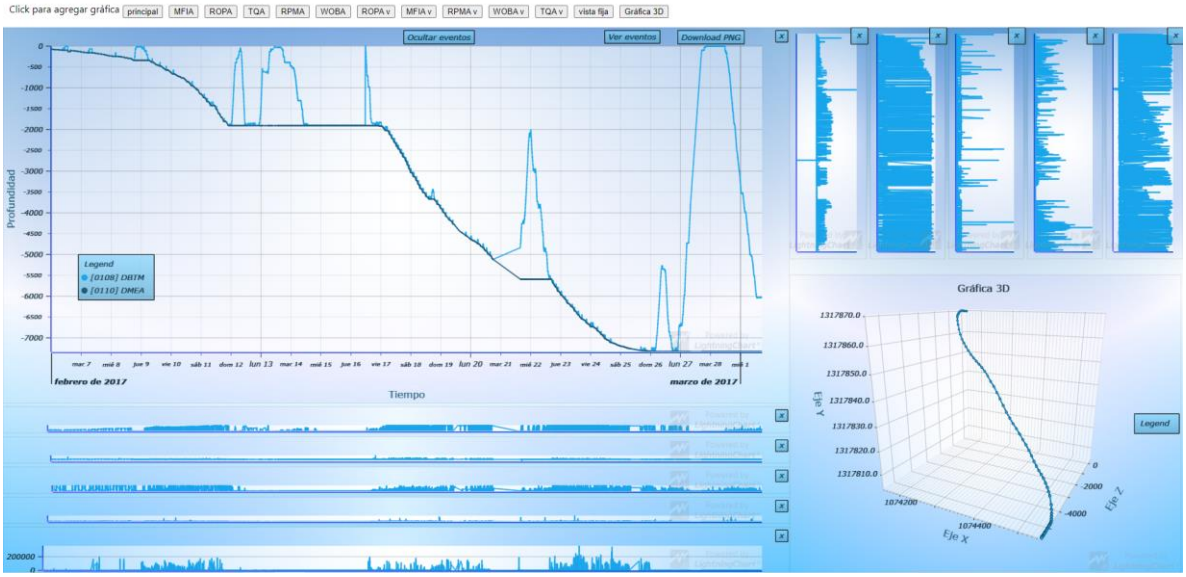
Inicialmente al ejecutar la aplicación el usuario se encontrará con la siguiente imagen:



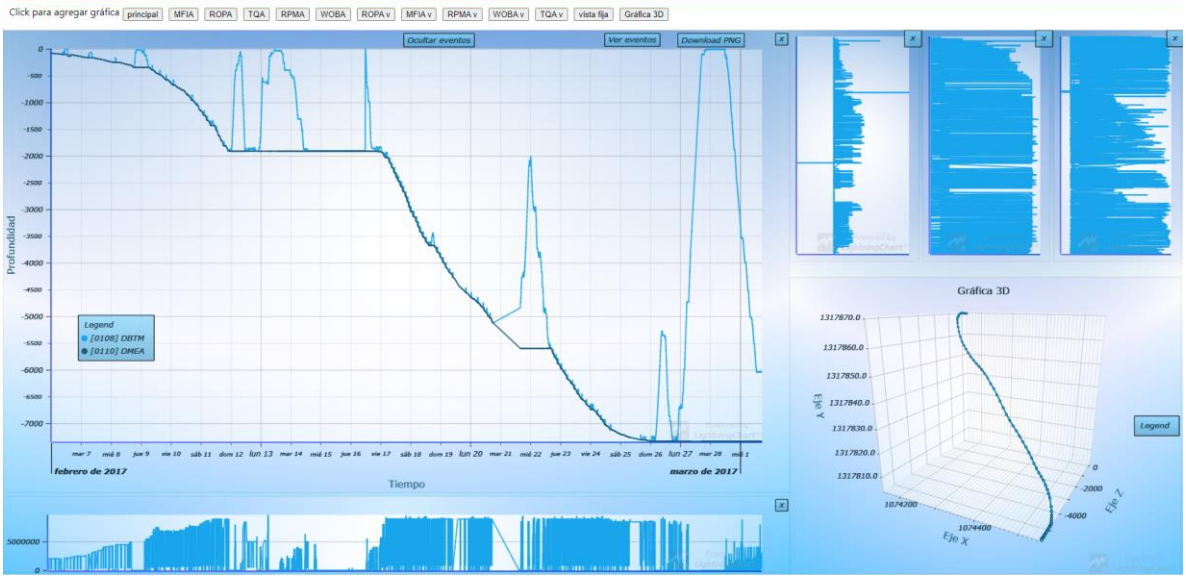
Esta es similar a una barra de navegación, el usuario decide cual gráfica quiere visualizar solo dando click en la deseada, las variables que contienen una “v” al final significa que son gráficas verticales, por ejemplo, si da click en la gráfica “principal” observará lo siguiente:



Y así con las demás gráficas hasta poder observar lo siguiente:



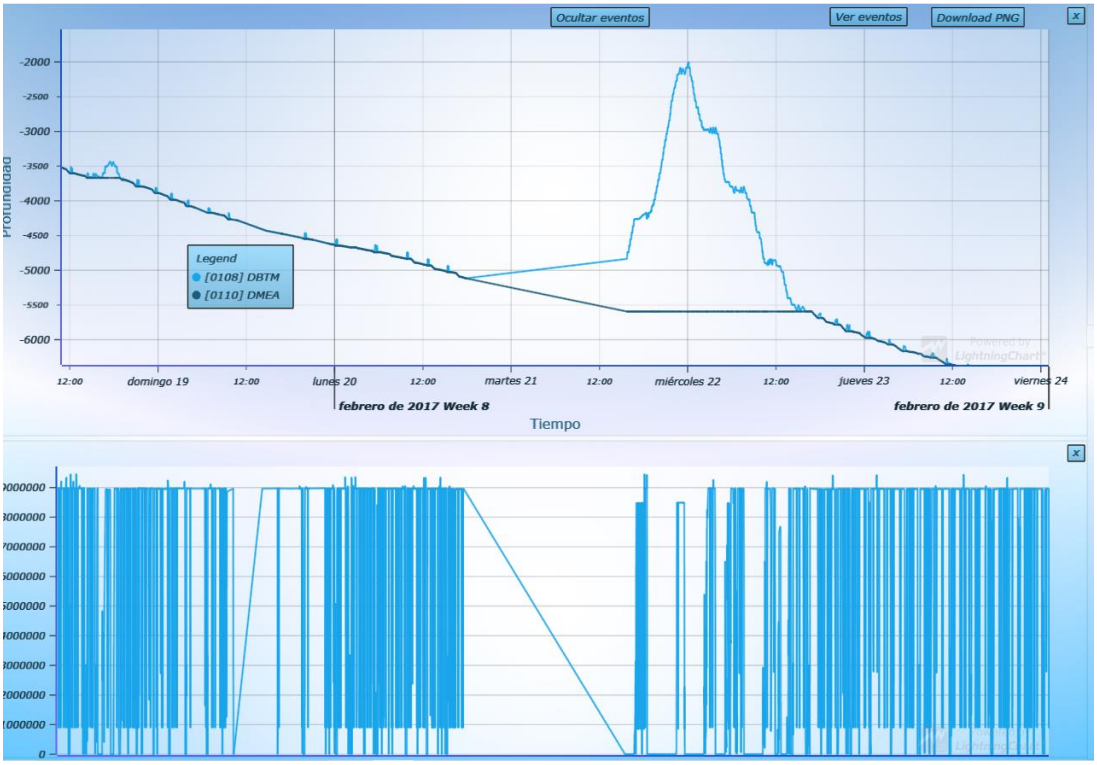
Si el usuario hace click en la “x” de alguna gráfica esta se quitará del tablero y las demás se ajustarán al tablero, esto para tener un aprovechamiento de la pantalla:



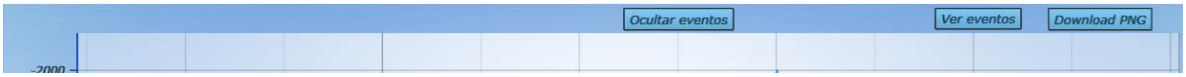
El usuario puede hacer zoom sobre cualquier gráfica pulsando y mantenido el click izquierdo del mouse, además de esto las demás graficas se ajustarán al zoom realizado:

Ilustración 1 Sin realizar zoom

Ilustración 2 Realizando zoom



La gráfica principal cuanta con los siguientes botones:

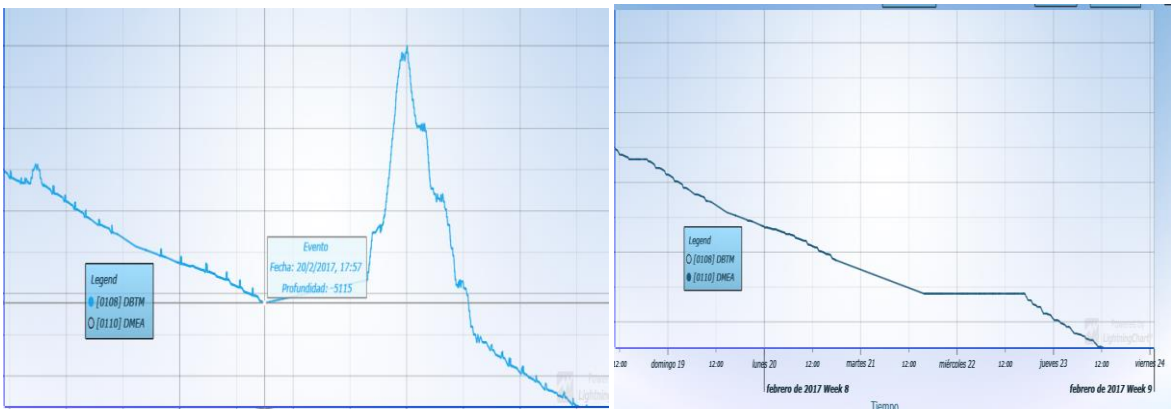


Esto botones se pueden mover sostenido el click derecho del mouse hacia cualquier parte de la gráfica.

- Ver eventos: muestra los eventos del pozo al dar click

- Ocultar eventos: Oculta los eventos
- Download PNG: descarga la gráfica en formato PNG

La gráfica principal tiene una leyenda la cual permite mostar/ocultar una serie para mayor comodidad del usuario:



Este cuadro de leyenda tambien se puede mover a gusto del usuario por cualquier parte de la gráfica.