**Informe: Trabajo 2 Segunda parte Bases de datos 2 2021-1**

Santiago Espinosa Arteaga cc 1037671102 Juan Pablo Carvajal cc 1152472428

**Variables globales y mappings**

Se tendrán algunas variables globales, deudaGlobal, la cual iniciará en 0 al crearse el contrato y será modificada de acuerdo se realicen y se paguen compras fiadas, otra intentosDestruccion usada en la función de autodestrucción, la dirección dueno que guardará la dirección de quien instancie el contrato. Además de estos, se tendrán los siguientes mappings productos, el cual toma como clave un string y apunta hacia un objeto producto, compradores el cual toma como clave una dirección y apunta a un objeto comprador, cedulasOcupadas el cual toma como clave un string y apunta a un valor booleano y paises cuya clave es un string y apunta a un uint.

**Estructuras**

La estructura comprador tendrá los parámetros código, nombre, pais como strings y totalComprado y totalDeuda como uints, la estructura producto tendrá nombre y descripción como strings y unidades y precioUnitario como uints.

**Modifiers**

**-esDueno():** Verifica que quien invoque la función tenga la misma dirección a la guardada en la constante dueno, este será usado para restringir el uso de funciones que según el enunciado sólo pueden ser realizadas por el dueño y algunas que a pesar de no especificarse, consideramos no tiene mucho sentido que sean realizadas por personas diferentes al dueño del contrato.

**-deudaCero(adress cliente)**: En el mapping compradores busca el comprador con dirección igual a la enviada al modificador como parámetro y verifica que su deuda actual sea 0.

**-noEsDueno()**: Verifica que quien invoque la función tenga una dirección diferente a la guardada en la constante dueno, su uso tiene un propósito análogo al del modificador esDueno().

**-comprobarInventario(string memory nombreProducto)**: En el mapping productos busca aquel que tenga nombre igual al enviado como parámetro y verifica que sus unidades sean mayores a 0.

**-comprobarExistenciaProducto(string memory nombreProducto):** Verifica que en el mapping el producto con nombre igual al enviado tenga un valor en la variable precioUnitario diferente a 0, por defecto las key no modificadas guardan un valor de 0 en este campo y al realizar el registro del producto se garantiza que este sea mayor que 0, por lo que no puede existir un producto creado con precioUnitario 0.

**-comprobarExistenciaComprador()**: Verifica con bytes().length que no haya un valor en el mapping compradores cuya dirección corresponda con quien llama la función modificada, es usado para que compradores no puedan realizar ciertas acciones sin haberse registrado.

**-comprobarPagoExacto(uint valor)**: Verifica que el valor de msg.value enviado por quien llama la función sea igual al valor del parámetro valor del modifier.

**Función calcularDescuento**

Recibe una dirección cliente y un nombre como parámetros, asigna a la variable precio el valorUnitario del producto cuya clave corresponda al encontrado en el mapping productos según el nombre enviado en caso de que precio sea mayor o igual a 3000000000000000000 o 3 ethers y que el totalComprado del cliente cuya dirección coincida con la enviada sea mayor a 50000000000000000000 o 50 ethers se actualiza la variable precio restándole 3 ethers, finalmente retorna el precio por lo que si no se cumplen ambas condiciones la función retorna el mismo precio enviado como parámetro.

**Reportes**

El cliente puede consultar el precio final que pagaría por un producto con la función calcularDescuento, ingresando su dirección y el producto que quiere comprar, para consultar los datos de un producto el cliente puede hacer uso del mapping productos ingresando el producto deseado, para consultar su deuda actual y el valor total de sus compras puede hacer uso también del mapping compradores ingresando su dirección.

**Reportes del dueño**

Para esta acción se crearon 3 getters que tienen como modificador esDueno, estas funciones son **valorTotalCompras** (Que devuelve el balance actual del contrato), **valorTotalComprasPorPais** (Que recibe como parámetro el nombre del país y devuelve el resultado del mapping de países cuyo nombre sea el indicado), **valorTotalDeudas** (Que devuelve el valor de la variable global deudaGlobal)y **obtenerIntentosDestrucción** (Que devuelve la variable global intentosDestruccion).

**Procesos de la tienda**

**Comprar Productos**

Para la compra de productos se construyó la función **comprar** la cual recibe como parámetro el nombre del producto a comprar y usa los modificadores noEsDueno, comprobarExistenciaComprador, deudaCero, comprobarExistenciaProducto, payable comprobarInventario y comprobarPagoExacto (El cual recibe como argumento la función **calcularDescuento**) Esta función en principio envía los fondos del comprador al contrato mediante la función **receiveDeposit,** luego actualiza el mapping de compradores en la dirección del comprador aumentando su valor total comprado, de igual manera se hace con el mapping de países aumentando el valor del país del comprador, por último se decrementan las unidades del producto.

**Comprar un producto fiado**

Para la compra fiada de productos se construyó la función **comprarFiado** la cual recibe como parámetro el nombre del producto fiado a pagar y usa los modificadores noEsDueno,

comprobarExistenciaComprador, deudaCero, comprobarExistenciaProducto y comprobarInventario. Esta función en principio mediante la función **calcularDescuento** guarda el valor del producto a fiar en una variable llamada precio, después de esto, se actualiza el mapping de compradores en la dirección del cliente, allí se aumenta el valor de su deuda con el valor de precio, de igual manera se hace con la variable deudaGlobal. Por último se decrementa el número de unidades del producto.

**Pagar un producto Fiado**

Para el pago de un producto fiado se construyó la función **pagarFiado** la cual tiene como modificadores noEsDueno, paybale, comprobarExistenciaComprador y comprobarPagoExacto (el cual recibe como parámetro el valor de la deuda del cliente guardado en el mapping de compradores). En principio se transfieren los fondos al contrato mediante la función **receiveDeposit**, luego se actualizan el mapping de compradores restando la deuda y aumentando las compras del comprador, la variable deudaGlobal y el mapping de países aumentando su valor.

**Registro de compradores**

Mediante la función **registrarComprador**  un comprador la cual recibe como parámetros el nombre cliente, su cédula, y su país, además la función tiene el modificador noEsDueno, se comprueba que la cédula y direcciones del comprador sean únicas dentro del contrato instancia la estructura de comprador y se agrega al mapping de compradores, también se agrega la cédula al mapping de cédulas para saber que esta cédula fue ya tomada.

**Registro de productos**

Mediante la función **registrarProductos** la cual tiene como parámetros el nombre del producto, su precio , unidades y descripción, además tiene como modificador esDueno. Primero la función verifica que el nombre sea único, luego mira que su precio sea mayor a cero. Luego el precio ingresado se multiplicar por 10^18 para convertir el precio ingresado de wei a ethers. Por último se agrega este producto al mapping de productos

**Destrucción del contrato**

Para esta tarea se creo la funcion **cerrarTienda** la cual usa el modificador es Dueno, una vez dentro evalúa el valor de la variable global IntentosDestrucción, en caso de que esta sea menor a 2 aumenta en 1 su valor, de lo contrario se usa la función propia del lenguaje **selfdestruct** a la cual se le pasa como parámetro la variable global dueno, allí se cierra el contrato y se transfieren todos los fondos al dueño que instancio la tienda Vanessa Daou.

**Nota**

Este contrato requiere más del gas inicial que da el editor, por favor aumentarlo.