Documentation: Basket Application

Project information 1

How to install and start the project 2

Requirements 6

Test coverage 10

Project information

To solve the proposed requirements, I have estimated the need to create an application that uses JAVA, as required, and SQL.

The JAVA project has the task of executing the necessary logic for the application and serving as a communication interface with the user through the IDE console. This part is responsible for controlling the flows that make it possible to fulfill each requirement, controlling the options entered by the user to avoid errors, communicating with the database, performing queries, and communicating with the user in the language chosen from the options of English or Spanish.

The SQL part has the mission of controlling the persistence of the data necessary for the application to work. Creating the database and the tables that will contain the information, avoiding duplication of ids, and eliminating records that are no longer necessary to maintain.

The database is made up of three tables: CARTS, which stores the different carts created and their modification date; PRODUCTS, which stores names, descriptions, and prices of the different products available; and CART_PRODUCTS, which stores the products and their quantity, mapping a product id and quantity to a cart id. It also includes a sequence, CART_ID_SEQ, to assign ids to the new carts created, and a cascading restriction to eliminate the CART_PRODUCTS records assigned to a cart id, when it`s deleted.

To carry it out I used the Eclipse and HeidiSQL tools.

How to install and start the project

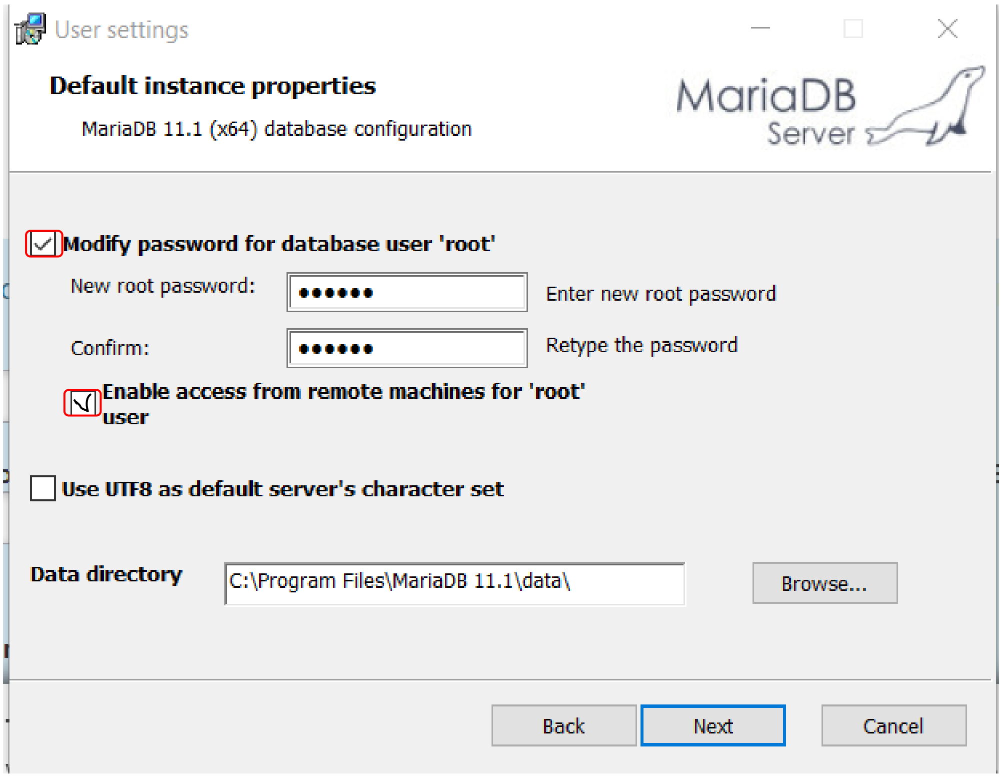
To make the application work, you must prepare the database and the Java project in the IDE.

Database:

For the database I have used MariaDB server (version 11.1.2) and HeidiSQL, a configuration that I recommend using to make the application work faster and easier, avoiding having to touch the dependencies and configuration of the project. It can be downloaded from this link:

https://mariadb.org/download/?t=mariadb&p=mariadb&r=11.1.2&os=windows&cpu=x86_64&pkg=msi&m=fe_up_pt

Run the installer and simply continue to the 'Default instance properties' screen, enable remote access, and configure the default user 'root' with the password 'ff5544' to make the application work with the configuration it already has.



User settings

Default instance properties

MariaDB 11.1 (x64) database configuration

☑ Modify password for database user 'root'

New root password: ●●●●●● Enter new root password

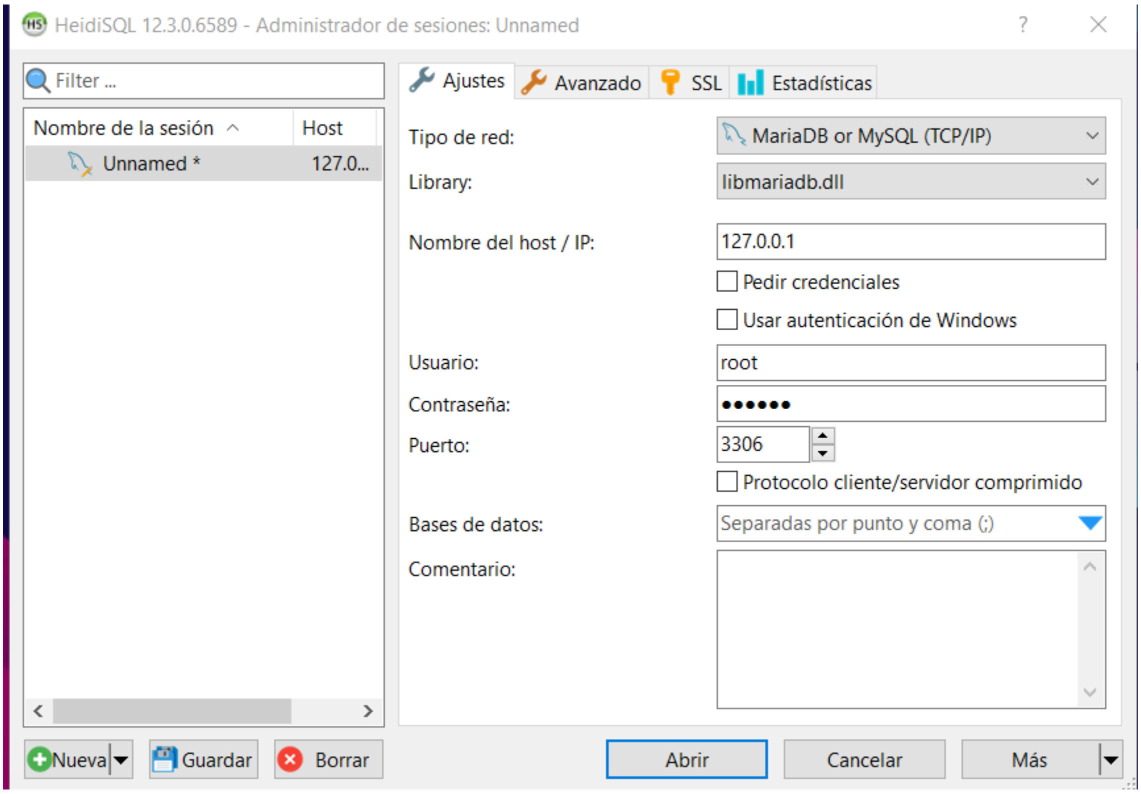
Confirm: ●●●●●● Retype the password

☑ Enable access from remote machines for 'root' user

□ Use UTF8 as default server's character set

Data directory C:\Program Files\MariaDB 11.1\data\

Once the installation process is finished. We open the HeidiSQL application, and create a new session, “Nueva” button at the bottom left, enter the user password ‘ff5544’, and open the session in the central button below.



Tipo de red: MariaDB or MySQL (TCP/IP)

Library: libmariadb.dll

Nombre del host / IP: 127.0.0.1

□ Pedir credenciales

□ Usar autenticación de Windows

Usuario: root

Contraseña: ••••••

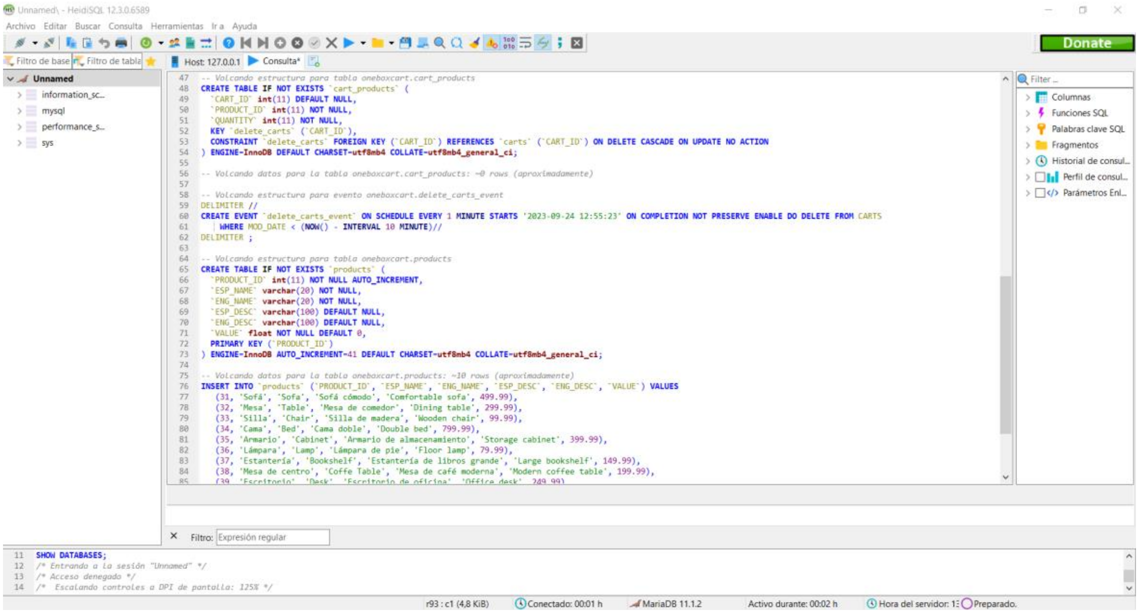
Puerto: 3306

□ Protocolo cliente/servidor comprimido

Bases de datos: Separadas por punto y coma (;)

Comentario:

To create the database, you simply must execute the SQL from the attached oneboxcart.sql file, you can drag the file to the query screen for greater convenience. Go to the query tab, paste the query, and execute it by pressing F9 or the blue play icon.

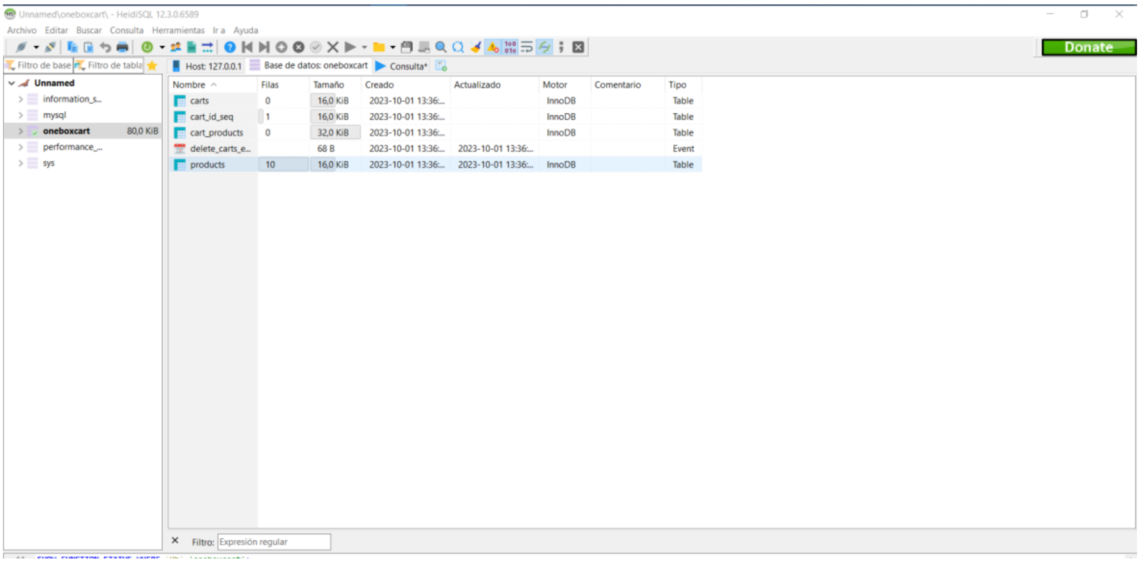


```sql
-- Volcando estructura para tabla oneboxcart.cart_products
CREATE TABLE IF NOT EXISTS `cart_products` (
  `CART_ID` int(11) DEFAULT NULL,
  `PRODUCT_ID` int(11) NOT NULL,
  `QUANTITY` int(11) NOT NULL,
  KEY `delete_carts` (`CART_ID`),
  CONSTRAINT `delete_carts` FOREIGN KEY (`CART_ID`) REFERENCES `carts` (`CART_ID`) ON DELETE CASCADE ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- Volcando datos para la tabla oneboxcart.cart_products: ~0 rows (aproximadamente)

-- Volcando estructura para evento oneboxcart.delete_carts_event
DELIMITER //
CREATE EVENT `delete_carts_event` ON SCHEDULE EVERY 1 MINUTE STARTS '2023-09-24 12:55:23' ON COMPLETION NOT PRESERVE ENABLE DO DELETE FROM CARTS
    WHERE MOD_DATE < (NOW() - INTERVAL 10 MINUTE)//
DELIMITER ;

-- Volcando estructura para tabla oneboxcart.products
CREATE TABLE IF NOT EXISTS `products` (
  `PRODUCT_ID` int(11) NOT NULL AUTO_INCREMENT,
  `ESP_NAME` varchar(20) NOT NULL,
  `ENG_NAME` varchar(20) NOT NULL,
  `ESP_DESC` varchar(100) DEFAULT NULL,
  `ENG_DESC` varchar(100) DEFAULT NULL,
  `VALUE` float NOT NULL DEFAULT 0,
  PRIMARY KEY (`PRODUCT_ID`)
) ENGINE=InnoDB AUTO_INCREMENT=41 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- Volcando datos para la tabla oneboxcart.products: ~10 rows (aproximadamente)
INSERT INTO `products` (`PRODUCT_ID`, `ESP_NAME`, `ENG_NAME`, `ESP_DESC`, `ENG_DESC`, `VALUE`) VALUES
	(31, 'Sofá', 'Sofa', 'Sofá cómodo', 'Comfortable sofa', 499.99),
	(32, 'Mesa', 'Table', 'Mesa de comedor', 'Dining table', 299.99),
	(33, 'Silla', 'Chair', 'Silla de madera', 'Wooden chair', 99.99),
	(34, 'Cama', 'Bed', 'Cama doble', 'Double bed', 799.99),
	(35, 'Armario', 'Cabinet', 'Armario de almacenamiento', 'Storage cabinet', 399.99),
	(36, 'Lámpara', 'Lamp', 'Lámpara de pie', 'Floor lamp', 79.99),
	(37, 'Estantería', 'Bookshelf', 'Estantería de libros grande', 'Large bookshelf', 149.99),
	(38, 'Mesa de centro', 'Coffe Table', 'Mesa de café moderna', 'Modern coffee table', 199.99),
	(39, 'Escritorio', 'Desk', 'Escritorio de oficina', 'Office desk', 249.99)
```

```
SHOW DATABASES;
/* Entrando a la sesión "Unnamed" */
/* Acceso denegado */
/*  Escalando controles a DPI de pantalla: 125% */
```

r93 : c1 (4,8 KiB) | Conectado: 00:01 h | MariaDB 11.1.2 | Activo durante: 00:02 h | Hora del servidor: 1: | Preparado.

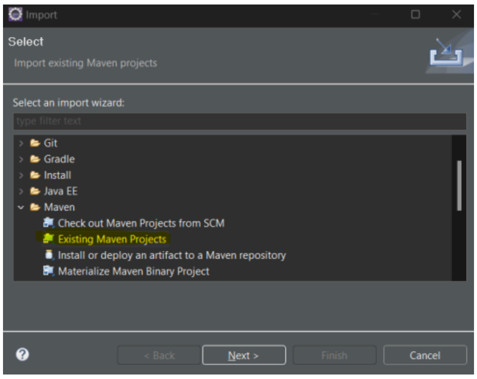
Finally, we update by pressing F5 or the icon of a green circle with an arrow inside, so that the new database and its content appear.


Unnamed\oneboxcart\ - HeidiSQL 12.3.0.6589
Archivo Editar Buscar Consulta Herramientas Ir a Ayuda
Donate
Filtro de base
Filtro de tabla
Host: 127.0.0.1
Base de datos: oneboxcart
Consulta*
Unnamed
information_s...
mysql
oneboxcart 80,0 KiB
performance_...
sys
Filtro: Expresión regular

Nombre	Filas	Tamaño	Creado	Actualizado	Motor	Comentario	Tipo
carts	0	16,0 KiB	2023-10-01 13:36:...		InnoDB		Table
cart_id_seq	1	16,0 KiB	2023-10-01 13:36:...		InnoDB		Table
cart_products	0	32,0 KiB	2023-10-01 13:36:...		InnoDB		Table
delete_carts_e...		68 B	2023-10-01 13:36:...	2023-10-01 13:36:...			Event
products	10	16,0 KiB	2023-10-01 13:36:...	2023-10-01 13:36:...	InnoDB		Table

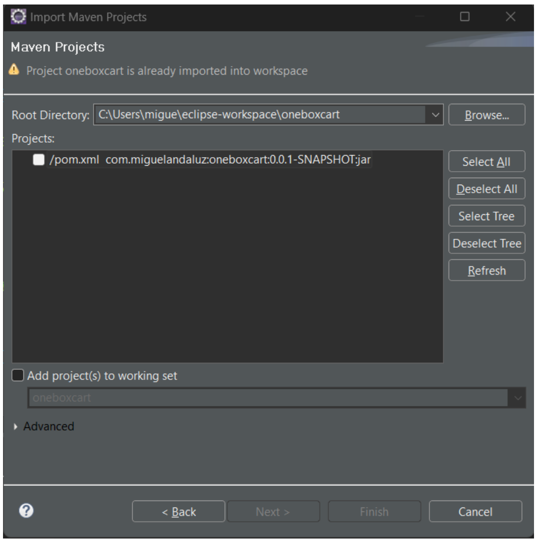
JAVA project:

For development in JAVA, I have used Eclipse IDE. To make the project work, we must unzip the ZIP in the workspace folder we are using, and then import it as an existing Maven project...



Import existing Maven projects

We look for the project in the folder and select it.



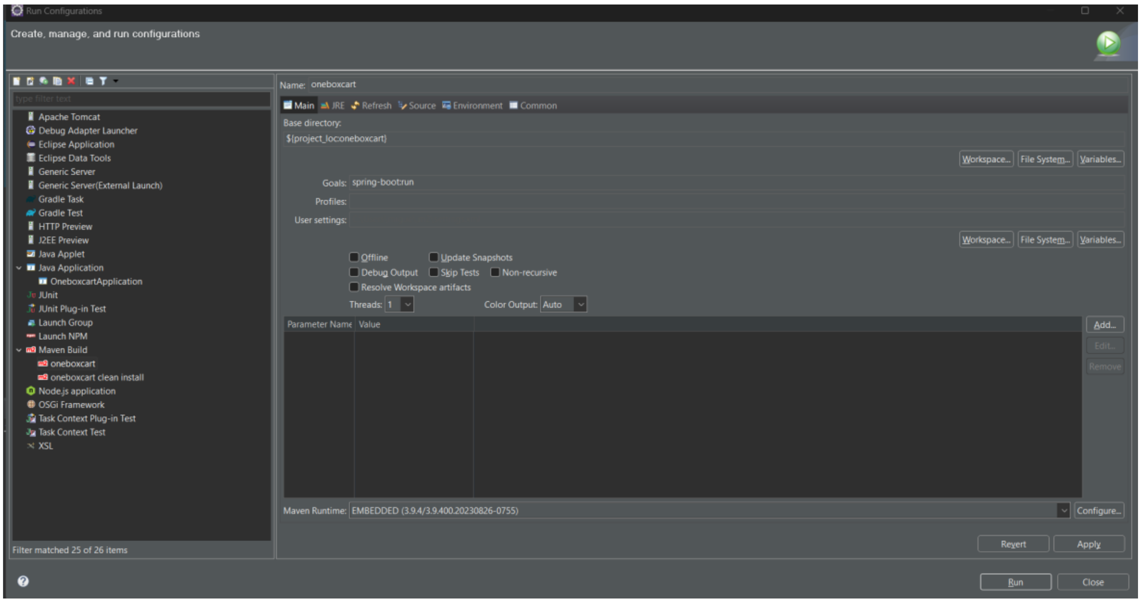
Project oneboxcart is already imported into workspace

Root Directory: C:\Users\migue\eclipse-workspace\oneboxcart

☐ /pom.xml com.miguelandaluz:oneboxcart:0.0.1-SNAPSHOT:jar

To start the application there are two options:

- Run it as a java application: search for the OneboxcartApplication class and select Run as -> Java application in Eclipse the first time, the rest of the times you can directly select the Oneboxcartapplication option.
- The second option is to build it as a Maven project, selecting the project, Run as -> Maven build… and in the Run Configurations pop-up screen, setting the goal: spring-boot:run and in the Common tab, choosing the encoding: use system encoding… to avoid display problems with special Spanish characters, and press Run. After the first configuration, the configuration will be saved with the configuration name.



Name: oneboxcart

Base directory: ${project_loc:oneboxcart}

Goals: spring-boot:run

Profiles:

User settings:

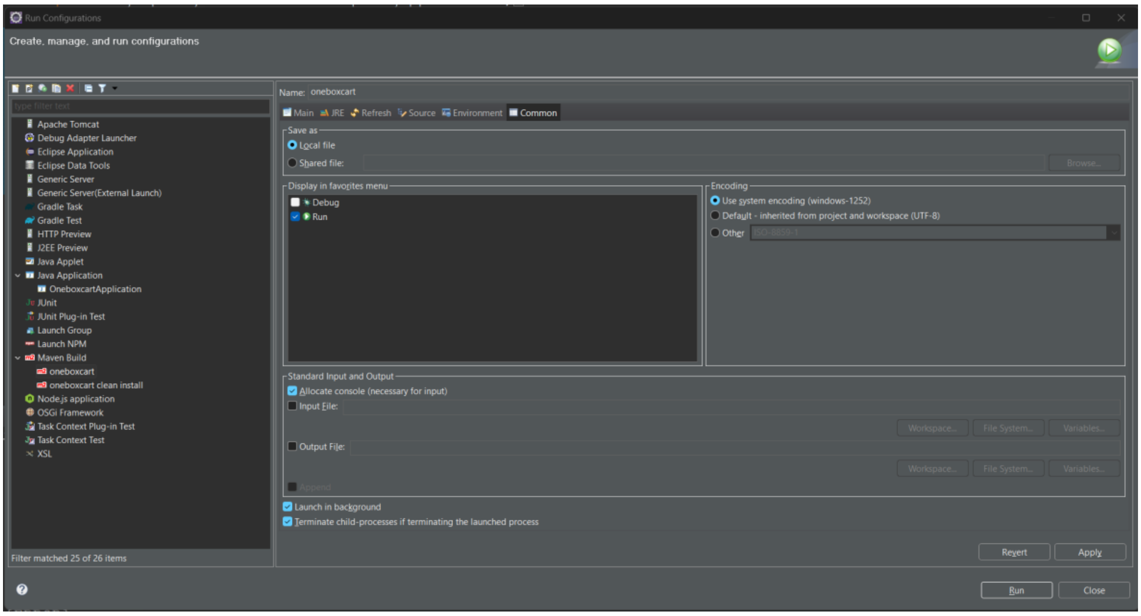
□ Offline □ Update Snapshots

□ Debug Output □ Skip Tests □ Non-recursive

□ Resolve Workspace artifacts

Threads: 1 Color Output: Auto

Maven Runtime: EMBEDDED (3.9.4/3.9.400.20230826-0755)



Name: oneboxcart

Save as
● Local file
○ Shared file:

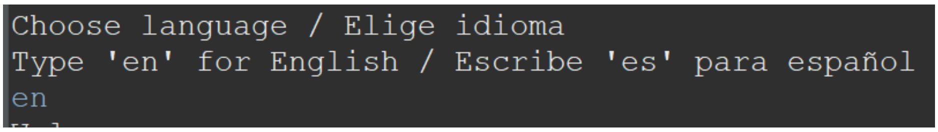
Display in favorites menu
□ Debug
☑ Run

Encoding
● Use system encoding (windows-1252)
○ Default - inherited from project and workspace (UTF-8)
○ Other ISO-8859-1

Standard Input and Output
☑ Allocate console (necessary for input)
□ Input File:
□ Output File:
□ Append

☑ Launch in background
☑ Terminate child-processes if terminating the launched process

Once the initialization is complete, we must choose the language, entering 'es' for Spanish, or 'en' for English through the console.

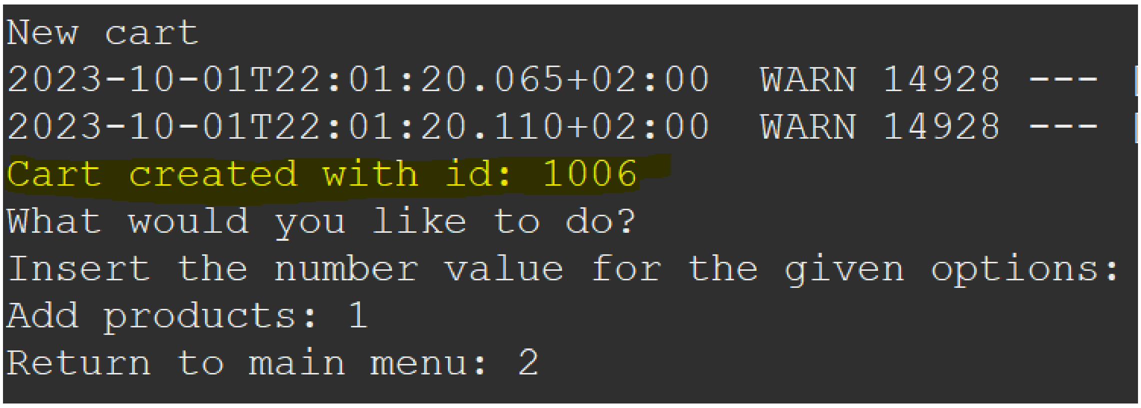


```
Choose language / Elige idioma
Type 'en' for English / Escribe 'es' para español
en
```

Requirements

App generated cart ID:

To generate an id for the cart, I have created a sequence in SQL, 'cart_id_seq', which gives us a value every time it is queried. This value is obtained by adding one to the previous cached value, and which will be inserted into the table as the basket id, CART_ID of the CARTS table.



```
New cart
2023-10-01T22:01:20.065+02:00  WARN 14928 ---
2023-10-01T22:01:20.110+02:00  WARN 14928 ---
Cart created with id: 1006
What would you like to do?
Insert the number value for the given options:
Add products: 1
Return to main menu: 2
```

Deleting carts after 10 minutes of inactivity:

For the deletion, I have created an event in SQL, 'delete_carts_event', which runs every minute, and checks that no more than 10 minutes have passed since the cart modification time, MOD_DATE of the CARTS table, and the current time. With a restriction on cascade deletion by cart id, all products associated with that id are removed from the CART_PRODUCTS table.

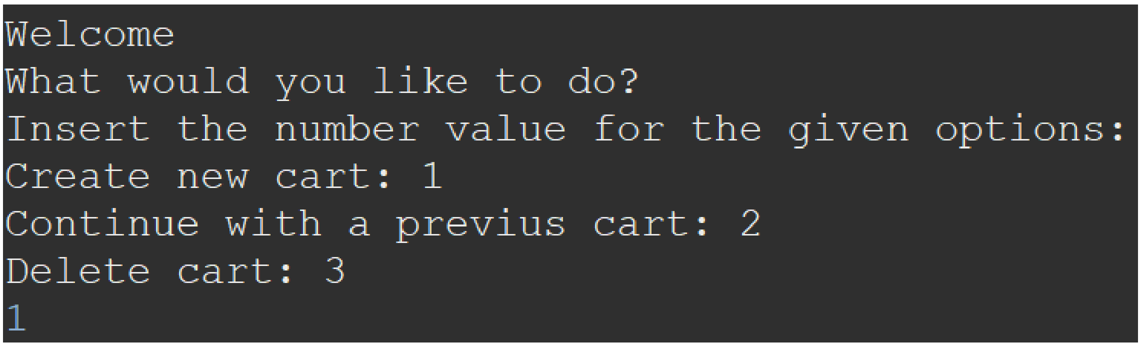
It is considered inactivity if that basket has not been consulted by id, or if no products have been added.

For its correct functioning, it is necessary for the database to have events activated, which is achieved by executing the following query, which is already provided in the oneboxcarty.sql file, but perhaps if it does not work it will be necessary to execute it again.

SET GLOBAL event_scheduler = ON;

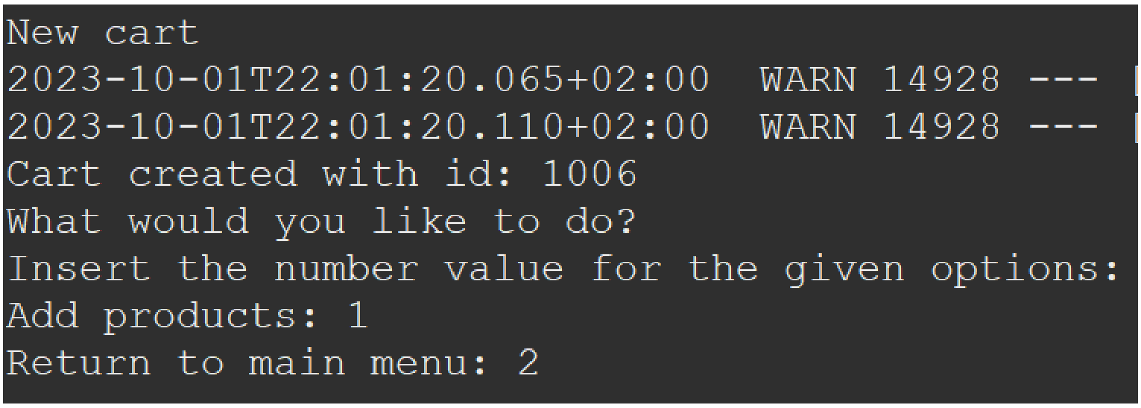
Creation of cart with capacity for products:

To create a cart we must enter 1 in the console, and press enter.



```
Welcome
What would you like to do?
Insert the number value for the given options:
Create new cart: 1
Continue with a previus cart: 2
Delete cart: 3
1
```

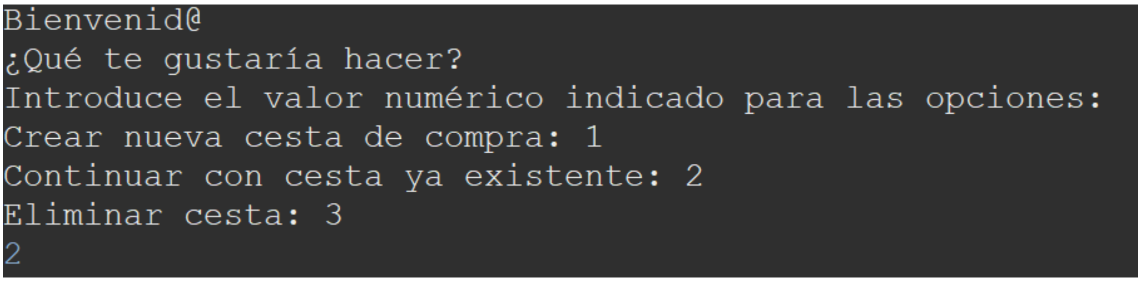
The application will consult the sequence 'cart_id_seq' to obtain the id of the cart and will perform an insert in the CARTS table creating the cart. Once the cart is created, we are allowed to add products or return to the main menu.



```
New cart
2023-10-01T22:01:20.065+02:00  WARN 14928 ---
2023-10-01T22:01:20.110+02:00  WARN 14928 ---
Cart created with id: 1006
What would you like to do?
Insert the number value for the given options:
Add products: 1
Return to main menu: 2
```

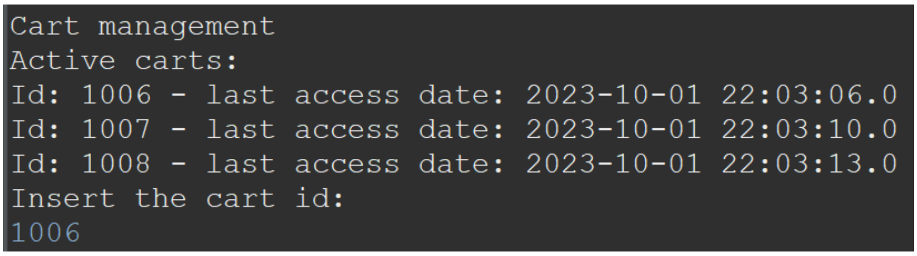
Obtaining cart information by ID:

To obtain information about a cart we must enter 2 in the console, and press enter.

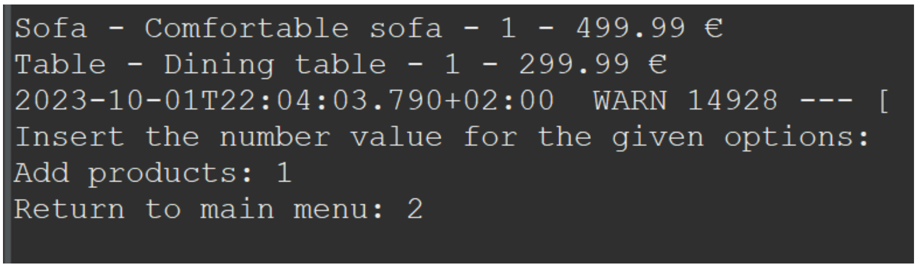


```
Bienvenid@
¿Qué te gustaría hacer?
Introduce el valor numérico indicado para las opciones:
Crear nueva cesta de compra: 1
Continuar con cesta ya existente: 2
Eliminar cesta: 3
2
```

The application will perform a query in the database to show what carts are currently available. To consult the information, we enter the id of the cart to be consulted in the console and press enter. With this, the application shows us a list of the products in the cart, with their name, description, quantity, and the total cost of that product line.



```
Cart management
Active carts:
Id: 1006 - last access date: 2023-10-01 22:03:06.0
Id: 1007 - last access date: 2023-10-01 22:03:10.0
Id: 1008 - last access date: 2023-10-01 22:03:13.0
Insert the cart id:
1006
```

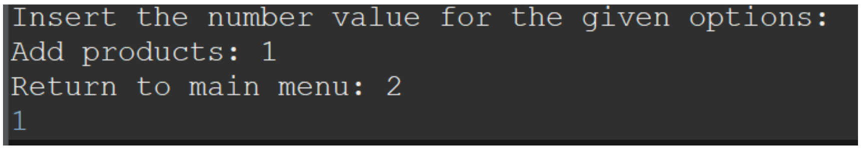


```
Sofa - Comfortable sofa - 1 - 499.99 €
Table - Dining table - 1 - 299.99 €
2023-10-01T22:04:03.790+02:00  WARN 14928 --- [
Insert the number value for the given options:
Add products: 1
Return to main menu: 2
```

Once consulted, the application will allow us to add products to that cart or return to the main menu.

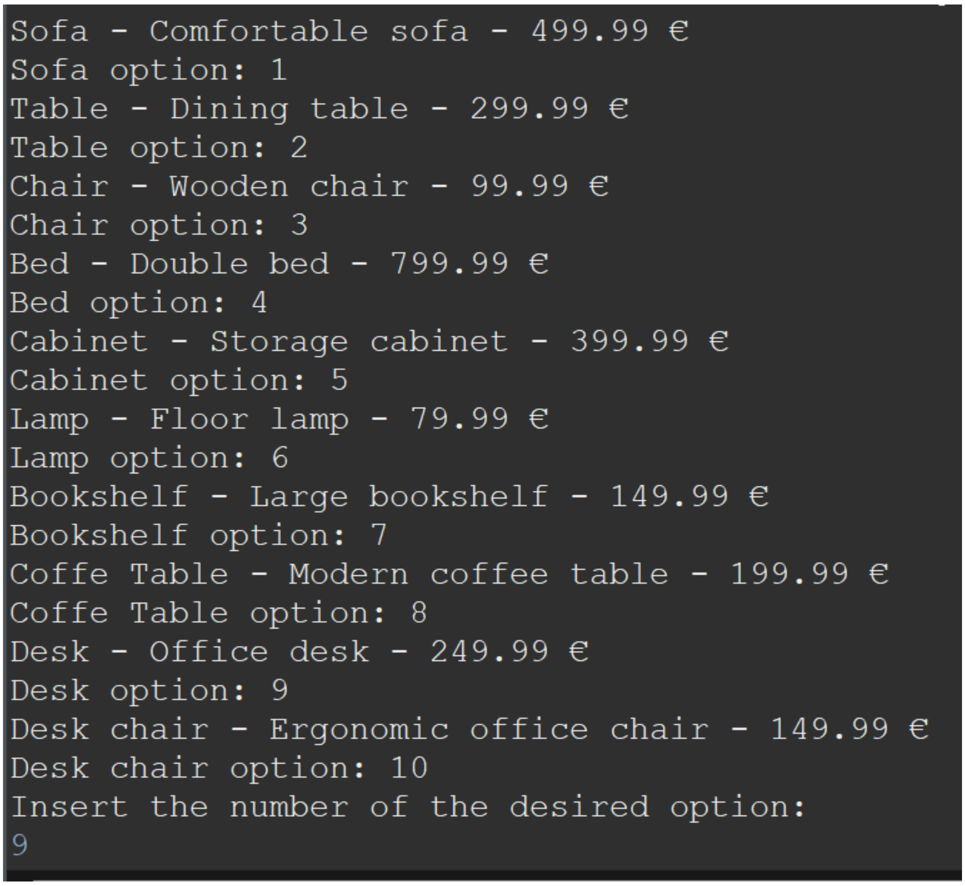
Adding products to cart:

To reach the addition of products we must have followed the course of the previous requirements, options 1 or 2, and have chosen the option to add products.



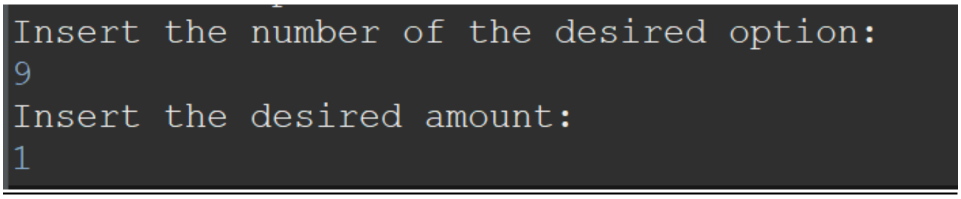
```
Insert the number value for the given options:
Add products: 1
Return to main menu: 2
1
```

The application will show us a list of the products available through a database query, with their name, description (depending on the chosen language), and value.



```
Sofa - Comfortable sofa - 499.99 €
Sofa option: 1
Table - Dining table - 299.99 €
Table option: 2
Chair - Wooden chair - 99.99 €
Chair option: 3
Bed - Double bed - 799.99 €
Bed option: 4
Cabinet - Storage cabinet - 399.99 €
Cabinet option: 5
Lamp - Floor lamp - 79.99 €
Lamp option: 6
Bookshelf - Large bookshelf - 149.99 €
Bookshelf option: 7
Coffe Table - Modern coffee table - 199.99 €
Coffe Table option: 8
Desk - Office desk - 249.99 €
Desk option: 9
Desk chair - Ergonomic office chair - 149.99 €
Desk chair option: 10
Insert the number of the desired option:
9
```

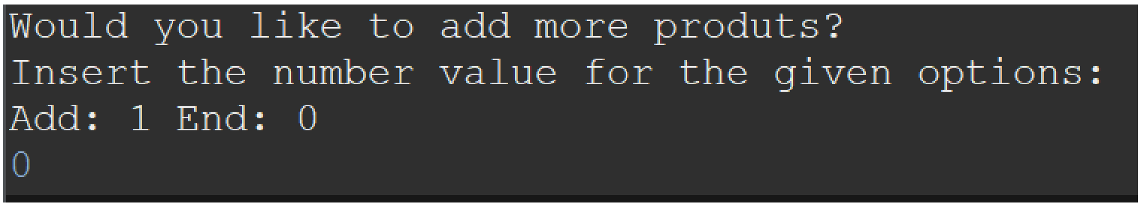
We must enter the number of the desired option and press enter, and finally choose the desired quantity, and press enter.



```
Insert the number of the desired option:
9
Insert the desired amount:
1
```

Once this is done, the application will check if this product is already in our cart or not. If found, an UPDATE will be executed to the CART_PRODUCTS table, making a sum between the previous quantity and the new quantity. If it is not found, an INSERT will be made to said table assigning the product and the quantity to the cart.

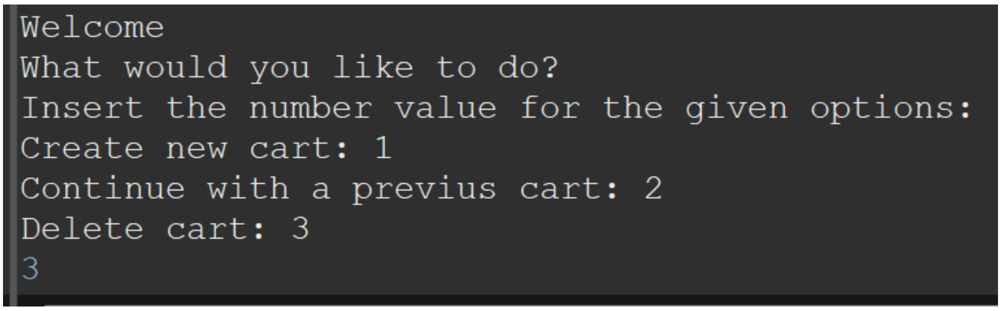
Once this process is finished, we can continue adding products, or we can return to the main menu.



```
Would you like to add more produts?
Insert the number value for the given options:
Add: 1 End: 0
0
```

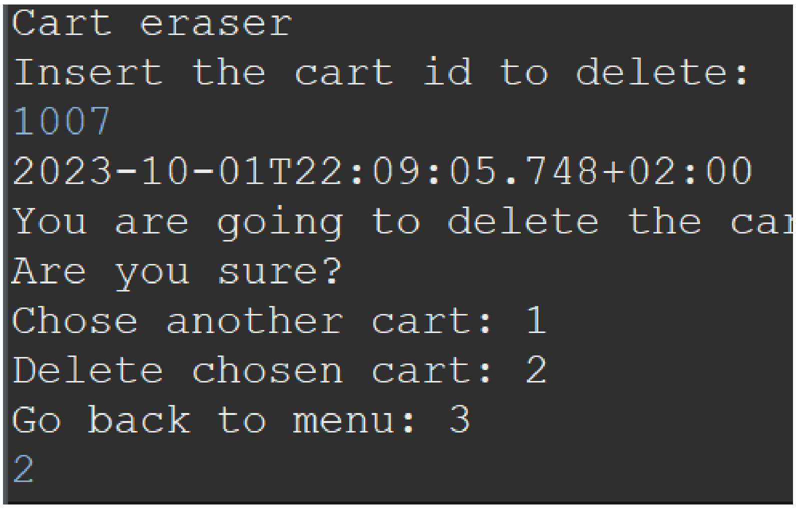
Cart removal:

To delete a cart, we must enter 3 in the console, and press enter.



```
Welcome
What would you like to do?
Insert the number value for the given options:
Create new cart: 1
Continue with a previus cart: 2
Delete cart: 3
3
```

At this point we must know the id of the basket that we want to delete. The application will check that the cart exists so it can be deleted. Once the id of the cart to be deleted has been inserted, a security question will be asked to allow us to choose between selecting another id to delete, option 1, deleting the cart with the entered id, option 2, or exiting to the main menu, option 3.



```
Cart eraser
Insert the cart id to delete:
1007
2023-10-01T22:09:05.748+02:00
You are going to delete the ca
Are you sure?
Chose another cart: 1
Delete chosen cart: 2
Go back to menu: 3
2
```

Option 1 again allows us to enter an id to delete. Option 2 deletes the basket whose id we have entered, and all the products associated with it thanks to the cascading restriction.

Test coverage

The flows and different options have been tested several times, in both available language options, checking the correct functioning of all the functionalities and requirements, and searching for and solving the different errors. To do this, I have not limited myself to the options available to the application, but rather I have carried out tests of what could be misuse by a user, to test the robustness of the application and the solidity of the flows.

To demonstrate its correct operation, I provide images for each requirement as evidence of correct operation and the tests carried out. Except for automatic deletion for which I cannot provide graphical proof.

Furthermore, I have carried out these tests on two different PCs, to verify that the configuration to run this project, which I have presented in this document, is valid.