**Informe segundo trabajo – Segunda parte - Bases de Datos II – 2021-1**

***Integrantes:***

*Aristizabal Giraldo Salomé - Rendón Giraldo Sebastián - Valencia Zapata Santiago Alexis*

El siguiente informe presenta la solución dada para cada uno de los requisitos del contrato pedido, codificados dentro del archivo *ContratoVanessaDaou.sol*.

Cabe resaltar que se resolvió el contrato bajo la suposición de que un *address* solo puede hacer referencia a un solo cliente.

Primero se crean 3 variables globales: *owner* donde se guardará la dirección del dueño del contrato, *deleteTries* que guardará los intentos de borrado restantes que le quedan al dueño antes de que sea borrado por completo y *totalDebts* que guardará las deudas totales acumuladas no pagadas.

El constructor no recibe parámetros, asigna a *owner* la dirección de quién llamó al contrato, define el *deleteTries* en 3 y el *totalDebts* en 0.

Se crean dos estructuras, *Product* que tiene nombre (*name*) y descripción (*description*) como *strings*, y unidades en inventario (*units*) y precio en Ethers (*price*) como *uints*; y la otra estructura, *Client*, que guarda la dirección del cliente (*clientAddress*), además de su nombre (*name*) y país (*country*) en *strings* y el código (*code*), total gastado en compras individualmente (*totalExpend*) y su deuda individual (*debt*) en *uints*.

También se crean varios *mappings*, *products* guarda estructuras *Product* identificadas con un *string* referente al nombre, *clients* guarda estructuras *Client* identificadas con un *uint* referente a su código; *clientCodeByAddress* mapea los códigos de las estructuras de *clients* con un *address* referente al *sender* que creó el respectivo *client*; *productNames* y *clientCodes* mapean booleanos con clave *string* referente al nombre de un producto y *uint* referente al código de un cliente, respectivamente, para identificar si estos ya están en uso (*true*) o no (*false*); *expendByCountries* y *debtsByCountries* mapean el valor de lo gastado y de la deuda respectivamente en *Ethers* guardados en *uints* para cada país representando su nombre en un *string*.

Se implementan varios *modifiers*, *isOwner* verifica que el *sender* sea el *owner*, *isNotClient* verifica que el *address* del *sender* no esté registrado como cliente, mientras que *isClient* hace lo contrario, verificando en *clientCodeByAddress*; *clientCodeNotExists* verifica que un código ingresado no se encuentre en *clientsCodes*; *hasNotDebt* verifica que el cliente del *sender* no tenga deuda, *hasDebt* hace lo contrario; *isPositive* verifica que un número ingresado sea mayor a 0; *productNotExists* verifica en *productNames* que no exista un producto dado un nombre, *productExists* verifica lo contrario; *productAvailable* verifica que un producto tenga existencias en inventario; *exactPrice* verifica que el *value* enviado sea exactamente igual a un precio dado en *Ethers*.

*newProduct* recibe un *name* y un *description* en *string*, y un *units* y *Price* en *uint*, se verifica con *isOwner*, *productNotExists* para el *name* y *isPositive* para *Price* y *units*. Genera un *Product* con los parámetros recibidos y lo almacena en *products* con el *name* como clave, y en *productNames* guarda un *true* con la misma clave.

*newClient* recibe un *code* en *uint* y un *name* y *country* en *string*, se verifica con *isNotClient*, y con *isPositive* y *clientCodeNotExists* para el *code*. Genera un *Client* con los parámetros, y le adiciona el *sender* como *clientAddress*, y se establece el *debt* y el *totalExpend* en 0. Luego se guarda en *clients* con el *code* como llave, y se pone un *true* en *clientCodes* con la misma llave, y se agrega el *code* al *clientCodeByAddress* con el *sender* como llave.

*finalPrice* recibe el nombre de un producto, se verifica con *isClient* y *productExists* para el nombre del producto. Se llama la información del cliente del *sender* y del producto y se verifica que el cliente haya gastado más de 50 *Ethers* y que el precio del producto sea mayor o igual a 3 para aplicar el descuento de 3 *Ethers*, en otro caso, se retorna el precio base.

*productInformation* recibe el nombre de un producto, se verifica con *isClient* y con *productExists* para el nombre, retorna la información del producto dado.

*buyProduct* es una función *payable* que recibe el nombre de un producto, se verifica con *isClient*, *productExists* y *productAvailable* para el nombre, *hasNotDebt* y *exactPrice* usando la función *finalPrice* y el nombre. Define un *buyPrice* con *finalPrice*, se lo suma al totalExpend del cliente y al *expendByCountries* con el país del cliente, además, disminuye el *units* del producto en -1, y actualiza el *clients* y el *products*.

*creditProduct* funciona de manera similar a *buyProduct* pero esta no es *payable*, y difiere en que en vez de actualizar al *totalExpend* del cliente y al *expendByCountries*, actualiza al *debt* del cliente y el *debtsByCountries* del país del cliente, y además, el *totalDebts* sumándole el *buyPrice*, que se calcula igual.

*debtInformation* se verifica con *isClient* y retorna el valor *debt* del *Client* del *sender*.

*payDebt* es una función *payable*, que se verifica con *isClient*, *hasDebt* y *exactPrice* con la función *debtInformation*, la función extrae un *buyPrice* con *debtInformation*, se lo suma al *totalExpend* del cliente y al *expendByCountries* del país del cliente, y se lo resta al *debt* del cliente, al *debtsByCountries* del país del cliente y al *totalDebts*.

*destructContract* es una función *payable*, se verifica con *isOwner*, y disminuye el *deleteTries*, y si este llega a cero en una de sus llamadas, hace uso del *selfdestruct* con el *owner* como parámetro, para destruir el contrato y darle las ventas pagadas al *owner*.

*purchasesInformation* se verifica con *isClient* y permite ver al cliente su *totalExpend*.

*sellsInCountry*, *debtsInCountry* y *sellsAndDebtsInCountry* se verifican con *isOwner* y reciben el nombre de un país para poder ver las ventas, deudas y el acumulado de ambas consultando el *expendByCountries* y el *debtsByCountries* con el nombre dado.

*totalPurchasesInformation*, *totalDebtsInformation* y *totalPurchasesWithDebtsInformation* se verifican con *isOwner* y permiten ver al dueño, el total en compras, en deudas y las dos acumuladas respectivamente, consultando el *address(this).balance*, que devuelve el valor de las compras en *weis*, por lo que se le aplica un factor de conversión a *Ethers* de 1 a 10^18, y las deudas, por medio de *totalDebts*.