

SGE - T1 - HITO GRUPAL -

INFORME INDIVIDUAL



Sergio Camino Saiz

09/12/2022



ÍNDICE

INTRODUCCIÓN DEL PROYECTO	2
PARTES REALIZADAS	3
JIRA	18

INTRODUCCIÓN DEL PROYECTO

Hemos creado un proyecto el cual permite ingresar productos a una base de datos, permite consultar esos productos, nos permite filtrar los productos, actualizar y eliminar los productos, importar los productos consultados a un archivo de tipo excel, crear y eliminar tablas de la base de datos y mostrar el stock de los productos en forma de gráficos. El problema que hemos resuelto era crear una aplicación para un supermercado en la que pudiera manejar los productos del supermercado.

En mi grupo una persona se encargaba de ser SCRUM MASTER, esa persona se encargaba de gestionar al equipo y de guiarle. La persona elegida fue Javier Sanchez. Javier repartió de forma equitativa las tareas a realizar y cuando surgen problemas Javier era el primero en ayudar.

Partes realizadas por mis compañeros :

Javier Sanchez : Interfaz 'ListarProductos', interfaz 'Crear/Eliminar Tabla', función 'read_Productos', función 'tc' (crear tabla), función 'dc' (eliminar tabla) y la interfaz de 'gráfico productos'.

Astrid Cruces : Interfaz actualizar producto, interfaz crear/eliminar tabla, función 'update_productos', función 'tc' (crear tabla) y función 'dc' (eliminar tabla)

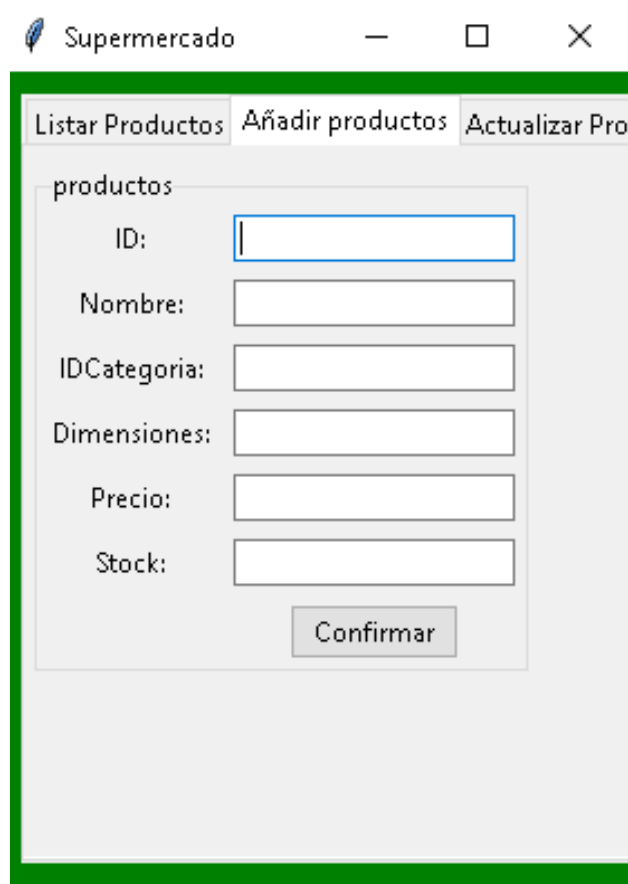
Isla Peinado : Interfaz eliminar producto, interfaz filtrar producto, conexión con bd, creación de bd, función conexión a bd, función 'delete_productos', función 'select_product' (asc y desc).

PARTES REALIZADAS

- Interfaz ‘Añadir producto’ con la función ‘insertProductos’ :

La interfaz la he creado con el módulo Tkinter. He creado cajas para que el usuario pueda almacenar los datos que quiera y después poder almacenarlos en la base de datos. La función ‘insertar_producto’ inserta el producto y la función ‘insertProductos2’ lo que hace es eliminar los campos introducidos y dejarlos en blanco.

Parte visual :



The image shows a screenshot of a Tkinter application window titled "Supermercado". The window has a green border and standard window controls (minimize, maximize, close). Inside the window, there are three tabs: "Listar Productos", "Añadir productos", and "Actualizar Pro". The "Añadir productos" tab is currently selected. Below the tabs, there is a section labeled "productos" which contains a form with the following fields and labels: "ID:", "Nombre:", "IDCategoria:", "Dimensiones:", "Precio:", and "Stock:". Each label is followed by a text input field. At the bottom of the form, there is a "Confirmar" button. The entire form is enclosed in a light gray box.

Parte de código :

Visual

```
# Añadir Producto
def insert_Productos(self):
    self.pagina1 = ttk.Frame(self.cuaderno1)
    self.cuaderno1.add(self.pagina1, text="Añadir productos")
    self.labelframe1 = ttk.LabelFrame(self.pagina1, text="productos")
    self.labelframe1.grid(column=0, row=0, padx=5, pady=10)

    self.label1 = ttk.Label(self.labelframe1, text="ID:")
    self.label1.grid(column=0, row=0, padx=4, pady=4)
    self.idcarga = tk.StringVar()
    self.entryid = ttk.Entry(self.labelframe1, textvariable=self.idcarga)
    self.entryid.grid(column=1, row=0, padx=4, pady=4)

    self.label2 = ttk.Label(self.labelframe1, text="Nombre:")
    self.label2.grid(column=0, row=1, padx=4, pady=4)
    self.nombrecarga = tk.StringVar()
    self.entrynombre = ttk.Entry(self.labelframe1, textvariable=self.nombrecarga)
    self.entrynombre.grid(column=1, row=1, padx=4, pady=4)

    self.label3 = ttk.Label(self.labelframe1, text="IDCategoria:")
    self.label3.grid(column=0, row=2, padx=4, pady=4)
    self.idcatcarga2 = tk.StringVar()
    self.entryidcat = ttk.Entry(self.labelframe1, textvariable=self.idcatcarga2)
    self.entryidcat.grid(column=1, row=2, padx=4, pady=4)

    self.label4 = ttk.Label(self.labelframe1, text="Dimensiones:")
    self.label4.grid(column=0, row=3, padx=4, pady=4)
    self.dimencarga = tk.StringVar()
    self.entrydimen = ttk.Entry(self.labelframe1, textvariable=self.dimencarga)
    self.entrydimen.grid(column=1, row=3, padx=4, pady=4)

    self.label5 = ttk.Label(self.labelframe1, text="Precio:")
    self.label5.grid(column=0, row=4, padx=4, pady=4)
    self.preccarga = tk.StringVar()
    self.entriyprec = ttk.Entry(self.labelframe1, textvariable=self.preccarga)
    self.entriyprec.grid(column=1, row=4, padx=4, pady=4)

    self.label6 = ttk.Label(self.labelframe1, text="Stock:")
    self.label6.grid(column=0, row=5, padx=4, pady=4)
    self.stockcarga = tk.StringVar()
    self.entriystock = ttk.Entry(self.labelframe1, textvariable=self.stockcarga)
    self.entriystock.grid(column=1, row=5, padx=4, pady=4)

    self.boton1 = ttk.Button(self.labelframe1, text="Confirmar", command=self.insertProductos2)
    self.boton1.grid(column=1, row=6, padx=4, pady=4)
```

Lógica de las funciones

```
# Añadir
def insertProductos(self, supermercadoList):
    con = self.conexion()
    cur = con.cursor()
    instruccion = f"INSERT INTO producto VALUES(?,?,?,?,?,?)"
    cur.execute(instruccion,supermercadoList)
    con.commit()
    con.close()
```

```
def insertProductos2(self):
    datos = (self.idcarga.get(), self.nombrecarga.get(), self.idcatcarga2.get(), self.dimencarga.get(),
self.preccarga.get(), self.stockcarga.get())
    self.producto1.insertProductos(datos)
    mb.showinfo("Información", "Los datos fueron cargados")
    self.idcarga=""
    self.nombrecarga=""
    self.idcatcarga2=""
    self.dimencarga=""
    self.preccarga=""
    self.stockcarga=""
```

- Interfaz 'filtrar productos' con las funciones 'filtasc' y 'filtdesc':

Esta parte la hemos hecho entre mi compañera Isla y yo. La interfaz de filtrar productos funciona de la siguiente manera. Escribimos la columna que queremos filtrar y mediante 2 funciones podemos filtrar de manera ascendente o descendente.

Parte visual :

The image shows a web application interface for filtering products. At the top, there is a horizontal tabbed menu with five tabs: 'Listar Productos', 'Añadir productos', 'Actualizar Productos', 'Eliminar Producto', and 'Filtrar Productos'. The 'Filtrar Productos' tab is currently selected. Below the tabs, the main content area is titled 'Filtro'. It contains a label 'Columna a filtrar:' followed by a text input field. Below the input field are two buttons: 'Filtrar Ascendente' and 'Filtrar Descendente'. At the bottom of the 'Filtro' section is a large, empty rectangular box with a vertical scrollbar on its right side, intended for displaying the filtered results.

Parte de código :

Visual

```
def select_Product(self):
    self.pagina9 = ttk.Frame(self.cuaderno1)
    self.cuaderno1.add(self.pagina9, text="Filtrar Productos")
    self.labelframe9 = ttk.LabelFrame(self.pagina9, text="Filtro")
    self.labelframe9.grid(column=0, row=1, padx=5, pady=10)
    self.scrolledtext1 = st.ScrolledText(self.labelframe9, width=30, height=10)
    self.scrolledtext1.grid(column=0, row=9, padx=10, pady=10)
```

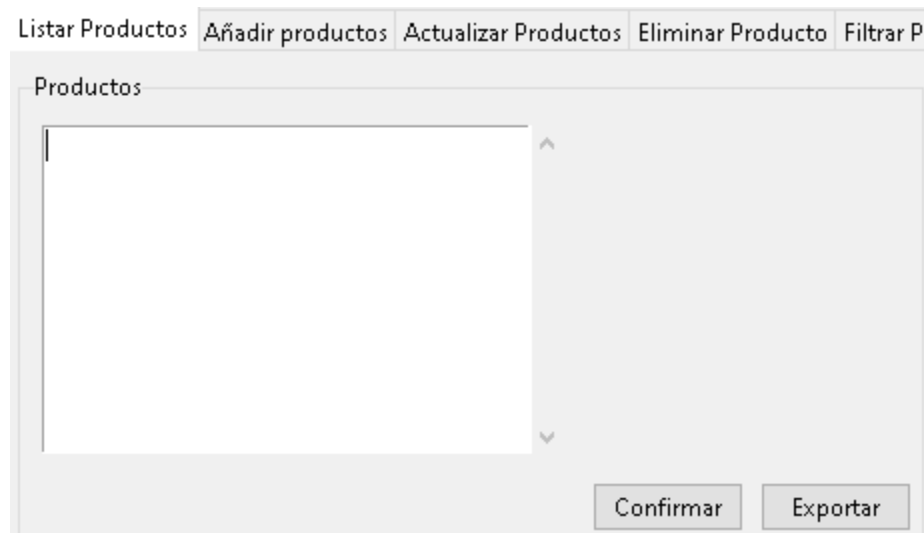
Lógica de las funciones

```
def filtasc(self):
    respuesta=self.producto1.readOrderedMayorProducto(self.idcol.get())
    self.scrolledtext1.delete("1.0", tk.END)
    for fila in respuesta:
        self.scrolledtext1.insert(tk.END, "id:" + str(fila[0]) + "\nnombre:" + fila[1] + "\nidcategoria: " + str(
            fila[2]) + "\nmedida: " + fila[3] + "\nprecio: " + str(fila[4]) + "\nstock: " + str(fila[5]) + "\n")
def filtdesc(self):
    respuesta=self.producto1.readOrderedMenorProducto(self.idcol.get())
    self.scrolledtext1.delete("1.0", tk.END)
    for fila in respuesta:
        self.scrolledtext1.insert(tk.END, "id:" + str(fila[0]) + "\nnombre:" + fila[1] + "\nidcategoria: " + str(
            fila[2]) + "\nmedida: " + fila[3] + "\nprecio: " + str(fila[4]) + "\nstock: " + str(fila[5]) + "\n")
```


- Exportar a excel :

Esta función nos permite exportar los datos que nos ha pintado el programa en la interfaz 'Listar productos' a un archivo excel. En la interfaz hay un botón llamado 'exportar' y ese nos los exporta.

Parte visual :



Parte de código :

Visual

```
self.boton2 = ttk.Button(self.labelframe2, text="Exportar", command=self.exportarExcel)
self.boton2.grid(column=2, row=6, padx=4, pady=4)
```

Lógica de las funciones

```
Exportar excel - Exportar la consulta (resultado)
def exportExcel(self, respuesta):

    con = self.conexion()
    cur = con.cursor()

    listaProductos = []
    #Recibir productos de la lista productos (de la parte visual).
    for producto in respuesta:
        print(producto)
        listaProductos.append(producto)

    #Guardar los productos en un dataframe.
    dtProductos = pd.DataFrame(listaProductos)

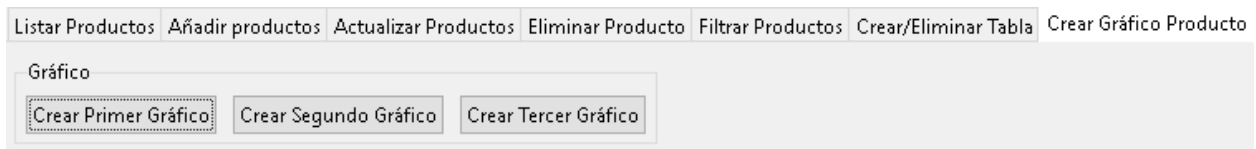
    #Guardar el dataframe en un excel.
    writer = pd.ExcelWriter('datos_lista.xls')
    dtProductos.to_excel(writer, sheet_name='productos')
    writer.save()

    con.commit()
    con.close()
```

- Gráficos :

He creado 3 tipos distintos de gráficos; barras horizontales (barh), barras verticales (bar) y puntos (scatter). Cada uno contiene el nombre del producto y el stock del mismo.

Parte visual :



Parte de código :

Visual

```
def creategraf(self):
    self.pagina5 = ttk.Frame(self.cuaderno1)
    self.cuaderno1.add(self.pagina5, text="Crear Gráfico Producto")
    self.labelframe8 = ttk.LabelFrame(self.pagina5, text="Gráfico")
    self.labelframe8.grid(column=0, row=0, padx=5, pady=10)
    self.boton1 = ttk.Button(self.labelframe8, text="Crear Primer Gráfico", command=self.g1)
    self.boton1.grid(column=1, row=6, padx=4, pady=4)

    self.boton2 = ttk.Button(self.labelframe8, text="Crear Segundo Gráfico", command=self.g2)
    self.boton2.grid(column=2, row=6, padx=4, pady=4)

    self.boton3 = ttk.Button(self.labelframe8, text="Crear Tercer Gráfico", command=self.g3)
    self.boton3.grid(column=3, row=6, padx=4, pady=4)
```

```
def g1(self):
    respuesta = self.producto1.readProducto()
    self.producto1.graficoGenerico(respuesta)

def g2(self):
    respuesta = self.producto1.readProducto()
    self.producto1.graficoPie(respuesta)

def g3(self):
    respuesta = self.producto1.readProducto()
    self.producto1.graficoHistograma(respuesta)
```

Lógica de funciones

Gráfico 1 :

```
#Gráfico generico - BAR
def graficoGenerico(self, respuesta):
    con = self.conexion()
    cur = con.cursor()

    # Stock.
    # Stock producto 1.
    instruccion1 = f"SELECT nombre FROM producto WHERE producto.nombre = 'Café de Malasia'"
    cur.execute(instruccion1)
    producto1 = cur.fetchall()

    # Stock producto 2.
    instruccion2 = f"SELECT nombre FROM producto WHERE producto.nombre = 'Pez espada'"
    cur.execute(instruccion2)
    producto2 = cur.fetchall()

    # Stock producto 3.
    instruccion3 = f"SELECT nombre FROM producto WHERE producto.nombre = 'Sirope de arce'"
    cur.execute(instruccion3)
    producto3 = cur.fetchall()
```

```
#Stock.
#Stock producto 1.
instruccion1 = f"SELECT stock FROM producto WHERE producto.nombre = 'Café de Malasia'"
cur.execute(instruccion1)
stock1 = cur.fetchall()

#Stock producto 2.
instruccion2 = f"SELECT stock FROM producto WHERE producto.nombre = 'Pez espada'"
cur.execute(instruccion2)
stock2 = cur.fetchall()

#Stock producto 3.
instruccion3 = f"SELECT stock FROM producto WHERE producto.nombre = 'Sirope de arce'"
cur.execute(instruccion3)
stock3 = cur.fetchall()

#productos = [producto1, producto2, producto3]
#stock = [stock1, stock2, stock3]
```

```

#DISEÑO GRÁFICO
sec = tk.Tk()
sec.title("Gráfico")
sec.config(width=800, height=400)

figure = Figure(figsize=(6, 4), dpi=100)

# create FigureCanvasTkAgg object
figure_canvas = FigureCanvasTkAgg(figure, master=sec)
figure_canvas.draw()

# create the toolbar
NavigationToolbar2Tk(figure_canvas, sec)
matplotlib.pyplot.subplots()
matplotlib.pyplot.bar("Café de Malasia", 17)
matplotlib.pyplot.bar("Pez espada", 31)
matplotlib.pyplot.bar("Siropo de arce", 113)
matplotlib.pyplot.xlabel("Productos")
matplotlib.pyplot.ylabel("Stock")
matplotlib.pyplot.title("Stock de productos")
matplotlib.pyplot.show()

figure_canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

sec.mainloop()

con.commit()
con.close()

```

Gráfico 2 :

```
# PIE
def graficoPie(self, respuesta):
    #pie
    con = self.conexion()
    cur = con.cursor()

    # Stock.
    # Stock producto 1.
    instruccion1 = f"SELECT nombre FROM producto WHERE producto.nombre = 'Café de Malasia'"
    cur.execute(instruccion1)
    producto1 = cur.fetchall()

    # Stock producto 2.
    instruccion2 = f"SELECT nombre FROM producto WHERE producto.nombre = 'Pez espada'"
    cur.execute(instruccion2)
    producto2 = cur.fetchall()

    # Stock producto 3.
    instruccion3 = f"SELECT nombre FROM producto WHERE producto.nombre = 'Siropo de arce'"
    cur.execute(instruccion3)
    producto3 = cur.fetchall()
```

```
#Stock.
#Stock producto 1.
instruccion1 = f"SELECT stock FROM producto WHERE producto.nombre = 'Café de Malasia'"
cur.execute(instruccion1)
stock1 = cur.fetchall()

#Stock producto 2.
instruccion2 = f"SELECT stock FROM producto WHERE producto.nombre = 'Pez espada'"
cur.execute(instruccion2)
stock2 = cur.fetchall()

#Stock producto 3.
instruccion3 = f"SELECT stock FROM producto WHERE producto.nombre = 'Siropo de arce'"
cur.execute(instruccion3)
stock3 = cur.fetchall()

#productos = [producto1, producto2, producto3]
#stock = [stock1, stock2, stock3]
```

```

#DISEÑO GRÁFICO
sec = tk.Tk()
sec.title("Gráfico")
sec.config(width=800, height=400)

figure = Figure(figsize=(6, 4), dpi=100)

# create FigureCanvasTkAgg object
figure_canvas = FigureCanvasTkAgg(figure, master=sec)
figure_canvas.draw()

# create the toolbar
NavigationToolbar2Tk(figure_canvas, sec)
matplotlib.pyplot.subplots()
matplotlib.pyplot.scatter("Café de Malasia", 17)
matplotlib.pyplot.scatter("Pez espada", 31)
matplotlib.pyplot.scatter("Siropo de arce", 113)
matplotlib.pyplot.xlabel("Productos")
matplotlib.pyplot.ylabel("Stock")
matplotlib.pyplot.title("Stock de productos")
matplotlib.pyplot.show()

figure_canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

sec.mainloop()

con.commit()
con.close()

```


Gráfico 3 :

```
# HISTOGRAMA
def graficoHistograma(self, respuesta):
    #fill_between
    con = self.conexion()
    cur = con.cursor()

    # Stock.
    # Stock producto 1.
    instruccion1 = f"SELECT nombre FROM producto WHERE producto.nombre = 'Café de Malasia'"
    cur.execute(instruccion1)
    producto1 = cur.fetchall()

    # Stock producto 2.
    instruccion2 = f"SELECT nombre FROM producto WHERE producto.nombre = 'Pez espada'"
    cur.execute(instruccion2)
    producto2 = cur.fetchall()

    # Stock producto 3.
    instruccion3 = f"SELECT nombre FROM producto WHERE producto.nombre = 'Siropo de arce'"
    cur.execute(instruccion3)
    producto3 = cur.fetchall()
```

```
#Stock.
#Stock producto 1.
instruccion1 = f"SELECT stock FROM producto WHERE producto.nombre = 'Café de Malasia'"
cur.execute(instruccion1)
stock1 = cur.fetchall()

#Stock producto 2.
instruccion2 = f"SELECT stock FROM producto WHERE producto.nombre = 'Pez espada'"
cur.execute(instruccion2)
stock2 = cur.fetchall()

#Stock producto 3.
instruccion3 = f"SELECT stock FROM producto WHERE producto.nombre = 'Siropo de arce'"
cur.execute(instruccion3)
stock3 = cur.fetchall()

#productos = [producto1, producto2, producto3]
#stock = [stock1, stock2, stock3]
```

```

#DISEÑO GRÁFICO
sec = tk.Tk()
sec.title("Gráfico")
sec.config(width=800, height=400)

frame = tk.Frame(sec)

figure = Figure(figsize=(6, 4), dpi=100)

# create FigureCanvasTkAgg object
figure_canvas = FigureCanvasTkAgg(figure, master=frame)
figure_canvas.draw()

# create the toolbar
NavigationToolbar2Tk(figure_canvas, frame)
matplotlib.pyplot.subplots()
matplotlib.pyplot.barh("Café de Malasia", 17)
matplotlib.pyplot.barh("Pez espada", 31)
matplotlib.pyplot.barh("Siropo de arce", 113)
matplotlib.pyplot.xlabel("Productos")
matplotlib.pyplot.ylabel("Stock")
matplotlib.pyplot.title("Stock de productos")
matplotlib.pyplot.show()

figure_canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

frame.mainloop()

sec.pack()
sec.mainloop()

```

JIRA

He asumido el rol de developer team. Las tareas que se me han asignado son las siguientes :

The screenshot shows four Jira task cards. Each card has a title, a key, a status, and an assignee. The tasks are: 'Interfaz Añadir' (HS1-18), 'Interfaz Creación de Gráficos' (HS1-26), 'Exportar a Excel' (HS1-24), and 'Interfaz Filtrar' (HS1-25). All tasks are marked as 'Done' with a green checkmark and assigned to 'SS'.

Task Name	Key	Status	Assignee
Interfaz Añadir	HS1-18	Done	SS
Interfaz Creación de Gráficos	HS1-26	Done	SS
Exportar a Excel	HS1-24	Done	SS
Interfaz Filtrar	HS1-25	Done	I

La tarea de 'Filtrar' está encargada a una sola persona porque Jira no nos permite asignar a más de una persona para la misma tarea. Pero en realidad nos hemos encargado mi compañera Isla y yo de la tarea.

▼ Memoria 25 nov – 4 dic (10 incidencias)		0 0 0 Completar sprint ...
HS1-23	Base de datos(Conexión, Creación)	FINALIZADA ✓ I
HS1-17	Interfaz Listar	FINALIZADA ✓ JC
HS1-18	Interfaz Añadir	FINALIZADA ✓ SS
HS1-19	Interfaz Actualizar	FINALIZADA ✓ AH
HS1-20	Interfaz Eliminar	FINALIZADA ✓ I
HS1-25	Interfaz Filtrar	FINALIZADA ✓ I
HS1-21	Interfaz Crear Tabla	FINALIZADA ✓ AH
HS1-22	Interfaz Eliminar Tabla	FINALIZADA ✓ JC
HS1-26	Interfaz Creación de Gráficos	FINALIZADA ✓ SS
HS1-24	Exportar a Excel	FINALIZADA ✓ SS

Interfaz Añadir

[Adjuntar](#) [Añadir una incidencia secundaria](#) [Vincular incidencia](#) [▼](#)

Descripción

Crear métodos para añadir:

1. insert_Productos()
2. insertProductos2()
3. insertProductos()

Actividad

Mostrar: [Todo](#) [Comentarios](#) [Historial](#)

Más recientes primero

Consejo de expertos: pulsa para comentar

Interfaz Creación de Gráficos

[Adjuntar](#) [Añadir una incidencia secundaria](#) [Vincular incidencia](#) [▼](#)

Descripción

Hacer métodos para crear gráficos:

1. creategraf()
2. g1()
3. gráficogenérico()
4. g2()
5. gráficoPie()
6. g3()
7. gráficoHistogramas()

Exportar a Excel

[Adjuntar](#) [Añadir una incidencia secundaria](#) [Vincular incidencia](#) [▼](#)

Descripción

Métodos para exportar a Excel:

1. exportarExcel()

Actividad

Mostrar: [Todo](#) [Comentarios](#) [Historial](#)

Más recientes primero

Consejo de expertos: pulsa para comentar

Finalizada Listo

Detalles	
Responsable	SERGIO CAMINO SAIZ
Etiquetas	Ninguno
Sprint	Memoria
Story point estimate	Ninguno
Desarrollo	Crear rama Crear confirmación
Informador	JAVIER SANCHEZ CARRIZO

Creado hace 1 hora

Configurar

Finalizada Listo

Detalles	
Responsable	SERGIO CAMINO SAIZ
Etiquetas	Ninguno
Sprint	Memoria
Story point estimate	Ninguno
Desarrollo	Crear rama Crear confirmación
Informador	JAVIER SANCHEZ CARRIZO

Finalizada Listo

Detalles	
Responsable	SERGIO CAMINO SAIZ
Etiquetas	Ninguno
Sprint	Memoria
Story point estimate	Ninguno
Desarrollo	Crear rama Crear confirmación
Informador	JAVIER SANCHEZ CARRIZO

Interfaz Filtrar

- Adjuntar
- Añadir una incidencia secundaria
- Vincular incidencia
-

Descripción

Crear métodos de filtrado:

1. select_Product()
2. filtasc()
3. readOrderedMayorProducto()
4. filtdesc()
5. readOrderedMenorProducto()

Actividad

Mostrar:

TodoComentariosHistorial

Más recientes primero

Finalizada

✓ Listo

Detalles

Responsable	<div>ISLA PEINADO HENRIQUEZ</div> <div>Asignarme a mí</div>
Etiquetas	Ninguno
Sprint	Memoria
Story point estimate	Ninguno
Desarrollo	<div>Crear rama</div> <div>Crear confirmación</div>
Informador	<div>JAVIER SANCHEZ CARRIZO</div>