Universidad Carlos III

Curso Desarrollo de Software 2022-23

Ejercicio Guiado 3

DESARROLLO DE SOFTWARE

EJERCICIO 3

Fecha: **28/03/2023** - ENTREGA: **31/03/2023**

GRUPO: **80** EQUIPO: **05**

Alumnos: **Luis Gómez-Manzanilla Nieto / Ana María Ortega Mateo**

TABLA DE CONTENIDO

RF1

El primer método creado es *register_order()* el cuál, pasándole como parámetros el id de un producto, el tipo de pedido, la dirección de envío, un teléfono de contacto y el código zip. La función, valida los datos de entrada, crea una instancia de la clase *OrderRequest()* y lo almacena en un fichero llamado *store_request*, por último devuelve un hash de 32 bits que hace referencia a la instancia creada.

CLASES DE EQUIVALENCIA Y VALORES LÍMITE

En la primera función hay distintos parámetros que componen un pedido, para ello hemos tenido que crear distintos tests que validan estos componentes. Teniendo en cuenta sus clases de equivalencia y valores límites en la realización de estos tests. Hay dos tests globales correctos que comprueba que estos parámetros son los correctos y adecuados al formato de cómo tienen que ser, por ello los hemos clasificado como clase de equivalencia ya que verifican que todos los parámetros sean válidos. Se ha hecho uno con PREMIUM y otro con REGULAR para poder validarlo con los dos tipos de pedidos posibles.

● PRODUCT ID

Para un product ID correcto, debe cumplir con que su longitud sea igual a 13 ya que es un ean13 el código del pedido. Los valores límites que hemos definido han sido relacionados con su longitud y por ello hemos creado dos tests incorrectos que comprueban la longitud límite del productID:

-Longitud más corta: Se encuentra por debajo del límite inferior de longitud, por ello comprobamos un ean13 no válido de longitud 12.

-Lóngitud más larga: Se encuentra por encima del límite superior, lo comprobamos con un ean13 no válido de longitud 14.

Clase de equivalencias de este parámetro los cuales no serían test válidos son dos casos:

-Suma del ean13: Que el product ID no sea válido porque no sigue el formato de ean13 , la suma de las posiciones pares y la suma de las posiciones impares por 3, sumandolas y redondeando a decenas el valor dado y restando este resultado con la suma anterior tiene que dar el último dígito. Si no da el número correspondiente el ean13 no es válido por lo que se comprueba esta prueba con ean13 que no cumpla esta condición.

-Carácter no válido: Un test no válido sería un productID con un carácter ya que el productID está compuesto por todo números.

● ORDER_TYPE

Para hacer los test de este parámetro al solo haber dos casos posibles correctos, que son si el order_type es PREMIUM o es REGULAR, que ya se comprueba de por sí en los test válidos de todos los productos por lo que solo hemos hecho un test incorrecto de clase de equivalencia donde no es de ninguno de estos dos tipos.

- ### ADDRESS

La dirección está definida por una longitud de entre 20 y 100 caracteres que deberá de tener más de un espacio por lo que hemos valorado como test incorrectos sus valores límites de longitud:

-Longitud más corta: Una dirección de longitud 19 que está justo por debajo del límite inferior.

-Longitud más larga: Una dirección de longitud 101 que está justo por encima del límite superior.

Para probar una dirección que no cumple el número de espacios hemos creado un test de clase de equivalencia incorrecto con el caso de una dirección sin espacios.

- ### PHONE

El número de teléfono(parámetro phone), debe de ser de longitud 9 de dígitos, por lo que hemos creado dos test incorrectos valorando los límites de su longitud:

-Longitud más corta: Un phone de longitud de dígitos 8, justo por debajo del límite inferior.

-Longitud más larga: Un phone de longitud de dígitos 10, justo por encima del límite superior.

Un teléfono también no válido sería un número no formado por 9 dígitos sino que no cumpla dicho formato, para ello hemos creado un test de clase de equivalencia en el que el teléfono está compuesto por un carácter y ocho dígitos comprobando que el tipo de teléfono no es un parámetro de tipo válido.

- ### ZIP_CODE

Para probar los valores límites del parámetro zip code, al tener que ser un código postal perteneciente en España y en España los códigos son de longitud cuatro o cinco dígitos. Por ello hemos probado un test con el límite por debajo del límite inferior y otro test por encima del límite superior.

-Longitud más corta: Longitud(3), por debajo de cuatro.

-Longitud más larga: Longitud(6), por encima de cinco.

Para poder comprobar que cumpla con el tipo de zip_code y que sea de España hemos creado dos tests no válidos de clases de equivalencia en el está compuesto por un carácter y que el código no pertenezca a España siendo los primeros dos dígitos mayores que 52.

RF2

La segunda función es *send_product()* la cuál, tiene como único parámetro de entrada la ruta de un fichero json que contiene un diccionario con “OrderID” que contiene el código hash que es la salida del apartado anterior y “ContactEmail” que es un correo. En este ejercicio comprobamos que la entrada esté en el formato correcto y tenga las claves correctas. Para comprobar la entrada lo hacemos mediante duplicación, eliminación de los nodos no terminales y modificación de los nodos terminales. Además, comprobamos que el *OrderID* se

encuentre dentro del fichero *store_request*. Una vez habiendo validado lo anterior, se crea una instancia de la clase *OrderShipping()* y la guardamos en el almacén *store_shipping* y se crea el *tracking_code*. La función devuelve el tracking code.

- ## GRAMÁTICA

Fichero ::= Inicio Dato Fin

Inicio ::= {

Fin ::= }

Dato ::= Campo1 Separador Campo2

Separador ::= ,

Campo1 ::= Etiqueta1 Igualdad Dato1

Campo2 ::= Etiqueta2 Igualdad Dato2

Igualdad ::= :

Etiqueta1 ::= Comillas Valor1 Comillas

Etiqueta2 ::= Comillas Valor2 Comillas

Comillas ::= “

Valor1 ::= OrderId

Valor2 ::= ContactEmail

Dato1 ::= Comillas Valor_dato1 Comillas

Valor_dato1 ::= [0-9a-f]{32}

Dato2 ::= Comillas Valor_dato2 Comillas

Valor_dato2 ::= Correo Arroba Dominio Punto Extensión

Arroba ::= @

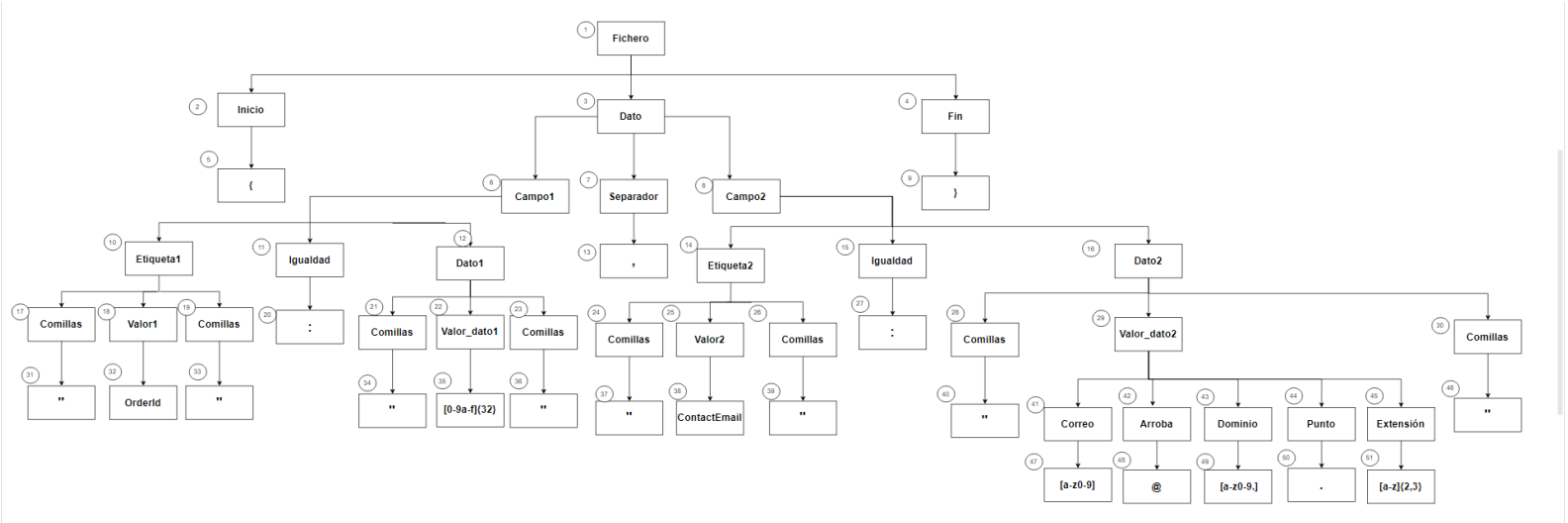
Punto ::= .

Correo ::= [a-z0-9]

Dominio ::= [a-z0-9.]

Extensión ::= [a-z]{2,3}

- ## ÁRBOL



- ## CÓDIGO DE APOYO

Para la creación de los ficheros json para probar los diferentes test, hemos utilizado este código para que se creen automáticamente los ficheros y solo haya que cambiar manualmente el contenido ya sea duplicando, modificando o eliminando el nodo al que haga referencia el test.

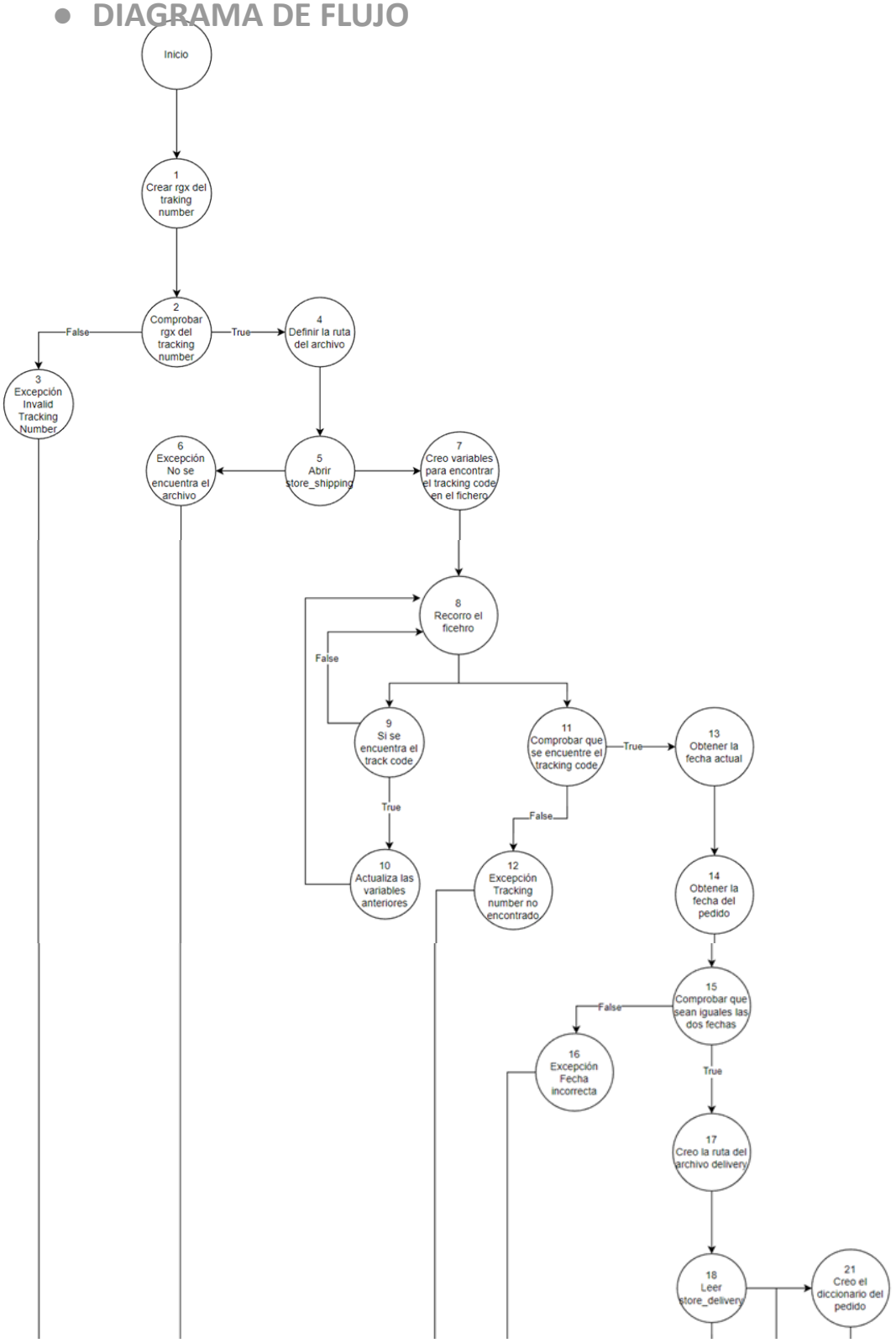
El código es el siguiente:

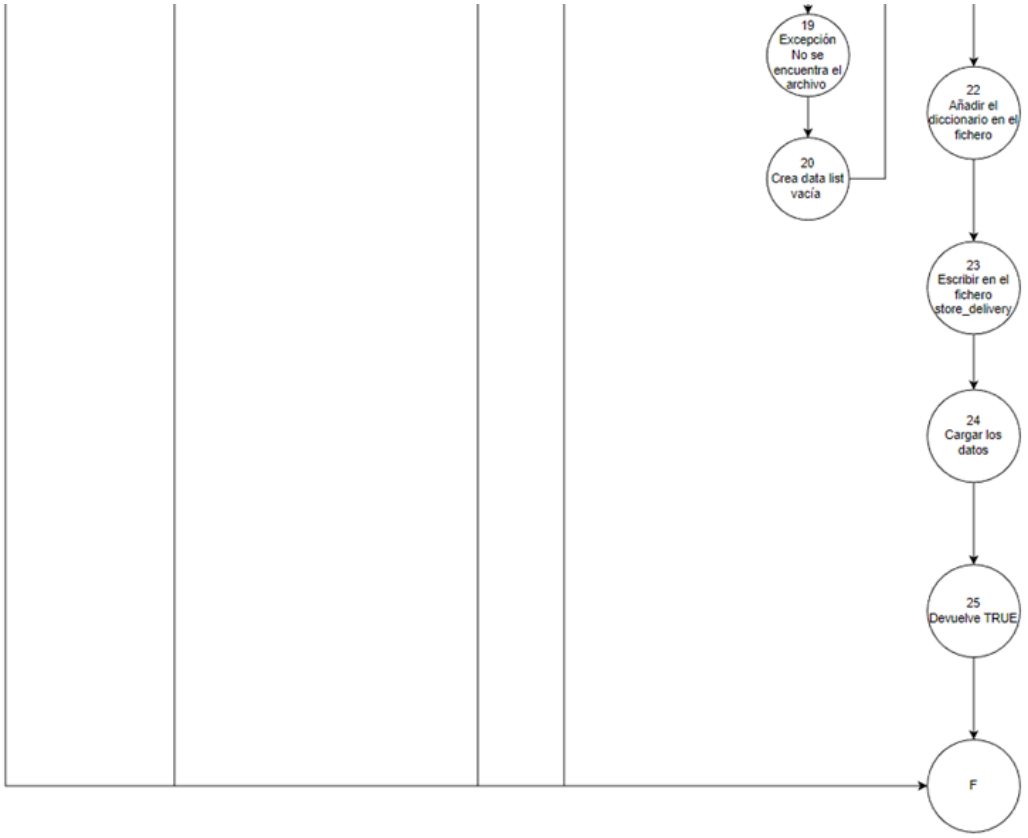
```
p1 > p.py > ...
import json
import os
data = []
add = ""
l1 = [2,3,4,6,7,8,10,11,12,14,15,16,17,18,19,21,22,23,24,25,26,28,29,30,41,42,43,44,45]
l2 = [5,9,13,20,27,31,32,33,34,35,36,37,38,39,40,46,47,48,49,50,51]
for i in l1:
    for j in [0,1]:
        if j==0:
            add = "delete"
        else:
            add = "duplicate"
        file = "C:/Users/189lu/Desktop/Universidad/VSC/SistemasE1/p1/test" + str(i) +"_"+ add +".json"
        test = "C:/Users/189lu/Desktop/Universidad/VSC/SistemasE1/p1/testtt.json"
        if os.path.isfile(file):
            os.remove(file)
        with open(test, "r", encoding= "utf8") as file1:
            data = json.load(file1)
        with open(file, "w", encoding= "utf8") as file2:
            json.dump(data, file2, indent=2)

for i in l2:
    file = "C:/Users/189lu/Desktop/Universidad/VSC/SistemasE1/p1/test" + str(i) +"_modify"+".json"
    test = "C:/Users/189lu/Desktop/Universidad/VSC/SistemasE1/p1/testtt.json"
    if os.path.isfile(file):
        os.remove(file)
    with open(test, "r", encoding= "utf8") as file1:
        data = json.load(file1)
    with open(file, "w", encoding= "utf8") as file2:
        json.dump(data, file2, indent=2)

```

RF3

La tercera función es deliver_product(tracking_number), su función es comprobar la entrega del pedido. Para ello debe comprobar que la fecha del producto y el tracking number almacenados en los envíos de los productos(store_shipping) son los correctos. El tracking number tiene que estar en hexadecimal y de longitud 64 y debe encontrarse en store_shipping, es decir que el producto existe dentro del almacén y es enviado. Para comprobar la fecha coge la fecha del pedido y comprueba que sea igual a la fecha de hoy(correspondiente con las fechas utilizadas en los test con la función freeze time). Si ambos cumplen con dichas condiciones se crea la entrega del producto en un fichero y escribe el tracking_number y su fecha de entrega correspondientes. Finalmente la función devolverá True si todo es correcto.

- **DIAGRAMA DE FLUJO**





- **CAMINOS**

PATH1: 1 2 3

En este camino el tracking_code que nos pasan es erróneo, por lo que no es válido.

PATH2: 1 2 4 5 6

En este camino el fichero store_shipping no existe, por lo que no es válido.

PATH3: 1 2 4 5 7 8 9 10 8 11 12

En este camino el tracking_code no está en el store_shipping, por lo que no es válido.

PATH4: 1 2 4 5 7 8 9 10 8 11 13 14 15 16

En este camino la fecha en la que se produce el delivery es errónea, por lo que no es válido.

PATH5: 1 2 4 5 7 8 9 10 8 11 13 14 15 17 18 19 20 21 22 23 24 25

Este camino es válido pero no existe el archivo store_delivery por lo que lo crea.

PATH6: 1 2 4 5 7 8 9 10 8 11 13 14 15 17 18 21 22 23 24 25

Este camino es válido y el fichero store_delivery ya existe por lo que agrega el nuevo delivery.

Por otro lado, el bucle for lo comprobamos aparte.