Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de Ciencias Computacionales.

Ingeniería en Computación

Actividad 9.

Programa 4. Algoritmo de Planificación FCFS (First Come First Server) Continuación

Materia: Sistemas Operativos ***Sección:*** D04

Profesor: Becerra Velázquez Violeta del Rocío.

Equipo:

- Carriola Navarro Aldo Fernando | 221978167
- Cervantes Araujo Maria Dolores | 217782452

Fecha: 22/10/2023

Índice

Tabla de Imágenes

Programa 4. FCFS Continuación

Objetivos

Este cuarto programa, se busca incorporar dos teclas más a comparación del programa anterior, dichas teclas serán:

- Tecla ‘N’: La tecla ‘nuevo’ donde al presionarla se generará un nuevo proceso, creando con ello los datos necesarios de forma aleatoria. El planificador a largo plazo es el que definirá su ingreso.
- Tecla ‘B’: La tecla ‘tabla de procesos’ donde el programa se pausará y se deberá visualizar la tabla de procesos, es decir los BCP de cada uno de los procesos. Con la tecla “C” continua la simulación de su programa en el punto donde quedó.

Siendo así que la tabla BCP de cada proceso, seguirá mostrando lo mismo que la tabla final del programa anterior, a excepción que ahora se tendrá en consideración en que estado se encuentra cada proceso, para así poder determinar que datos poder mostrar en la tabla de procesos cada que el usuario tecleé ‘B’.

Notas Sobre el Lenguaje

El lenguaje utilizado para llevar a cabo el programa fue *Python*, dado que nos permite experimentar con diversas funciones que apoyan en la creación del programa, permitiéndonos que sea orientado a objetos, estructural, con interfaz, a consola, con color e incluso animación.

Dado que es un lenguaje de nuestro total conocimiento y dominio para ambos, optamos por crear el programa de forma estructural, es decir, mediante funciones. Buscando siempre mediante acuerdos, que ambos nos sintiéramos familiarizados con el lenguaje y lógica del programa, de esta manera optimizamos tiempos y nos fue más sencillo modelar y visualizar el programar conforme lo esperado.

TDA o Estructuras Utilizadas en el Programa

Siguiendo la estructura del programa anterior y tomando en cuenta que esta actividad es una continuación, solo fue necesario complementar dos funciones del algoritmo mostrado anteriormente.

Modificamos la función de *impresión final* para poder implementar algunas condicionales integradas para las funciones en tecla sobre los procesos:

```
def impresion_final():
    lista_final=[]
    copia_terminados = copy.deepcopy(procesos_terminados)
    copia_ejecucion = copy.deepcopy(proceso_ejecucion)
    copia_actuales = copy.deepcopy(procesos_actuales)
    copia_bloqueados = copy.deepcopy(procesos_bloqueados)
    copia_pendiente = copy.deepcopy(procesos_pendientes)

    if copia_terminados:
        for x in copia_terminados:
            lista_final.append(x)

    if copia_ejecucion:
        for x in copia_ejecucion:
            x[2] = ''
            x[12] = ''
            x[11] = contadorGeneral - x[7]
            x[9] = x[11]-x[8]

            lista_final.append(x)

    if copia_bloqueados:
        for x in copia_bloqueados:
            x[2] = ''
            x[12] = ''
            x[11] = contadorGeneral - x[7]
            x[9] = x[11]-x[8]

            lista_final.append(x)

    if copia_actuales:
        for x in copia_actuales:
            if x[13] == 1:
                x[2] = ''
                x[12] = ''
                x[11] = contadorGeneral - x[7]
                x[9] = x[11]-x[8]
```

Ilustración 1. Función impresión_final modificada 1.

Programa 4. FCFS Continuación

```
        else:
            x[2] = ''
            x[4] = ''
            x[5] = ''
            x[12] = ''
            x[14] = ''
            x[11] = contadorGeneral - x[7]
            x[9] = x[11]-x[8]
            x[8] = ''
            x[10] = ''

        lista_final.append(x)

if copia_pendiente:
    for x in copia_pendiente:
        x[2] = ''
        x[4] = ''
        x[5] = ''
        x[14] = ''
        x[7] = ''
        x[8] = ''
        x[9] = ''
        x[10] = ''
        x[11] = ''
        x[12] = ''
        lista_final.append(x)

print("\n\n\nProcesos")
headersB = ["ID", "Ope", "Res", "TME", "TT", "TR", "TB", "TLl", "TS", "TE", "TRes", "TRet", "TF"]
if lista_final:
    procesoGen = [[dato[0], dato[1], dato[2], dato[3], dato[4], dato[5], dato[14], dato[7], dato[8], dato[9], dato[10], dato[11], dato[12]] for dato in lista_final]
    print(tabulate(procesoGen, headers=headersB, tablefmt="double_outline"))
    print("\nContador General:",contadorGeneral)
```

Ilustración 2. Función impresión_final modificada 2.

Estas condicionales nos permitirán capturar datos temporales de los procesos según los solicite el usuario, tomando en consideración los 5 escenarios por los que pasa o puede pasar u proceso, siendo: “procesos pendientes/nuevos”, “procesos listos”, “procesos en ejecución”, “procesos bloqueados” y “procesos “terminados”, creando nuevas operaciones para ciertos campos como el tiempo de espera o el tiempo de retorno.

```
def intoTecla():
    global tecla

    tecla='q'
    if mv.kbhit():
        tecla = mv.getch().decode('utf-8')
        if tecla == 'p':
            os.system('cls')
            imprimir()
            print("Este programa se encuentra pausado...")
            while tecla != 'c':
                try:
                    tecla = mv.getch().decode('utf-8')
                    if tecla == 'c':
                        print("Reanudando Programa...")
                        t.sleep(1)
                        os.system('cls')
                except UnicodeDecodeError:
                    pass

        elif tecla == 'b':
            os.system('cls')
            impresion_final()
            print("Presiona c para continuar con la ejecución...")
            while tecla != 'c':
                try:
                    tecla = mv.getch().decode('utf-8')
                    if tecla == 'c':
                        print("Reanudando Programa...")
                        t.sleep(1)
                        os.system('cls')
                except UnicodeDecodeError:
                    pass
```

Otro cambio fue en la función de *intoTecla* debido a que se agrego un nuevo funcionamiento en caso de que el usuario presione “b”, quedando declarado de la siguiente manera:

Ilustración 3. Función intoTecla modificada.

Por último, fue necesario agregar una condición más dentro de la función principal *main*, para que el programa sea capaz de agregar n procesos desee el usuario en caso de presionar “n”:

```
elif tecla == 'n':
    ID()
    isOperador()
    tiempo_estimado()
    info = [id, operaciones, resultado, TME, contTiempo, Tr, tiempoBloq, tiempoLlegada, tiempoServicio, tiempoEspera, tiempoRespuesta, tiempoRetorno, tiempoFinal,
            bandera, tiempoTotalBloqueado]
    procesos_pendientes.append(info)
    numProcess += 1
    break
```

Ilustración 4. Condición extra en función main.

Funcionamiento

A continuación, se presentan algunas ilustraciones que reflejan el funcionamiento del programa de simulación con las mejoras:

1. Primero el programa pregunta los n procesos:

```
Dame un número de procesos: 12
```

Ilustración 5. Inicio del programa.

2. Se muestra la tabla con tiempos temporales:

Procesos

ID	Ope	Res	TME	TT	TR	TB	TLl	TS	TE	TRes	TRet	TF
1	61**32		8	2	6	0	0	2	0	0	2	
2	16-52		6				0		2		2	
3	79%23		18				0		2		2	
4	72%39		7				0		2		2	
5	19**64		12				0		2		2	
6	70-39		11									
7	5**32		11									
8	5**11		18									
9	89%9		14									
10	60/61		15									
11	76*9		9									
12	35/100		12									

```
Contador General: 2
Presiona c para continuar con la ejecución...
```

Ilustración 6. Datos temporales 1

3. Se bloquea un proceso:

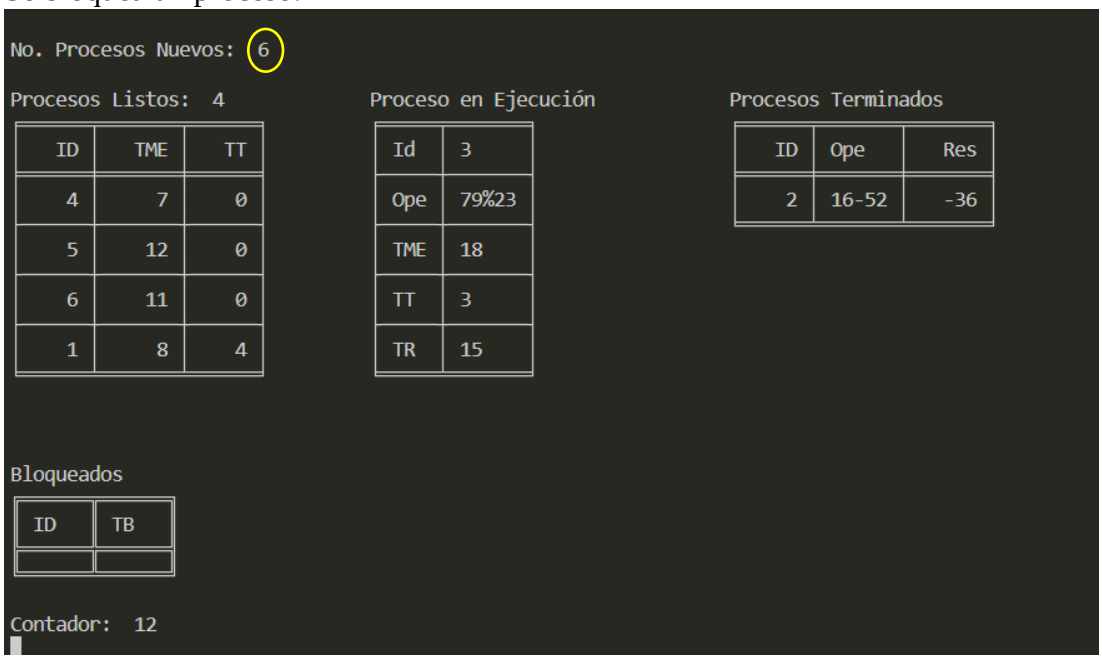

Ilustración 7. Bloqueo de proceso.

4. Se pausa la tabla para mostrar los datos que tiene hasta el momento cada proceso:

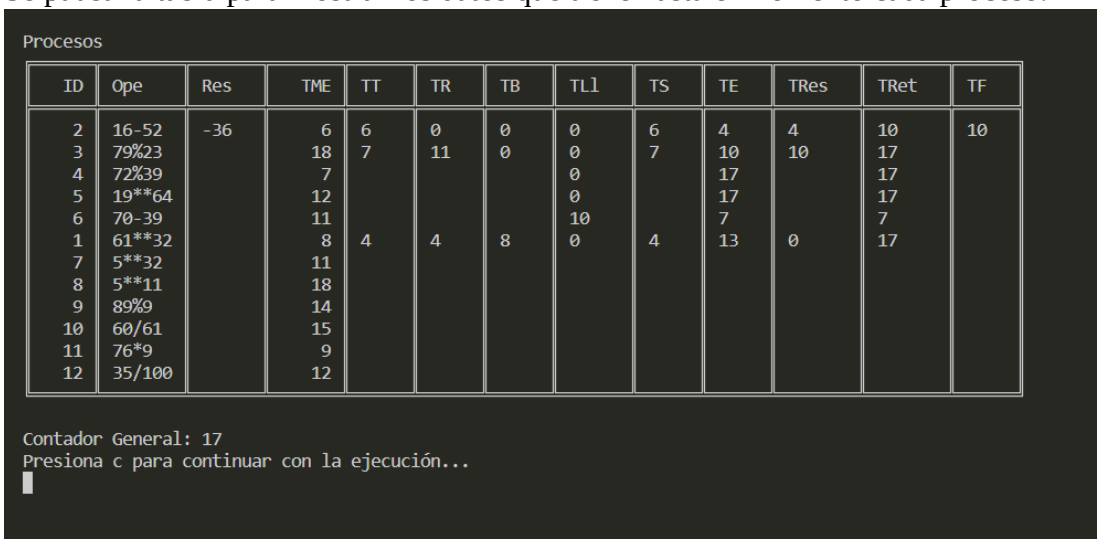

Ilustración 8. Datos temporales 2.

5. Una vez continua el programa, se agregan 3 procesos más:

No. Procesos Nuevos: 9

Procesos Listos: 4

ID	TME	TT
4	7	0
5	12	0
6	11	0
1	8	4

Proceso en Ejecución

Id	3
Ope	79%23
TME	18
TT	12
TR	6

Procesos Terminados

ID	Ope	Res
2	16-52	-36

Bloqueados

ID	TB

Contador: 21

Ilustración 9. Procesos nuevos.

6. Mostramos la tabla:

Procesos

ID	Ope	Res	TME	TT	TR	TB	TLl	TS	TE	TRes	TRet	TF
2	16-52	-36	6	6	0	0	0	6	4	4	10	10
3	79%23	10	18	18	0	0	0	18	10	10	28	28
4	72%39		7	1	6	0	0	1	28	28	29	
5	19**64		12				0		29		29	
6	70-39		11				10		19		19	
1	61**32		8	4	4	8	0	4	25	0	29	
7	5**32		11				28		1		1	
8	5**11		18									
9	89%9		14									
10	60/61		15									
11	76*9		9									
12	35/100		12									
13	7**14		17									
14	56+10		15									
15	80%9		6									

Contador General: 29
Presiona c para continuar con la ejecución...

Ilustración 10. Datos temporales 3.

7. Mandamos un proceso por error y dos procesos a bloqueado:

No. Procesos Nuevos: 7

Procesos Listos: 2

ID	TME	TT
7	11	0
8	18	0

Proceso en Ejecución

Id	1
Ope	61**32
TME	8
TT	6
TR	2

Procesos Terminados

ID	Ope	Res
2	16-52	-36
3	79%23	10
4	72%39	Error

Bloqueados

ID	TB
5	3
6	2

Contador: 34

Ilustración 11. Interrupciones de procesos.

8. Mostramos tabla:

Procesos

ID	Ope	Res	TME	TT	TR	TB	TLl	TS	TE	TRes	TRet	TF
2	16-52	-36	6	6	0	0	0	6	4	4	10	10
3	79%23	10	18	18	0	0	0	18	10	10	28	28
4	72%39	Error	7	2	5	0	0	2	28	28	30	30
1	61**32	1.35e+57	8	8	0	8	0	8	29	0	37	37
7	5**32		11	1	10	0	28	1	9	9	10	
5	19**64		12	2	10	5	0	2	36	30	38	
6	70-39		11	1	10	4	10	1	27	22	28	
8	5**11		18				30		8		8	
9	89%9		14				37		1		1	
10	60/61		15									
11	76*9		9									
12	35/100		12									
13	7**14		17									
14	56+10		15									
15	80%9		6									

Contador General: 38
Presiona c para continuar con la ejecución...

Ilustración 12. Datos temporales 4.

9. Dejamos que el termine todos los procesos el programa y esta es la tabla final:

Procesos

ID	Ope	Res	TME	TT	TR	TB	TLl	TS	TE	TRes	TRet	TF
2	16-52	-36	6	6	0	0	0	6	4	4	10	10
3	79%23	10	18	18	0	0	0	18	10	10	28	28
4	72%39	Error	7	2	5	0	0	2	28	28	30	30
1	61**32	1.35e+57	8	8	0	8	0	8	29	0	37	37
7	5**32	Error	11	3	8	0	28	3	9	9	12	40
10	60/61	Error	15	5	10	0	40	5	6	6	11	51
6	70-39	Error	11	3	8	8	10	3	40	22	43	53
8	5**11	4.88e+07	18	18	0	8	30	18	22	10	40	70
9	89%9	8	14	14	0	8	37	14	32	4	46	83
12	35/100	Error	12	2	10	0	53	2	33	33	35	88
5	19**64	6.92e+81	12	12	0	16	0	12	82	30	94	94
13	7**14	6.78e+11	17	17	0	0	70	17	24	24	41	111
15	80%9	8	6	6	0	0	88	6	32	32	38	126
11	76*9	684	9	9	0	8	51	9	72	32	81	132
16	96-4	92	9	9	0	0	127	9	5	5	14	141
14	56+10	66	15	15	0	8	83	15	49	28	64	147

Contador General: 147

Ilustración 13. Datos finales.

Si sumamos el tiempo de llegada, servicio y espera del último proceso, podremos corroborar que los tiempos son correctos, dado que el resultado es igual al contador general.

Errores y Soluciones.

Perdida de datos originales al presionar la tecla ‘B’: Al incorporar la funcionalidad para cuando se presione la tecla ‘B’, se generaba el error de Ilustración 14.

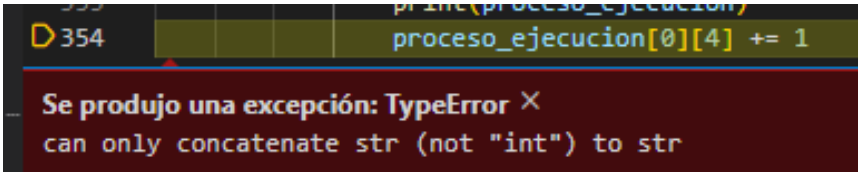

Ilustración 14. Error 1

Esto se debía, ya que, nuestra forma de incorporar esta tecla, fue generando una lista general de la impresión donde se copiarán los datos, de todas nuestras listas donde se contenían los 5 estados, es decir: nuevo, listo, ejecución, bloqueado y terminado; y conforme se copiaban los procesos de cada estado se modifican en espacios en blanco los datos que aún no podían/deberían mostrar en la pantalla final de BCP de cada proceso, o en su caso como era el tiempo de respuesta, retorno y servicio que dabían acomplarse al contador global actual. Sin embargo, detectamso que el error se estaba generando que a pesar de generar ‘copias’ de las lsitas originales estas también eran modificadas, por de cierta forma estar ‘indexadas’ o mejor dicho ’anidadas’ dentro de la lógica, del programa, así como se observa en la Ilustración 15.

```
219  def impresion_final():
220      lista_final=[]
221      copia_terminados = list(procesos_terminados)
222      copia_ejecucion = list(proceso_ejecucion)
223      copia_actuales = list(procesos_actuales)
224      copia_bloqueados = list(procesos_bloqueados)
225      copia_pendiente = list(procesos_pendientes)
226
227      if copia_terminados:
228          for x in copia_terminados:
229              lista_final.append(x)
230
231      if copia_ejecucion:
232          for x in copia_ejecucion:
233              x[2] = ''
234              x[12] = ''
235              x[11] = contadorGeneral - x[7]
236              x[9] = x[11]-x[8]
237
238              lista_final.append(x)
239
240      if copia_bloqueados:
241          for x in copia_bloqueados:
242              x[2] = ''
243              x[12] = ''
244              x[11] = contadorGeneral - x[7]
245              x[9] = x[11]-x[8]
246
247              lista_final.append(x)
```

Ilustración 15. Causa del Error 1

Así que, tras percatarnos de dicho error, la solución más puntutal que encontramos tras investigar en diversas fuentes, sin necesidad de hacer grandes modificaciones en el código, fue con la librería ‘copy’ y adaptando la copia de cada lista de la siguiente forma.

```
221  def impresion_final():
222      lista_final=[]
223      copia_terminados = copy.deepcopy(procesos_terminados)
224      copia_ejecucion = copy.deepcopy(proceso_ejecucion)
225      copia_actuales = copy.deepcopy(procesos_actuales)
226      copia_bloqueados = copy.deepcopy(procesos_bloqueados)
227      copia_pendiente = copy.deepcopy(procesos_pendientes)
228
229      if copia_terminados:
230          for x in copia_terminados:
231              lista_final.append(x)
232
233      if copia_ejecucion:
234          for x in copia_ejecucion:
235              x[2] = ''
236              x[12] = ''
237              x[11] = contadorGeneral - x[7]
238              x[9] = x[11]-x[8]
239              x[11] = ''
240
241              lista_final.append(x)
242
243      if copia_bloqueados:
244          for x in copia_bloqueados:
245              x[2] = ''
246              x[12] = ''
247              x[11] = contadorGeneral - x[7]
248              x[9] = x[11]-x[8]
249              x[11] = ''
250
251              lista_final.append(x)
252
```

Ilustración 16.Solución Error 1

Muestra de tiempo incorrecto: Una vez presentando el programa a la maestra, nos señaló que estaba incorrectó que se mostrará el tiempo de respuesta en la tabla BCP para aquellos procesos que teniían afirmativa la bandera de que ya habían entrado mínimo una vez a ejecución. Mostrandosé nuestros tiempos inicialmente como se observa en la Ilustración 17.

Procesos

ID	Ope	Res	TME	TT	TR	TB	TL1	TS	TE	TRes	TRet	TF
2	67-62		15	1	14	0	0	1	4	4	5	
1	69/43		13	4	9	0	0	4	1	0	5	
3	70%71		10				0		5		5	
4	9/42		10				0		5		5	
5	30*8		6				0		5		5	
6	98*25		10									
7	96*36		7									

Ilustración 17. Error 2

Siendo así, que para poder identificar dicho error, nos bastó buscar la parte del código correspondiente a las variables ‘vacías’ para la impresión final de BCP de cada proceso, así como se observa en la Ilustración 18.

```
231     if copia_ejecucion:
232         for x in copia_ejecucion:
233             x[2] = ''
234             x[12] = ''
235             x[11] = contadorGeneral - x[7]
236             x[9] = x[11]-x[8]
237 
238             lista_final.append(x)
239 
240     if copia_bloqueados:
241         for x in copia_bloqueados:
242             x[2] = ''
243             x[12] = ''
244             x[11] = contadorGeneral - x[7]
245             x[9] = x[11]-x[8]
246 
247             lista_final.append(x)
248 
249     if copia_actuales:
250         for x in copia_actuales:
251             if x[13] == 1:
252                 x[2] = ''
253                 x[12] = ''
254                 x[11] = contadorGeneral - x[7]
255                 x[9] = x[11]-x[8]
```

Ilustración 18. Causa del Error 2

Para lgorar corregir esto, teniendo en cuenta que necesitabamos este tiempo para calculart el tiempo de servicio hasta el momento, se optó por hacer una doble moodifiación dentro de dicho espacio asignado dentro de nuestro proceso, dejando una salida vacía, como la maestra lo indicó. Así como se observa en la Ilustración 19.

```
233     if copia_ejecucion:
234         for x in copia_ejecucion:
235             x[2] = ''
236             x[12] = ''
237             x[11] = contadorGeneral - x[7]
238             x[9] = x[11]-x[8]
239             x[11] = ''
240 
241             lista_final.append(x)
242 
243     if copia_bloqueados:
244         for x in copia_bloqueados:
245             x[2] = ''
246             x[12] = ''
247             x[11] = contadorGeneral - x[7]
248             x[9] = x[11]-x[8]
249             x[11] = ''
250 
251             lista_final.append(x)
252 
253     if copia_actuales:
254         for x in copia_actuales:
255             if x[13] == 1:
256                 x[2] = ''
257                 x[12] = ''
258                 x[11] = contadorGeneral - x[7]
259                 x[9] = x[11]-x[8]
260                 x[11] = ''
```

Ilustración 19. Solución Error 2

Conclusiones

Maria Dolores.

Para esta entrega me atrevería a decir que no fue nada complejo realizar las modificaciones pertinentes, debido a que en el programa anterior modificamos la estructura en como accedíamos a los datos dentro de la lista, nos facilitó el trabajo al momento de implementar las nuevas funciones con teclas.

Sin embargo, el único error que presentamos si nos resultó bastante peculiar, debido a que por alguna razón al crear las copias de las listas se modificaban los datos tanto de la copia como de las listas originales al momento de mostrar la tabla temporal con los tiempos de cada proceso. Pese a este inconveniente pudimos arreglarlo llevándonos un conocimiento extra sobre una nueva librería que es de gran utilidad para la realización de copias en listas.

Finalmente, concluyo que es interesante ver cómo va evolucionando nuestro programa con cada requerimiento extra, realizando un ejecutable más optimo y eficaz, logrando los objetivos previstos en la actividad y de la mano de mi compañero que es una pieza clave para complementar mejor el algoritmo presentado.

Aldo Fernando.

. Realmente, se me hizo este cuarto programa el más sencillo de todos, sin embargo, si me pareció algo increíble el error 1 que tuvimos, ya que, pensándolo lógicamente no tiene sentido alguno, sin embargo, puedo decir que afortunadamente este programa me enseñó a como realizar correctamente una copia totalmente independiente de una lista.

Por otro lado, nuevamente se logró hacer buen equipo con mi compañera, además de que ambos claramente teníamos la misma idea de por donde llevar la asignación de las dos nuevas teclas con sus respectivas funcionalidades, haciendo los cambios pertinentes iniciales de forma muy rápida.

Así que, personalmente puedo decir que me quedo muy satisfecho con el resultado obtenido, y no solamente con el programa como tal, si no, con todos los aprendizajes obtenidos.