Instituto Tecnológico de Costa Rica

Asignación 1b

Aseguramiento de la Calidad del Software

Número de Grupo: 2

Elaborado por:

Jeremy Madrigal Portilla - 2019258245

Randall Gabriel Zumbado Huertas - 2019082664

Randy Conejo Juárez - 2019066448

Profesor:

Ignacio Trejos Zelaya

San José, octubre de 2021

Análisis somero de las diferencias entre los componentes

Para las funciones realizadas en ambas asignaciones podemos observar que tenemos diferencia entre los componentes, podemos notar que en la asignación 1b las funciones realizadas a la hora de elaborarlas utilizamos diferentes funciones de la asignación 1a ya que esta misma facilitaba la elaboración, por lo que entre sus componentes podemos anotar que hay diferencia en cuanto a implementación de las funciones y su longitud en cuanto a código ya que se utilizó del ejercicio pasado. También se puede observar que las validaciones al estar en los componentes de la asignación 1a favoreció a esta asignación ya que al estar ya validadas y reutilizar el código anterior solo se implemento componentes mínimos de validación en las funciones creadas en esta asignación. Igualmente se realizó un cambio significativo entre los diccionarios de la asignación 1a y 1b, para la 1b se agregó el nombre de cada mes en una string para poder almacenarlos, ya que en el requerimiento R6 es necesario tener los nombres para poder manipularlos a la hora de imprimir los meses en pantalla y se deja previsto para futuras implementaciones.

Historial de modificaciones realizadas al código previo

Realizamos un cambio en el diccionario de los meses

Primera implementación de diccionario en python

```
meses = {
0  : 0,  #Para validar último mes
1  : 31, #Enero
2  : 28, #Febrero no bisiesto
3  : 31, #Marzo
4  : 30, #Abril
5  : 31, #Mayo
6  : 30, #Junio
```

```
7  : 31, #Julio
8  : 31, #Agosto
9  : 30, #Septiembre
10 : 31, #Octubre
11 : 30, #Noviembre
12 : 31 #Diciembre
}
```

Modificación de ese calendario

```
meses = {
0  : 0,  #Para validar último mes
1  : [31, "Enero"], #Enero
2  : [28, "Febrero"], #Febrero no bisiesto
3  : [31, "Marzo"], #Marzo
4  : [30, "Abril"], #Abril
5  : [31, "Mayo"], #Mayo
6  : [30, "Junio"], #Junio
7  : [31, "Julio"], #Julio
8  : [31, "Agosto"], #Agosto
9  : [30, "Septiembre"], #Septiembre
10 : [31, "Octubre"], #Octubre
11 : [30, "Noviembre"], #Noviembre
12 : [31, "Diciembre"] #Diciembre
}
```

Primera implementación de diccionario en JavaScript

```
let meses = new Map()
meses.set(0,0)
meses.set(1,31)
meses.set(2,28)
meses.set(3,31)
meses.set(4,30)
meses.set(5,31)
meses.set(6,30)
```

```
meses.set(7,31)
meses.set(8,31)
meses.set(9,30)
meses.set(10,31)
meses.set(11,30)
meses.set(12,31)
```

Modificación de ese calendario

```
let meses = new Map()
meses.set(0,0)
meses.set(1,[31, "Enero"])
meses.set(2,[28, "Febrero"])
meses.set(3,[31, "Marzo"])
meses.set(4,[30, "Abril"])
meses.set(5,[31, "Mayo"])
meses.set(6,[30, "Junio"])
meses.set(7,[31, "Julio"])
meses.set(8,[31, "Agosto"])
meses.set(9,[30, "Septiembre"])
meses.set(10,[31, "Octubre"])
meses.set(11,[30, "Noviembre"])
meses.set(12,[31, "Diciembre"])
```

Horas de esfuerzo

Actividad realizada	**Horas**
Análisis del dominio	0.25
Análisis de requerimientos	0.75
Investigaciones específicas	0.5
Diseño de estructuras de datos, algoritmos, bases de datos, etc	0.5
Programación	4
Documentación interna	0.75
Planificación y diseño de pruebas	0.5
Ejecución y análisis de pruebas	0.5
Integración y prueba de módulos o componentes	0.5
Correcciones a módulos o componentes	0.75
Elaboración de manuales de usuario, ayudas en línea, etc	0
Elaboración de documentación del proyecto	0.5
Otras actividades según se requiera	0.25

Requerimientos funcionales

- R6 (imprimir_4x3): Dado un año perteneciente al rango permitido, desplegar en consola el calendario de ese año en formato de 4 secuencias (‘filas’) de 3 meses cada una. El resultado debe lucir semejante al que se muestra al final de este enunciado.
- R7 (dia_semana): Dada una fecha válida, determinar el día de la semana que le corresponde, con la siguiente codificación: 0 = domingo, 1 = lunes, 2 = martes, 3 = miércoles, 4 = jueves, 5 = viernes, 6 = sábado. El resultado debe ser un número entero, conforme a la codificación indicada.
- R8 (fecha_futura): Dados una fecha válida f y un número entero no-negativo n, determinar la fecha que está n días naturales en el futuro. Si n es 0, entonces fecha_futura(f, 0) = f. El resultado debe ser una fecha válida.
- R9 (dias_entre): Dadas dos fechas válidas, f1 y f2, sin importar si f1 ≤ f2 o f2 ≤ f1, determinar el número de días naturales entre las dos fechas. Si f1 = f2, entonces días_entre(f1, f2) = 0. El resultado debe ser un número entero no negativo.

Resumen de hallazgos del calendario

Qué es el calendario juliano.

El calendario juliano según EcuRed se denomina así por quién se instauró, Julio César durante su gobierno para poner fin al caos con los diferentes tipos de calendarios y las incongruencias que habían entre ellos se propuso instaurar un único calendario que todos siguieran.

Junto con el inicio de Roma se utilizó este calendario, según los contribuidores de EcuRed (2018)

> Para que volviese a caer cada estación, con las fiestas y celebraciones correspondientes, en el Tiempo astronómico que le correspondía, se vio obligado a hacer el primer año de 445 días. Fue conocido con el nombre de año de la confusión

Luego se empezaría a regir por las siguientes reglas:

12 meses denominados en nuestra lengua Enero, Febrero, Marzo, Abril, Mayo, Junio, Julio (Por Julio César también), Agosto (Por Augusto), Setiembre, Octubre, Noviembre y Diciembre.

El mes de febrero con 28 días en año no bisiesto y 29 en año bisiesto, según los contribuidores de EcuRed esto debido a que Augusto estaba celoso de que el mes en su nombre tuviera menos días que el de Julio César, le quitó un día a febrero y lo puso en Agosto.

Hay un año bisiesto cada 4 años.

Antes del mes de Agosto se rige porque los meses impares tienen 31 días y los pares 30, después de Agosto que tiene 31 días los impares tendrán 30 y los pares 31.

Qué es el calendario gregoriano.

Es el calendario que utilizan la mayoría de países hoy en día, parafraseando a EcuRed este fue promovido por el Papa Gregorio XIII y fue el calendario encargado de reemplazar al juliano que había funcionado durante un largo periodo, este calendario mitigó un poco el error cometido en el calendario juliano de contar un año bisiesto cada 4 años.

Este calendario se rige por las siguientes reglas:

Existen los mismos meses que en el calendario juliano.

Los meses tendrán la misma cantidad de días que en el juliano y el bisiesto contará con el mismo comportamiento pero los años no serán de la misma manera.

Para los años bisiestos según EcuRed (2019)

> La nueva norma de los años bisiestos se formuló del siguiente modo: la duración básica del año es de 365 días; pero serán bisiestos (es decir tendrán 366 días) aquellos años cuyas dos últimas cifras son divisibles por 4, exceptuando los múltiplos de 100 (1700, 1800, 1900...), de los que se exceptúan a su vez aquellos que también sean divisibles por 400

Esto para corregir el problema con los días sobrantes del calendario juliano.

Cuándo fue instituido el calendario gregoriano.

Según Jouette, André (2008), en 1582, el papa Gregorio XIII instituyó el calendario gregoriano, esto basado en los estudios de Luis Lilio, Christhoper Clavius y Pedro Chacón, publica el libro "Inter Gravissimas" para derogar el calendario juliano.

Cuáles deficiencias o limitaciones del calendario juliano subsanó el calendario gregoriano.

Según los Colaboradores de Wikipedia (2021), las deficiencias o limitaciones que el calendario gregoriano subsanó al juliano fueron:

- El calendario juliano se atrasa un día con respecto al año trópico cada 128 años, osea equivale a 11 minutos con 14 segundos de exceso por año, en cambio para el gregoriano se atrasa 1 día con respecto al año trópico cada 3324 años por lo cual es más estable este último.
- El calendario juliano considera un año como 365,25 días cuando los griegos ya sabían que duraba unos minutos menos, este mismo perdía 3 días cada 4 siglos en comparación con los equinoccios, por lo cual el gregoriano arreglo este cálculo pasando los días a 365,2425 días.

Determinar la relación entre las fechas en que murieron los escritores Miguel de Cervantes y William Shakespeare.

Según CNN Español (2021), el día 23 de abril de 1616 murieron ambos escritores, día por el cual, en homenaje a ellos dos, se decidió que se celebraría el día del libro; sin embargo no es del todo cierto que ambos escritores murieron ese mismo día del mismo año, esto debido a las costumbres y calendario utilizados en la época por España y Gran Bretaña.

Miguel de Cervantes no murió el 23 de abril, ese día fue el de su entierro, él murió el día anterior, y en aquella época se solía colocar como fecha de defunción la fecha en la que se

enterraban a los difuntos, por lo que Miguel de Cervantes realmente murió el 22 de abril de 1616.
Con William Shakespeare el caso es distinto, pues en el Reino Unido por aquella época, el calendario que se utilizaba seguía siendo el juliano. El Reino Unido siguió con este calendario hasta el año 1752, cuando murieron ambos escritores, los calendarios de España y Gran Bretaña tenían una diferencia de 10 días, cuando murió William Shakespeare era 23 de abril en Gran Bretaña, pero en España y gran parte del occidente era 3 de mayo de 1616.
Por lo que vemos como una diferencia en el calendario de estos países hizo ser el día 23 de abril como la muerte de ambos escritores, cuando no fue así.

Decisiones de diseño tomadas

Para este proyecto se tuvo que cambiar nuestro diccionario base que se tenía de los meses del año junto con la cantidad de días, ya que necesitábamos el nombre de los meses para el requerimiento R6 de este proyecto, por lo que para almacenar este mismo la opción más sencilla que se nos ocurrió fue agregar el nombre del mes en el diccionario para así identificarlo. También se decidió tomar el código anterior para poder usarlo en estos requerimientos ya que nos permite reutilizar funciones que nos ayudan a minimizar código nuevo para tener una solución más sencilla en cuanto a longitud se refiere. Igualmente se implementó otras funciones extra como ordena_fechas, que servía para poner primero la fecha menor y luego la mayor, para dividir el código que algunos de estos requerimientos exigen, esto para tener una complejidad menor a la hora de la lectura del código fuente junto con los requerimientos que se desarrollaron en esta asignación.

Evidencias de las pruebas realizadas

Se realizaron pruebas dentro de los archivos fuente de código python y javascript, al final de cada código escribimos en prints los parámetros que enviamos a la función, lo que se espera que retorne la función, y lo que retornó la función, estas pruebas son las siguientes imágenes.

Pruebas realizadas en python

```
pruebas de dia_semana

Parámetros: + (2021, 12, 31); Esperado: 5 ; Obtenido: 5
Parámetros: + (2020, 12, 31); Esperado: 4 ; Obtenido: 4
Parámetros: + (2020, 3, 1); Esperado: 0 ; Obtenido: 0
Parámetros: + (2021, 3, 1); Esperado: 1 ; Obtenido: 1
Parámetros: + (2021, 1, 31); Esperado: 0 ; Obtenido: 0

pruebas de fecha_futura

Parámetros: (2021, 2, 28), 5 ; Esperado: (2021, 3, 5) ; Obtenido: (2021, 3, 5)
Parámetros: (2021, 12, 31), 7 ; Esperado: (2022, 1, 7) ; Obtenido: (2022, 1, 7)
Error: algún elemento de la tupla no es un número
Parámetros: ('2020', 1, 4), 5 ; Esperado: Fecha no válida ('2020', 1, 4) ; Obtenido: ('2020', 1, 4)
Número de días debe ser un entero
Parámetros: (2021, 2, 28), '7' ; Esperado: Número de días no válidos (2021, 2, 28) ; Obtenido: (2021, 2, 28)
Número de días debe ser positivo
Parámetros: (2021, 2, 28), -9 ; Esperado: Número de días negativos (2021, 2, 28) ; Obtenido: (2021, 2, 28)
Parámetros: (2021, 12, 14), 6  ; Esperado: (2021, 12, 20) ; Obtenido: (2021, 12, 20)
Parámetros: (2015, 1, 1), 2556  ; Esperado: (2021, 12, 31) ; Obtenido: (2021, 12, 31)

pruebas de ordena_fechas

Parámetros: (2021, 12, 31), (2021, 1, 31) ; Esperado: [(2021, 1, 31), (2021, 12, 31)] ; Obtenido: [(2021, 1, 31), (2021, 12, 31)]
Parámetros: (2021, 1, 31), (2021, 12, 31) ; Esperado: [(2021, 1, 31), (2021, 12, 31)] ; Obtenido: [(2021, 1, 31), (2021, 12, 31)]
Parámetros: (2021, 2, 28), (2021, 2, 29) ; Esperado: [(2021, 2, 28), (2021, 2, 29)] ; Obtenido: [(2021, 2, 28), (2021, 2, 29)]
Parámetros: (2021, 2, 29), (2021, 2, 28) ; Esperado: [(2021, 2, 28), (2021, 2, 29)] ; Obtenido: [(2021, 2, 28), (2021, 2, 29)]
Parámetros: (2020, 2, 28), (2021, 2, 28) ; Esperado: [(2020, 2, 28), (2021, 2, 28)] ; Obtenido: [(2020, 2, 28), (2021, 2, 28)]
Parámetros: (2021, 2, 28), (2020, 2, 28) ; Esperado: [(2020, 2, 28), (2021, 2, 28)] ; Obtenido: [(2020, 2, 28), (2021, 2, 28)]

pruebas de dias_entre

Parámetros: (2021, 12, 31), (2021, 1, 31) ; Esperado: 334 ; Obtenido: 334
Parámetros: (2021, 2, 28), (2016, 10, 15) ; Esperado: 1597 ; Obtenido: 1597
Parámetros: (2016, 10, 9), (2016, 10, 15) ; Esperado: 6 ; Obtenido: 6
Parámetros: (2016, 10, 15), (2016, 10, 9) ; Esperado: 6 ; Obtenido: 6
Parámetros: (2016, 10, 15), (2016, 10, 15) ; Esperado: 0 ; Obtenido: 0
Error: algún elemento de la tupla no es un número
Parámetros: ('2020', 1, 4), (2016, 10, 15) ; Esperado: Fecha inválida -1 ; Obtenido: -1
Error: algún elemento de la tupla no es un número
Parámetros: (2016, 10, 15), ('2020', 1, 4) ; Esperado: Fecha inválida -1 ; Obtenido: -1
Parámetros: (2015, 1, 1), (2021, 12, 31) ; Esperado: 2556 ; Obtenido: 2556
```

Pruebas realizadas en javascript

```
pruebas de dia_semana

Esperado: 5 ; Obtenido: 5
Esperado: 4 ; Obtenido: 4
Esperado: 0 ; Obtenido: 0
Esperado: 1 ; Obtenido: 1
Esperado: 0 ; Obtenido: 0

pruebas de fecha_futura

Parámetros: 2021,2,28, 5 ; Esperado: (2021, 3, 5) ; Obtenido: 2021,3,5
Parámetros: 2021,12,31, 7 ; Esperado: (2022, 1, 7) ; Obtenido: 2022,1,7
Error: algún elemento de la tupla no es un número
Parámetros: 2020,1,4, 5 ; Esperado: Fecha no válida ('2020', 1, 4) ; Obtenido: 2020,1,4
Número de días debe ser un entero
Parámetros: 2021,2,28, '7' ; Esperado: Número de días no válidos (2021, 2, 28) ; Obtenido: 2021,2,28
Número de días debe ser positivo
Parámetros: 2021,2,28, -9 ; Esperado: Número de días negativos (2021, 2, 28) ; Obtenido: 2021,2,28
Parámetros: (2021, 12, 14), 6  ; Esperado: (2021, 12, 20) ; Obtenido: 2021,12,20
Parámetros: (2015, 1, 1), 2556  ; Esperado: (2021, 12, 31) ; Obtenido: 2021,12,31

pruebas de ordena_fechas

Parámetros: 2021,12,31, 2021,1,31 ; Esperado: [(2021, 1, 31), (2021, 12, 31)] ; Obtenido: 2021,1,31,2021,12,31
Parámetros: 2021,1,31, 2021,12,31 ; Esperado: [(2021, 1, 31), (2021, 12, 31)] ; Obtenido: 2021,1,31,2021,12,31
Parámetros: 2021,2,28, 2021,2,29 ; Esperado: [(2021, 2, 28), (2021, 2, 29)] ; Obtenido: 2021,2,28,2021,2,29
Parámetros: 2021,2,29, 2021,2,28 ; Esperado: [(2021, 2, 28), (2021, 2, 29)] ; Obtenido: 2021,2,28,2021,2,29
Parámetros: 2020,2,28, 2021,2,28 ; Esperado: [(2020, 2, 28), (2021, 2, 28)] ; Obtenido: 2020,2,28,2021,2,28
Parámetros: 2021,2,28, 2020,2,28 ; Esperado: [(2020, 2, 28), (2021, 2, 28)] ; Obtenido: 2020,2,28,2021,2,28

pruebas de dias_entre

Parámetros: 2021,12,31, 2021,1,31 ; Esperado: 334 ; Obtenido: 334
Parámetros: 2021,2,28, 2016,10,15 ; Esperado: 1597 ; Obtenido: 1597
Parámetros: 2016,10,9, 2016,10,15 ; Esperado: 6 ; Obtenido: 6
Parámetros: 2016,10,15, 2016,10,9 ; Esperado: 6 ; Obtenido: 6
Parámetros: 2016,10,15, 2016,10,15 ; Esperado: 0 ; Obtenido: 0
Error: algún elemento de la tupla no es un número
Parámetros: 2020,1,4, 2016,10,15 ; Esperado: Fecha inválida -1 ; Obtenido: -1
Error: algún elemento de la tupla no es un número
Parámetros: 2016,10,15, 2020,1,4 ; Esperado: Fecha inválida -1 ; Obtenido: -1
Parámetros: (2015, 1, 1), (2021, 12, 31) ; Esperado: 2556 ; Obtenido: 2556
```

Análisis de resultados obtenidos

Durante la elaboración de este proyecto se amplió el conocimiento por parte de los integrantes en JavaScript, pudimos también profundizar en los conceptos asociados al calendario gregoriano, donde utilizando código previamente realizado pudimos implementar soluciones de forma más sencilla.

La sección más importante durante este proyecto es la realización de pruebas, en el anterior proyecto se ejecutaron algunas de estas, pero antes de implementar las nuevas vimos en clases los fundamentos de las pruebas de caja negra y de pruebas a código como tal, esto sirvió para que entre todos los integrantes realizarán pruebas más eficientes donde se prueba cada uno de los casos posibles para los bloques de código, además de esto al tener todos los integrantes conocimientos de estos fundamentos pudimos realizar revisión de pares no solamente del código escrito sino también de las pruebas realizadas, esto para mejorar la robustez del código.