Memoria Práctica 2 EDAT

Alejandro Monterrubio Navarro

Héctor Martín Cuadra

Grupo 1272

Todos los programas realizan unas acciones antes de ejecutarse, las cuáles son:

1. Se conecta a la base de datos y comprueba si se ha realizado correctamente.
2. Asigna una sentencia.
3. Prepara la sentencia.
4. Asigna el parámetro que recibe la función.
5. Ejecuta la sentencia.
6. Entra en un bucle donde el programa busca una fila, almacena cada resultado y lo muestra en la pantalla.
7. Libera toda la memoria.
8. Se desconecta de la base de datos.

Distintas funciones solicitadas:

**En las querys, el símbolo “?”, representa el parámetro introducido por el usuario.**

* **productsStock()**

Esta función recibe un char\*, que es productcode y la query devuelve el número de unidades en stock. La query simplemente selecciona la cantidad en stock en la que el productcode coincida con el introducido.

Query:

SELECT quantityinstock   
FROM   products   
WHERE  productcode = ?

* **productsFind()**

Esta función recibe un char\*, que es productcode y la query devuelve todos los productos que coincidan con la palabra introducida. En la función se utilizará LIKE en vez de =. También se utiliza un auxiliar para que la palabra se convierta en %palabra% y la detecte correctamente como fue sugerido en clase.

Query:

SELECT productcode,   
       productname   
FROM   products   
WHERE  productname LIKE ?   
ORDER  BY productcode

* **ordersOpen()**

Esta función no recibe ningún argumento y muestra una lista de todos los pedidos que aún no han sido enviados.

Query:

SELECT ordernumber   
FROM   orders   
WHERE  shippeddate IS NULL   
ORDER  BY ordernumber ASC

* **ordersRange()**

Esta función recibe dos fechas y muestra una lista de todos los pedidos que han sido solicitados entre esas fechas. En esta función se tiene que asignar valor a dos parámetros, en la query, la primera ? representa la primera fecha y la segunda ? representa la segunda.

Query:

SELECT ordernumber,   
       orderdate,   
       shippeddate   
FROM   orders   
WHERE  orderdate >= ?   
       AND orderdate <= ?   
ORDER  BY ordernumber ASC

* **ordersDetail()**

Esta función recibe un ordenumber y devuelve una lista en varias líneas, la primera, con la fecha en la que el pedido fue realizado y su status (si el pedido ya se ha enviado o no). La segunda línea muestra el coste total del pedido. Luego se muestra una línea por cada producto con su correspondiente ordernumber, la cantidad de unidades pedidas y el precio por cada unidad. Para realizar esta función se ha implementado con dos querys distintas.

Query 1:

SELECT orderdate,   
       status,   
       *Sum*(quantityordered \* priceeach) AS coste   
FROM   orders   
       natural JOIN orderdetails   
WHERE  ordernumber = ?   
GROUP  BY ordernumber

Query 2:

SELECT productcode,   
       quantityordered,   
       priceeach   
FROM   orderdetails   
WHERE  ordernumber = ?   
ORDER  BY orderlinenumber

* **customersFind()**

Esta función recibe un nombre o apellido de un cliente y utilizando like y de nuevo un auxiliar para que el nombre sea %nombre% devuelve una lista del nombre y apellido del cliente y su identificador (customernumber).

Query:

SELECT customername,   
       contactfirstname,   
       contactlastname,   
       customernumber   
FROM   customers   
WHERE  contactfirstname LIKE ?   
        OR contactlastname LIKE ?   
ORDER  BY customernumber

* **customersListProducts()**

La función recibe un identificador de cliente (customernumber) y muestra una lista de todos los productos que han sido solicitados por ese cliente, mostrará el nombre del producto y el total de unidades pedidas.

Query:

SELECT productname,   
       *Sum*(quantityordered) AS total   
FROM   orders,   
       orderdetails,   
       products   
WHERE  orders.customernumber = ?   
       AND orderdetails.ordernumber = orders.ordernumber   
       AND orderdetails.productcode = products.productcode   
GROUP  BY products.productcode

* **customersBalance()**

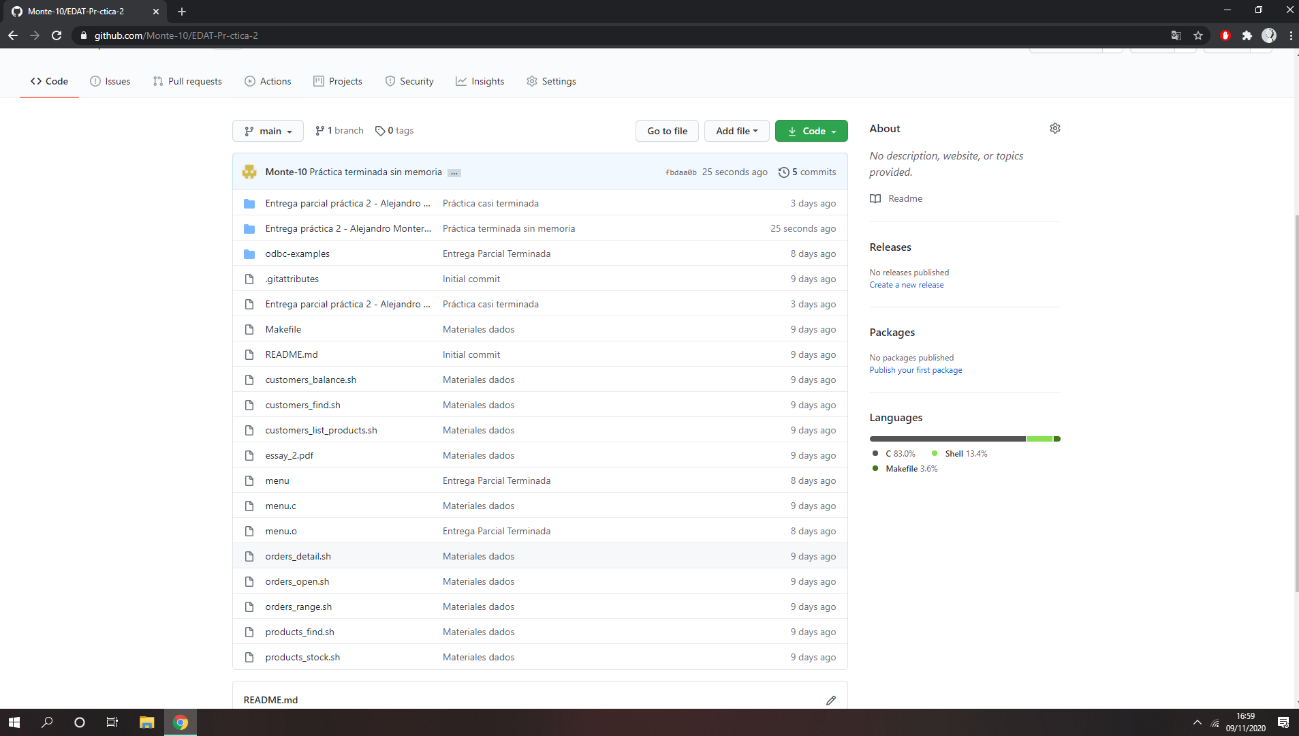
La función recibe un identificador de cliente (customernumber) y devuelve su saldo (suma de los pagos realizados – suma de todos los productos comprados), para realizar esta función se ha vuelto a utilizar dos querys distintas en la misma función. La primera obtiene la suma de los pagos al cliente y la segunda la suma de los precios de cada producto que ha comprado. Luego restamos el primer valor del segundo y obtenemos el valor del saldo.

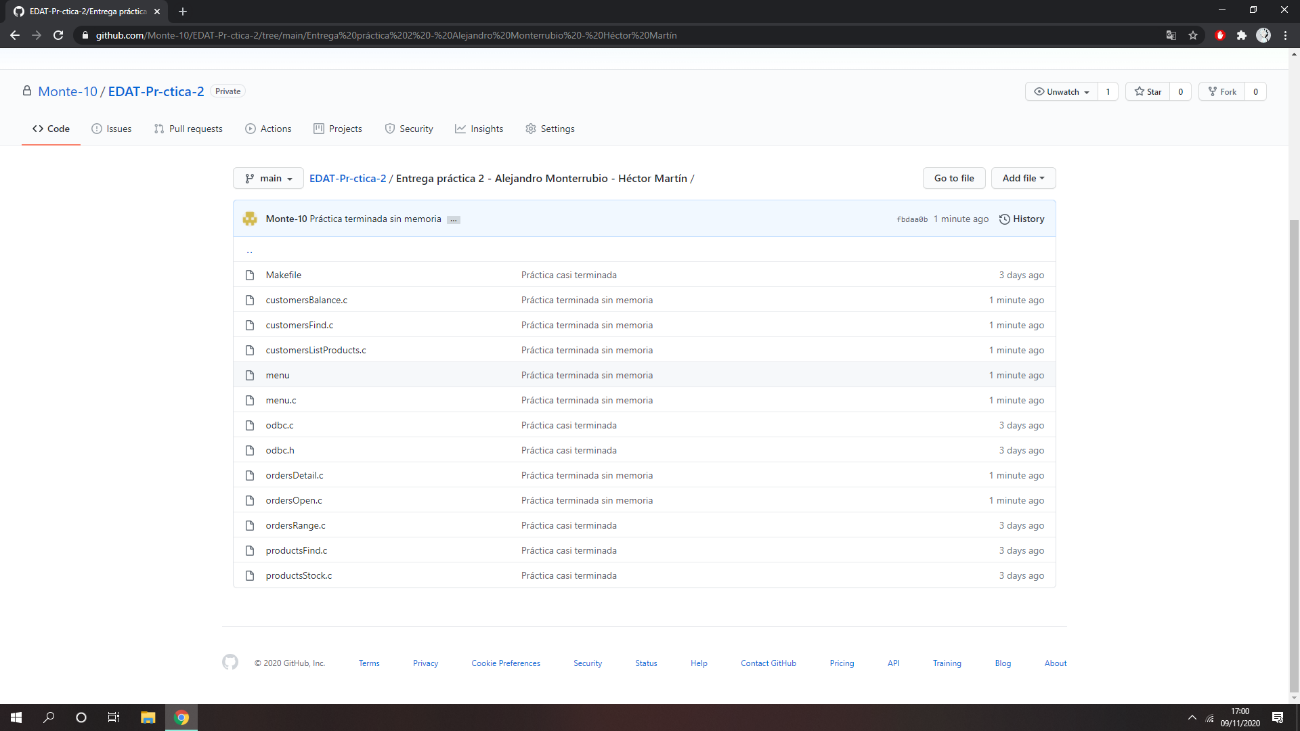
Query 1:

SELECT *Sum*(amount) AS saldo   
FROM   payments   
WHERE  payments.customernumber = ?

Query 2:

SELECT *Sum*(priceeach \* quantityordered) AS sum   
FROM   orders,   
       orderdetails   
WHERE  orders.customernumber = ?   
       AND orders.ordernumber = orderdetails.ordernumber

**CAPTURAS GIT**

****