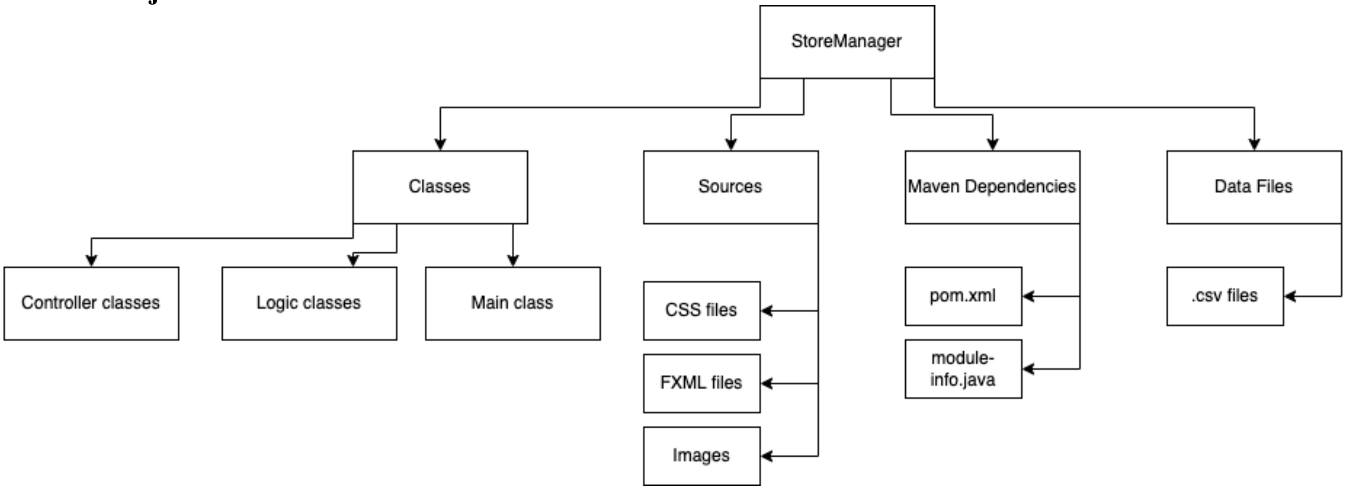
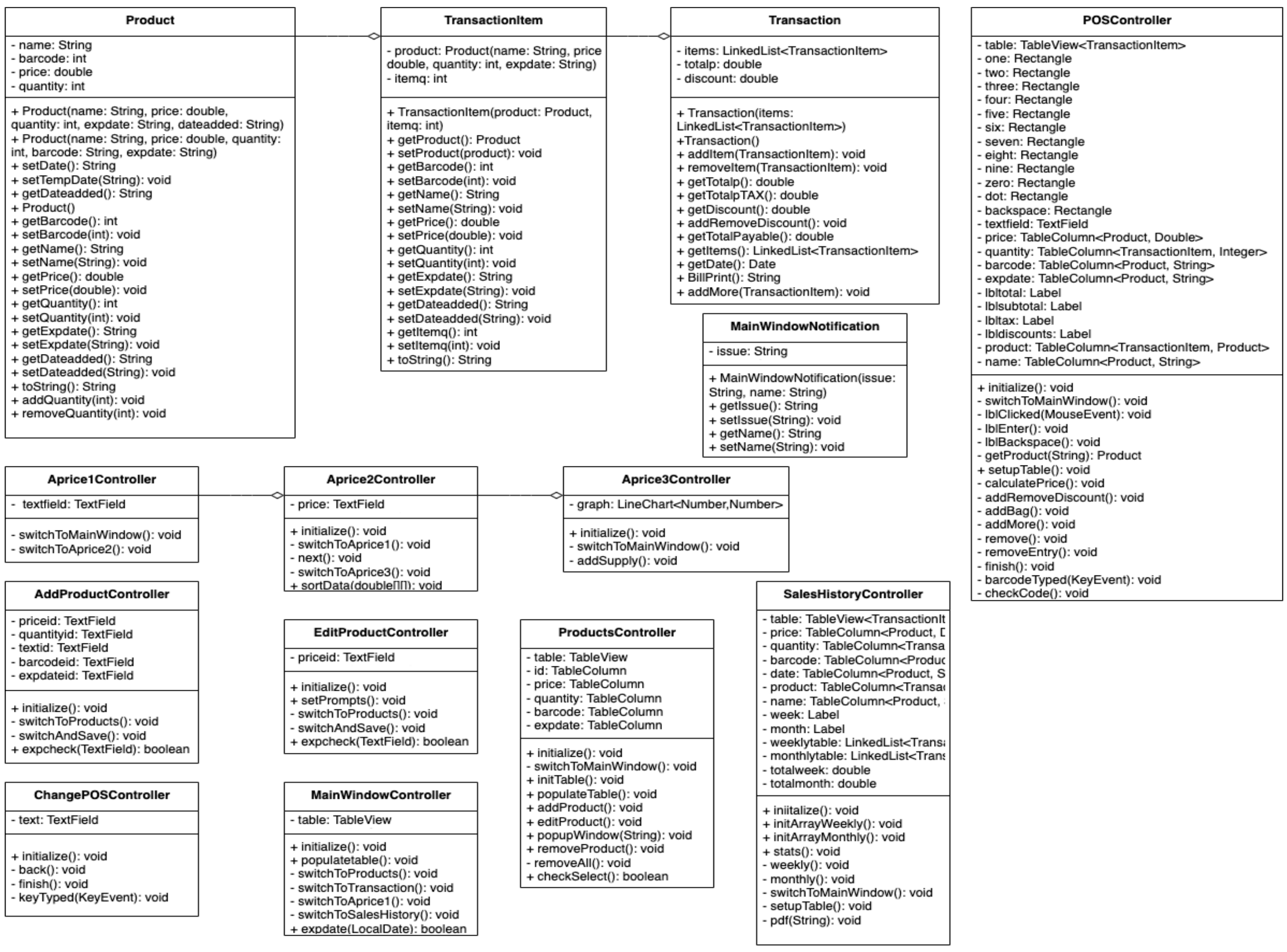
Criterion B: Design

1. Project structure scheme



2. UML diagrams

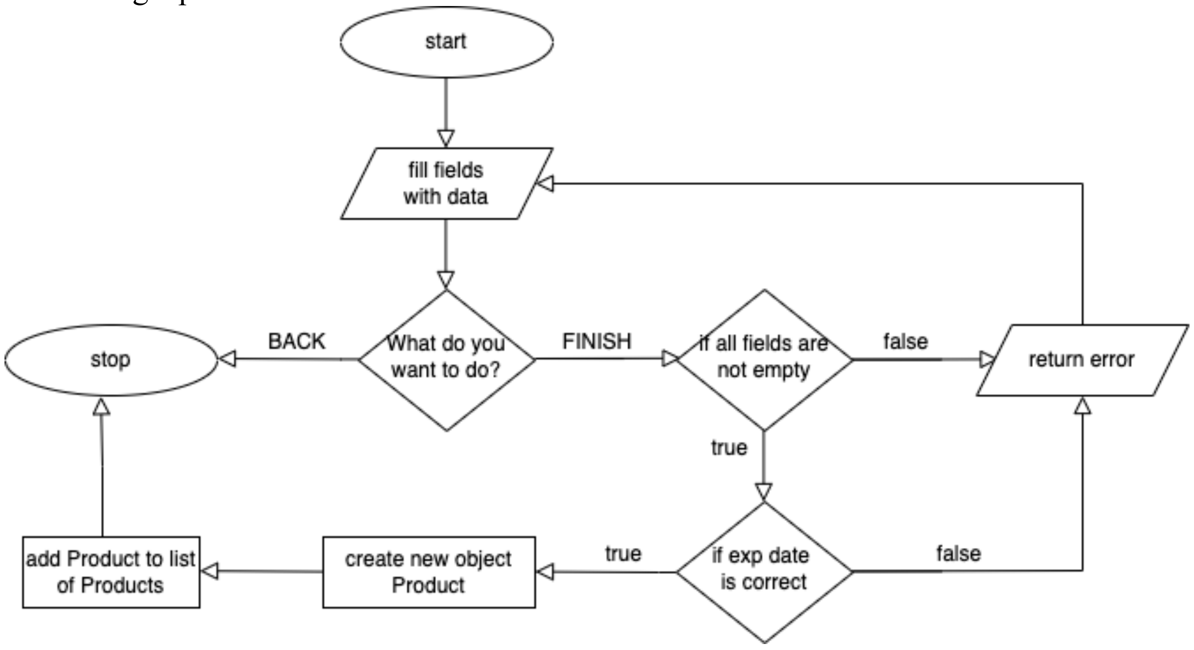


3. Classes purpose

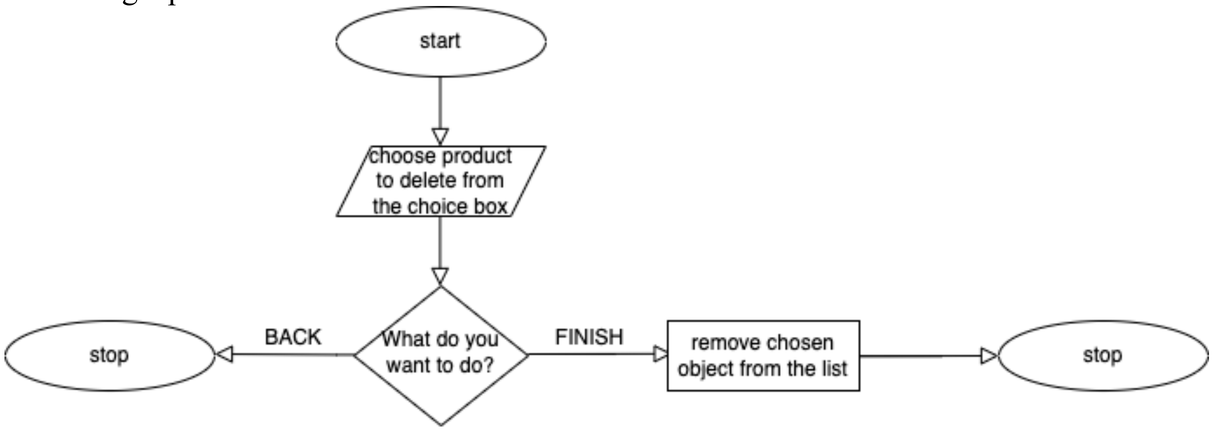
Product	A class responsible for all operations that can be done for a certain product. Based on the Product class, lists of the products are created in the graphical interface.
TransactionItem	It stores designated information about a product in a specific transaction. Important for transactions with many quantities.
Transaction	It is responsible for setting up a final version of a receipt for the customer.
Aprice1Controller	Responsible for receiving datapoints number for the demand and supply curves.
Aprice2Controller	Allows for entering demand and supply curves data.
Aprice3Controller	Creates a demand and supply curve showing the appropriate price for the good.
ProductsController	Displays the table of products and allows for modifying them.
AddProductController	Allows for adding a new product.
EditProductController	Allows for editing a given product.
MainWindowController	Allows for switching to different scenes, displays the notification table.
MainWindowNotification	Responsible for handling notifications displayed in the table on the main window.
POSController	Responsible for the Point of Sale system.
ChangePOSController	Calculates the change to be given to the customer. Modifies products quantity after transaction.
SalesHistoryController	Shows revenue from the previous week and month, table with last sold products. It allows for generating a PDF file with a revenue report.
App.java	Loads file data into LinkedLists, holds passing parameters and methods for reading and writing .csv files.
Main.java	Main class, launches arguments from App.java.

4. Flowcharts

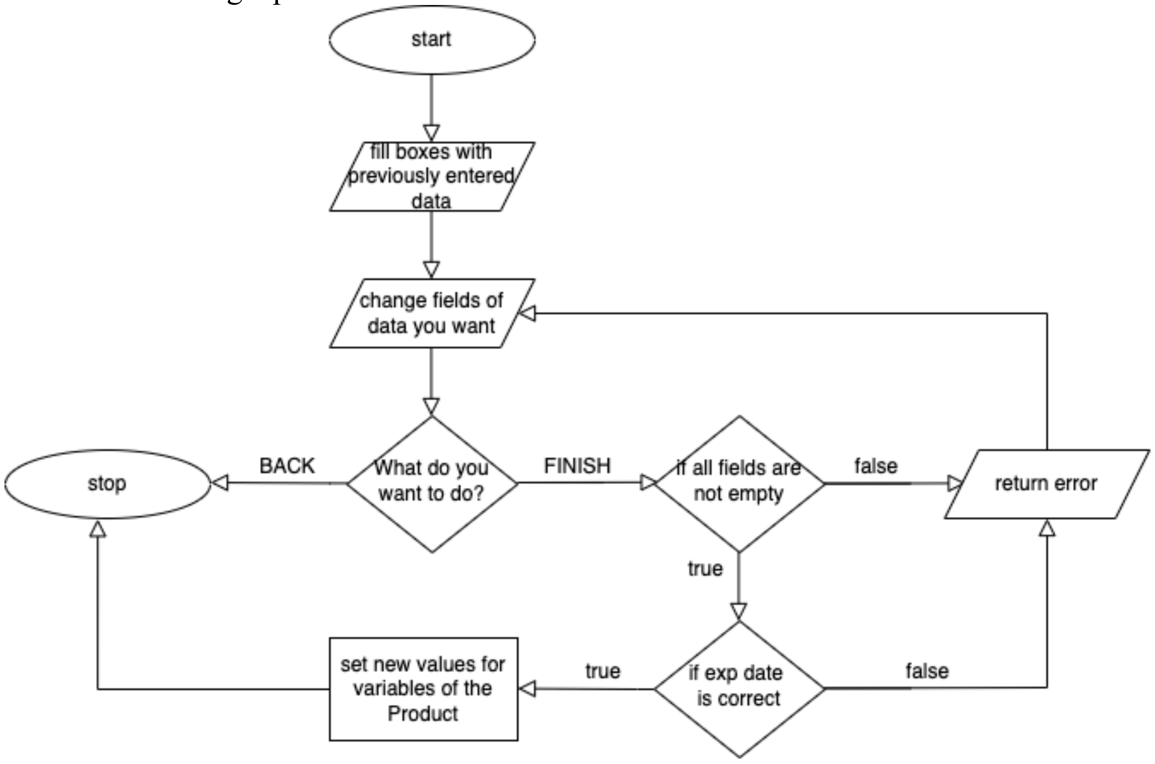
4.1 Adding a product



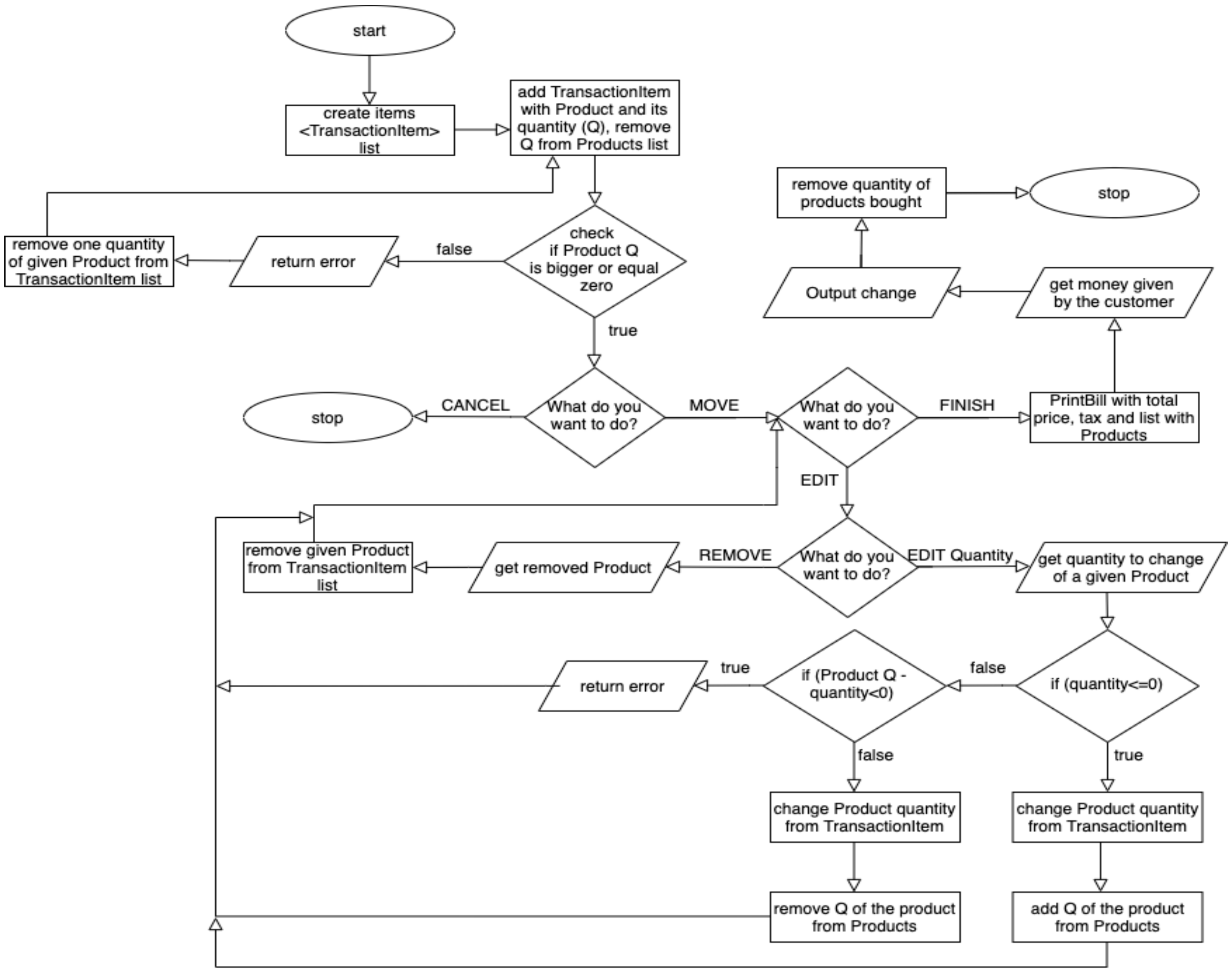
4.2 Removing a product


start
choose product to delete from the choice box
What do you want to do?
BACK
stop
FINISH
remove chosen object from the list
stop

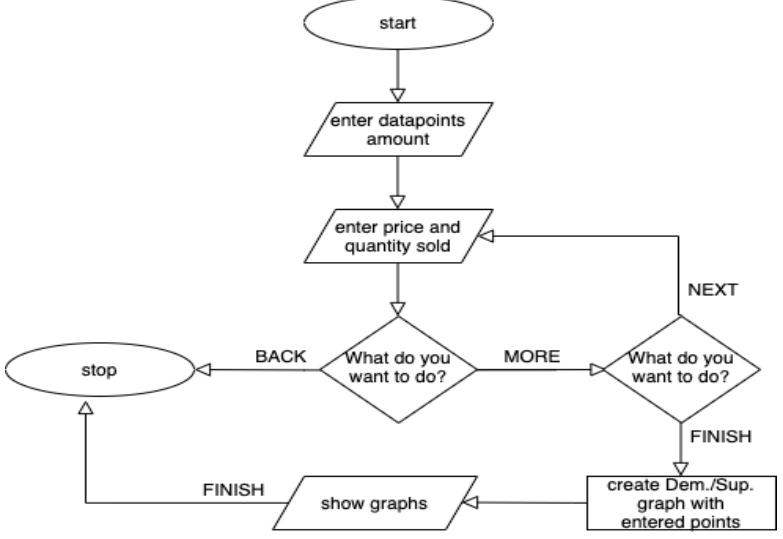
4.3 Editing a product


start
fill boxes with previously entered data
change fields of data you want
What do you want to do?
BACK
stop
FINISH
if all fields are not empty
false
return error
true
if exp date is correct
false
true
set new values for variables of the Product

4.4 General mechanism of the transaction

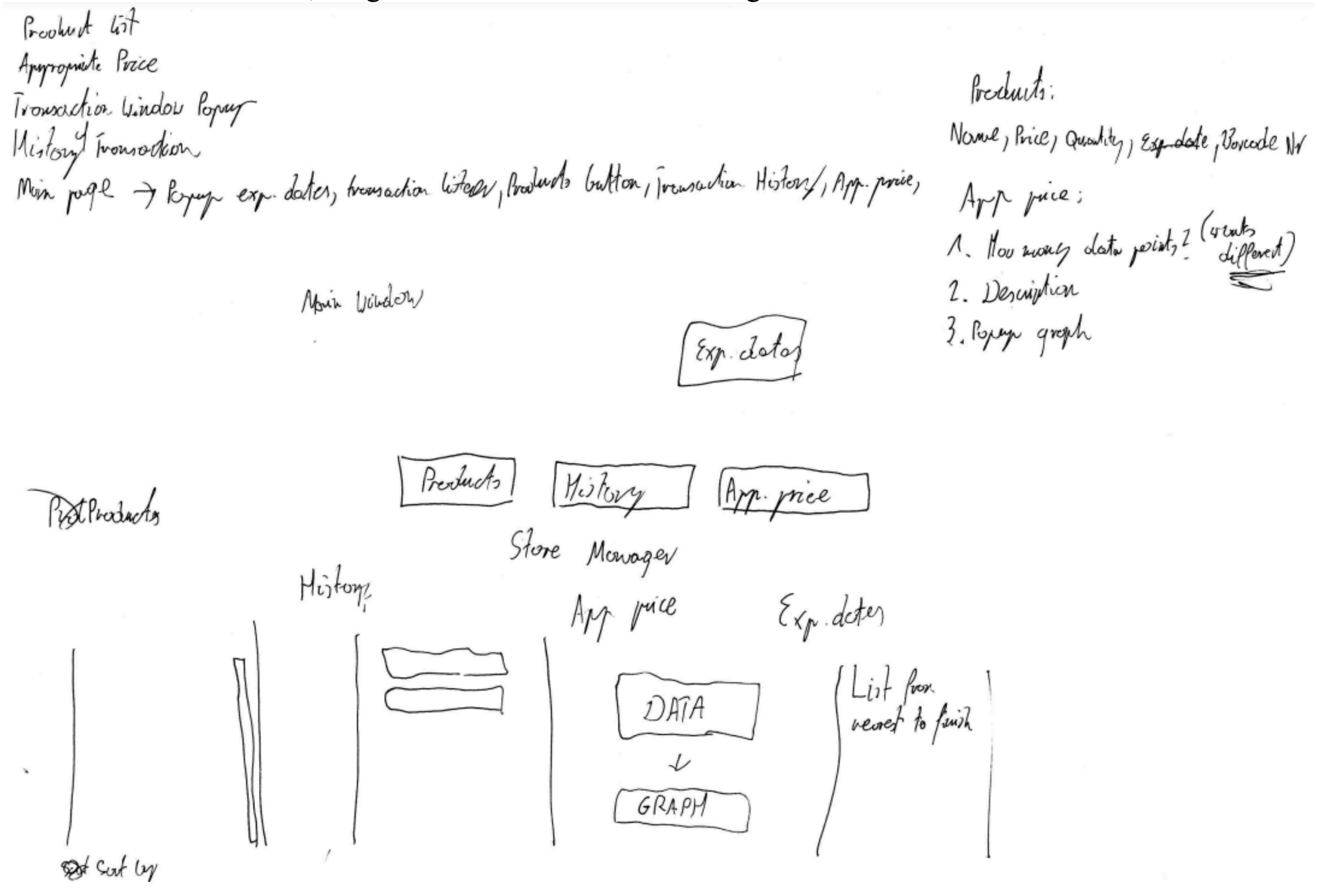

start
create items <TransactionItem> list
add TransactionItem with Product and its quantity (Q), remove Q from Products list
check if Product Q is bigger or equal zero
false
return error
remove one quantity of given Product from TransactionItem list
true
What do you want to do?
CANCEL
stop
MOVE
What do you want to do?
FINISH
PrintBill with total price, tax and list with Products
get money given by the customer
Output change
remove quantity of products bought
stop
EDIT
What do you want to do?
REMOVE
get removed Product
remove given Product from TransactionItem list
EDIT Quantity
get quantity to change of a given Product
if (quantity<=0)
false
if (Product Q - quantity<0)
true
return error
false
change Product quantity from TransactionItem
remove Q of the product from Products
true
change Product quantity from TransactionItem
add Q of the product from Products

4.5 Mechanism of the appropriate price function.



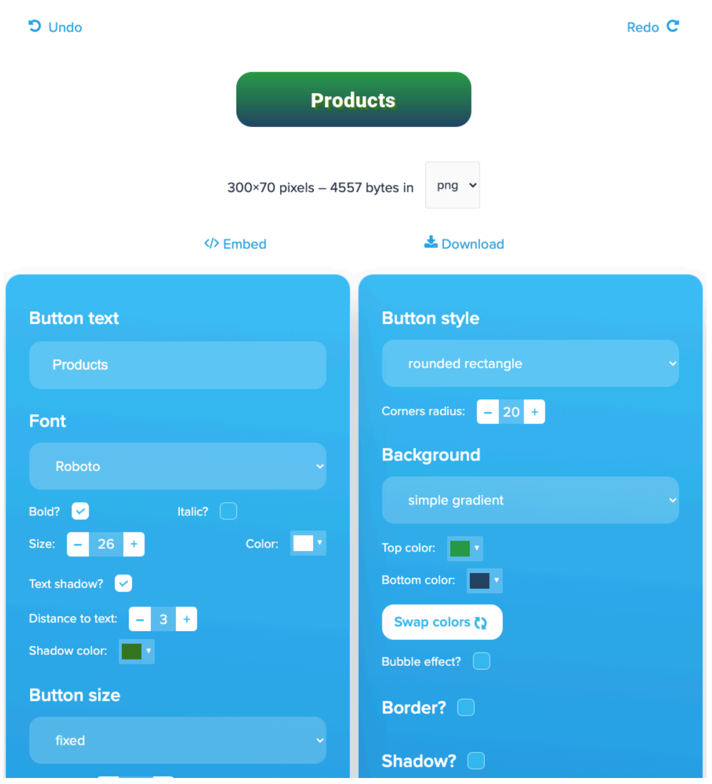
5. GUI

5.1 Personal, rough sketch of ideas while talking with the client



5.2 GUI Windows development in the SceneBuilder software

Designing buttons



Undo

Redo

Products

300×70 pixels – 4557 bytes in png

Embed

Download

Button text

Products

Font

Roboto

Bold? ✓ Italic?

Size: 26 Color:

Text shadow? ✓

Distance to text: 3

Shadow color:

Button size

fixed

Button style

rounded rectangle

Corners radius: 20

Background

simple gradient

Top color:

Bottom color:

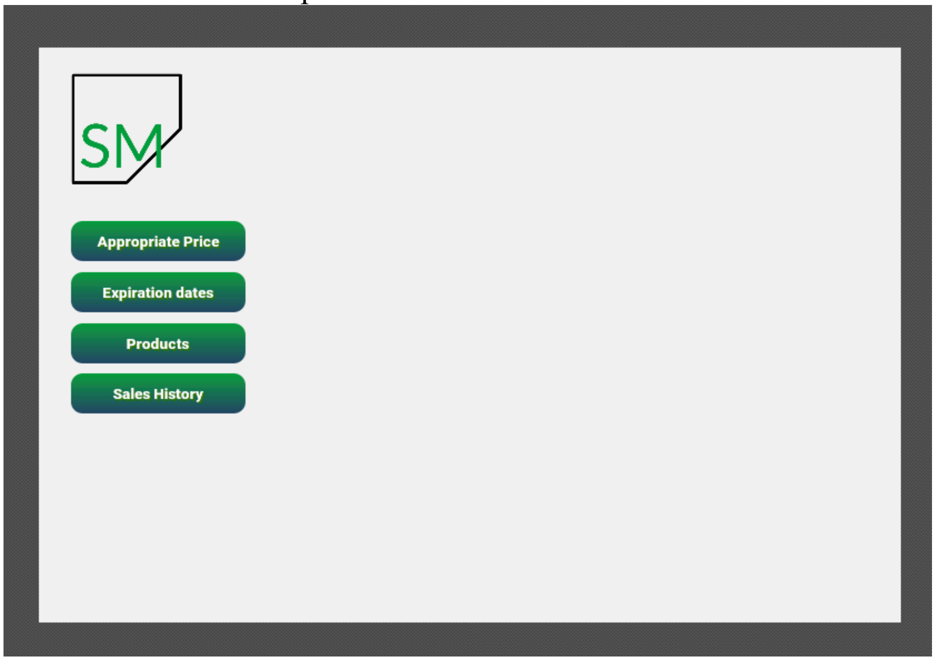
Swap colors

Bubble effect?

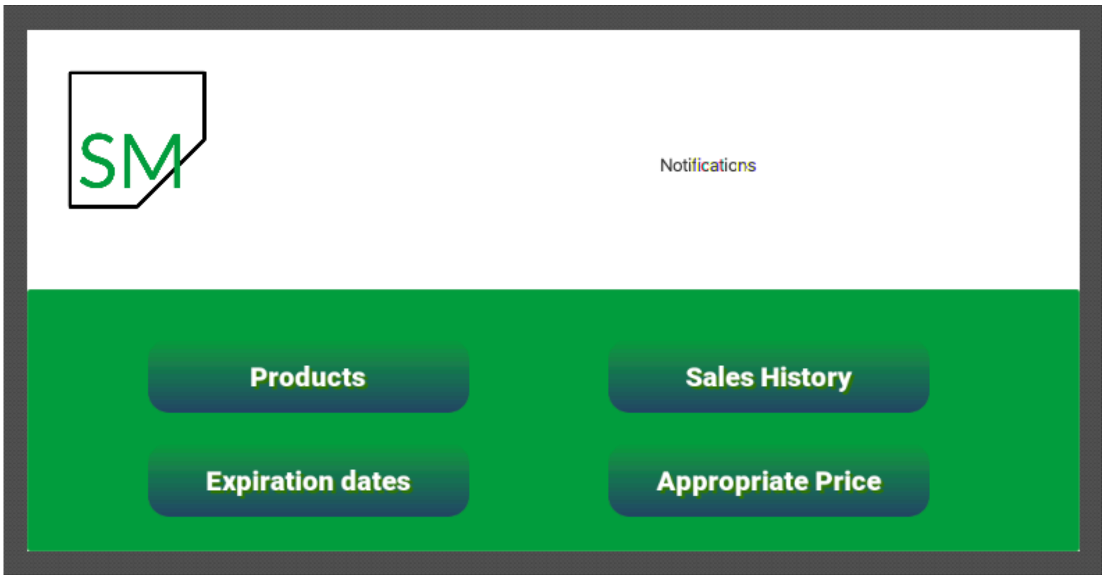
Border?

Shadow?

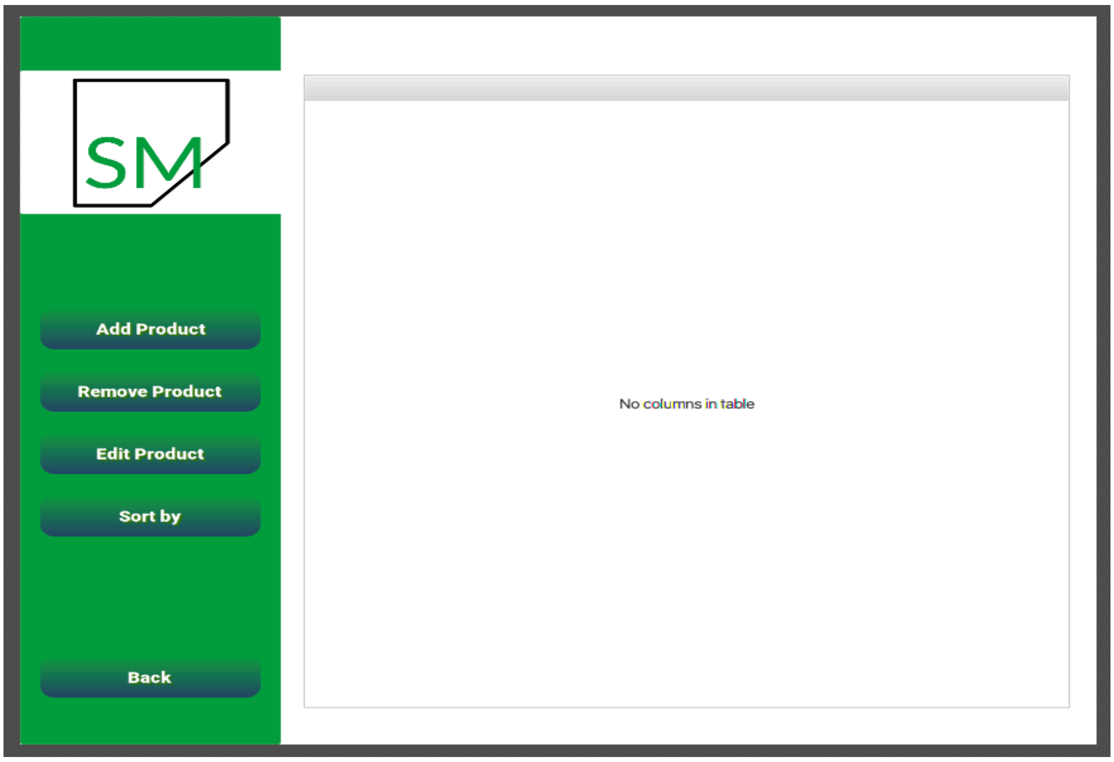
First development of the software's main window



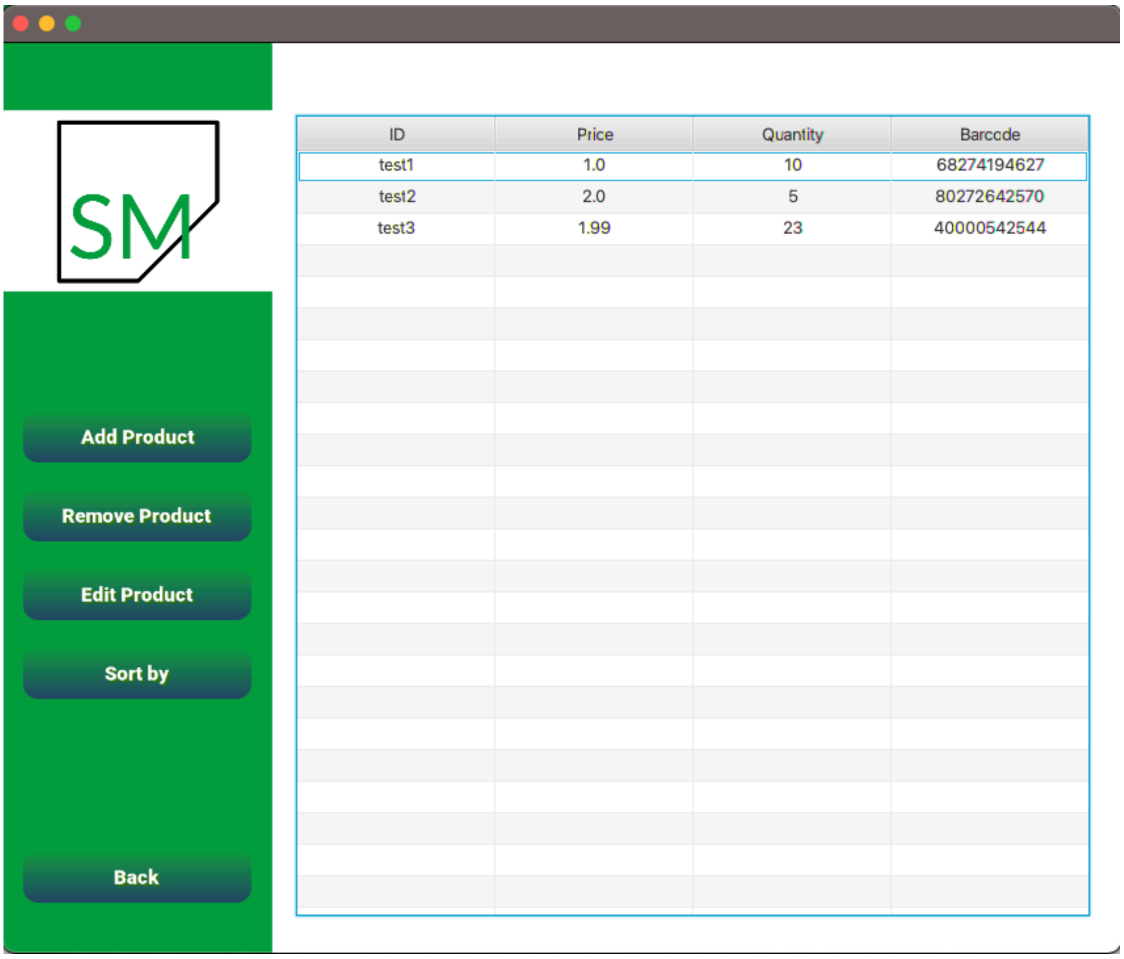
Final version of the software's main window



The products.fxml window to list current commodities and allow to modify them

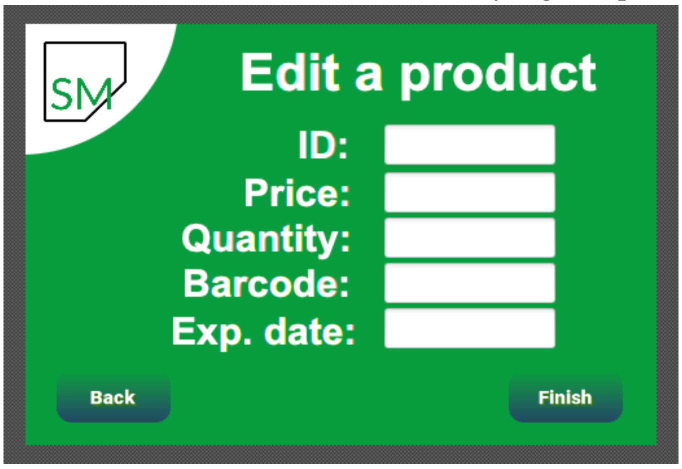


Final version of products.fxml with an implemented list in the table

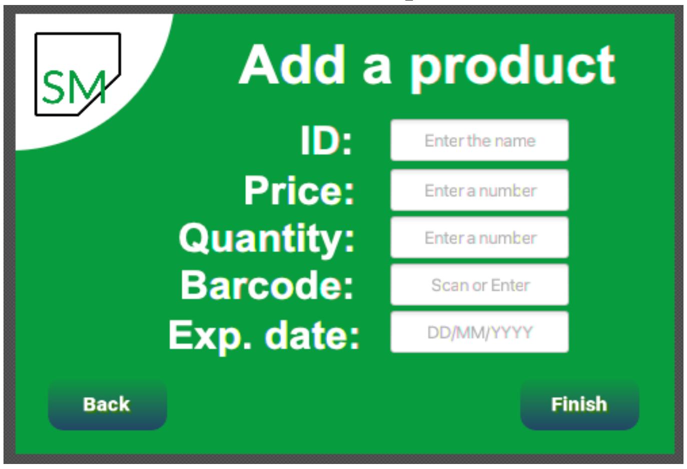


ID	Price	Quantity	Barcode
test1	1.0	10	68274194627
test2	2.0	5	80272642570
test3	1.99	23	40000542544

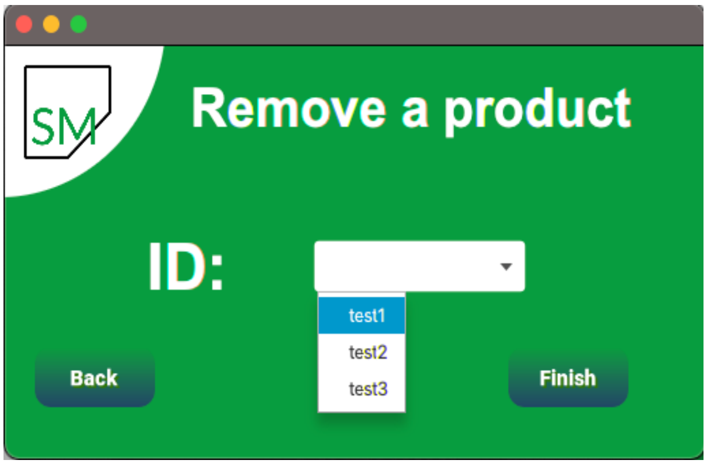
editProduct.fxml window to modify a given product



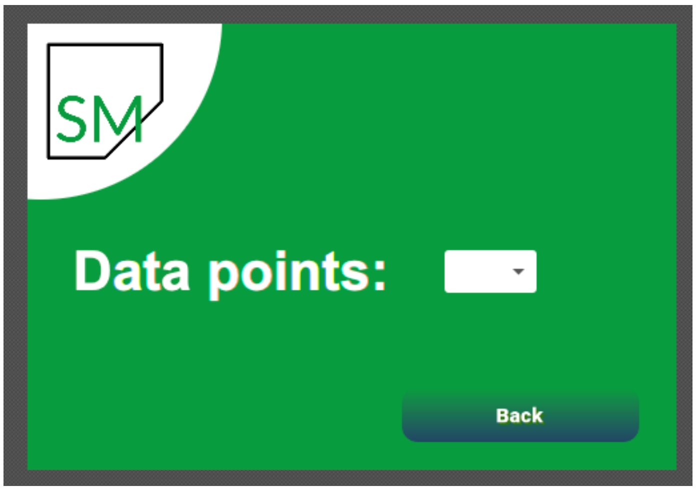
addProduct.fxml to add a product



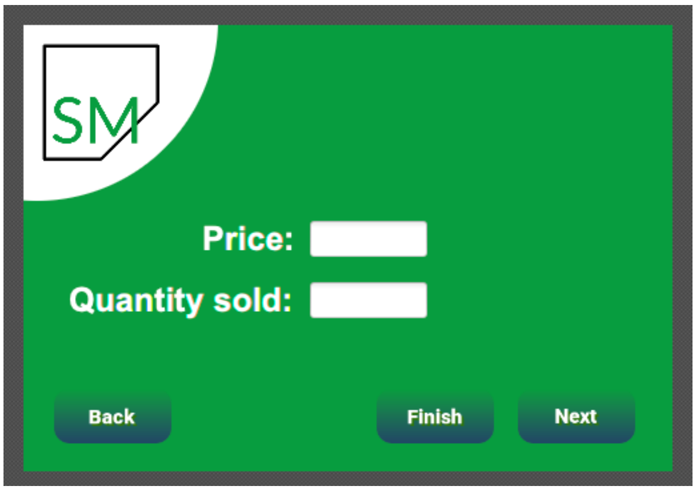
removeProduct.fxml first project



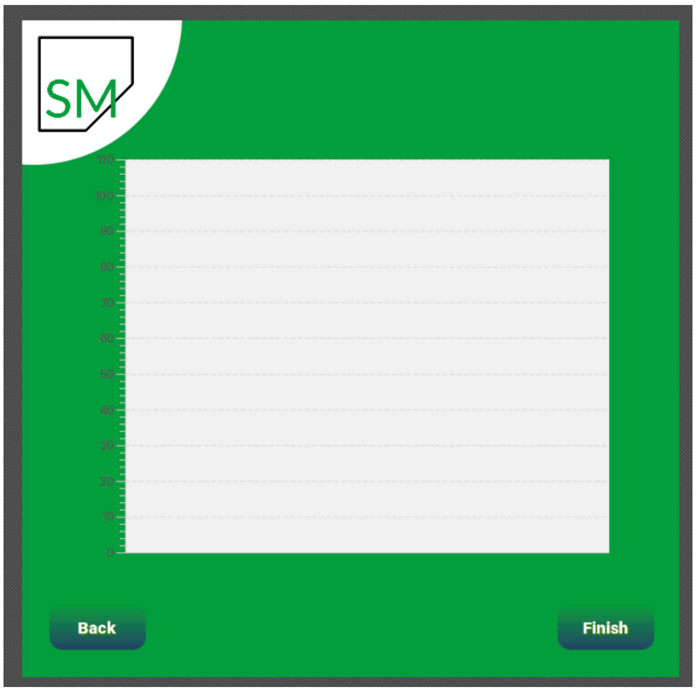
aprice1.fxml to set datapoints number



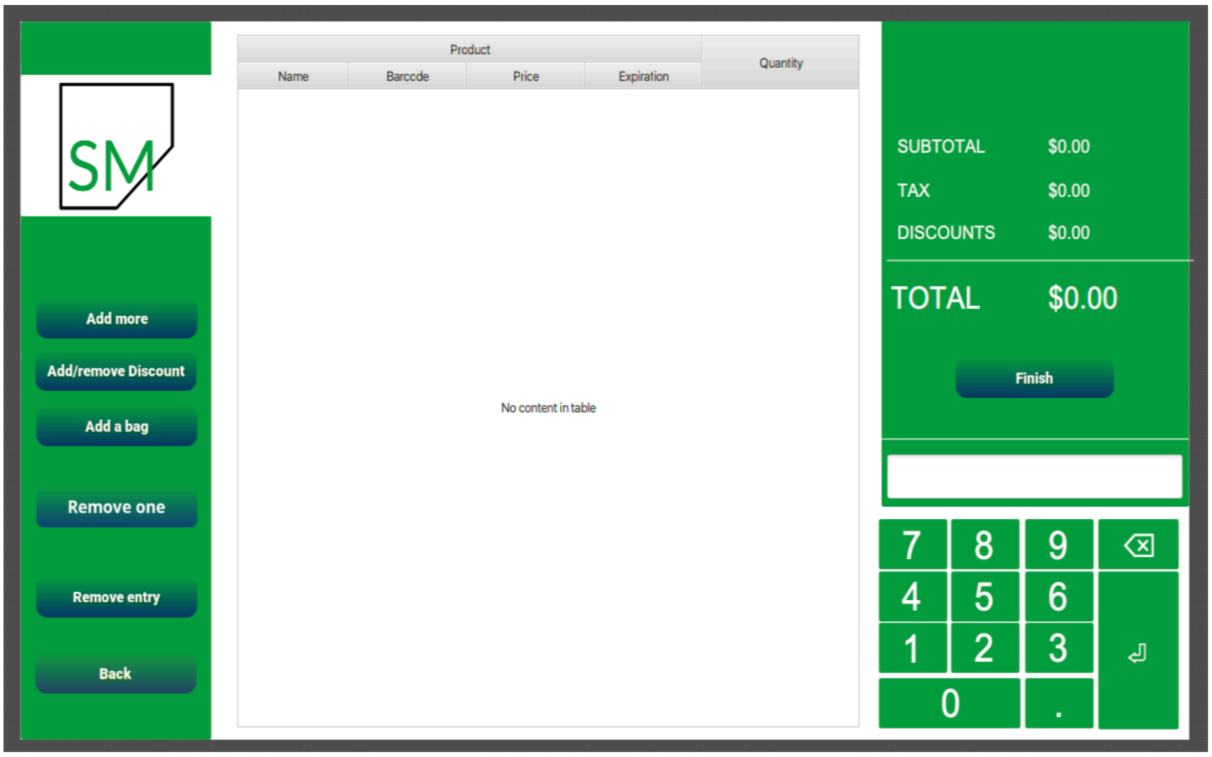
aprice2.fxml to enter points' values



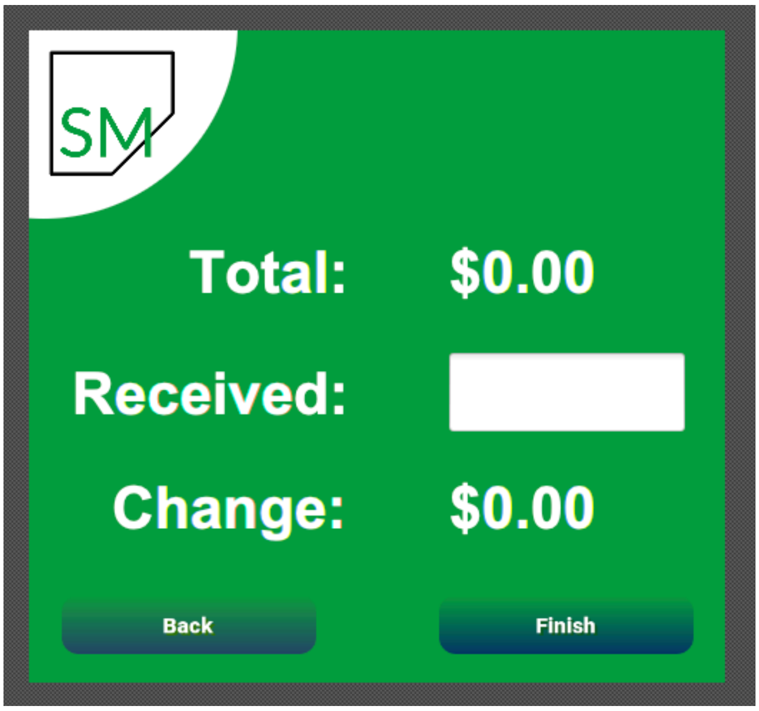
aprice3.fxml displaying a graph based on entered data



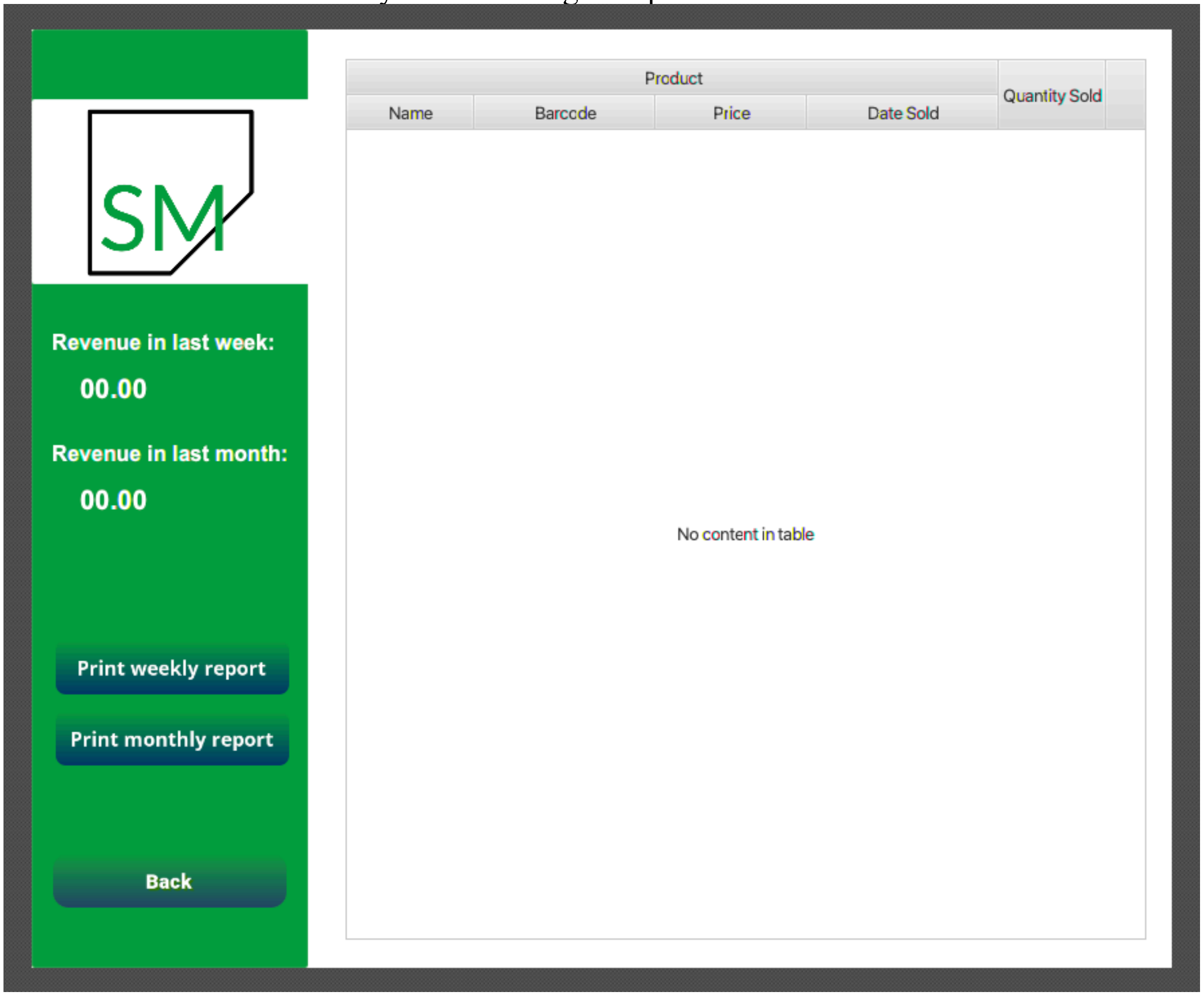
POS.fxml window responsible for point-of-sale



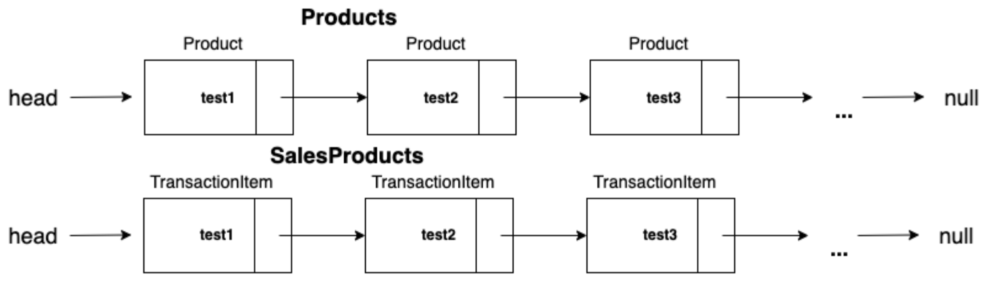
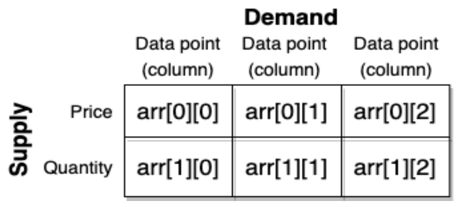
changePOS.fxml showing change to be given



salesHistory.fxml showing sold products and revenues

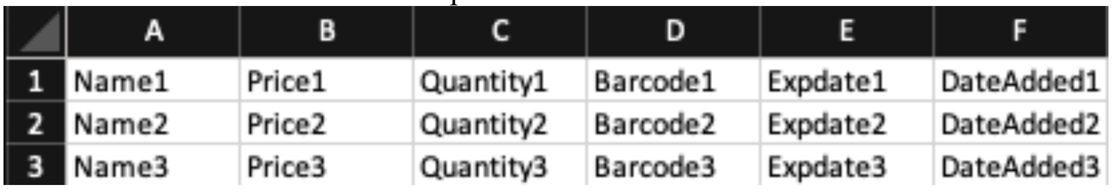


6. Data structures

Type	Function
LinkedList	Used to store Products in one list.  Products Product head test1 test2 test3 ... null SalesProducts TransactionItem
Array 2D	Used to store data points with price and quantity to form a graph based on 2D Array portraying useful information to the client.  Demand Data point (column) Supply Price Quantity arr[0][0] arr[0][1] arr[0][2] arr[1][0] arr[1][1] arr[1][2]

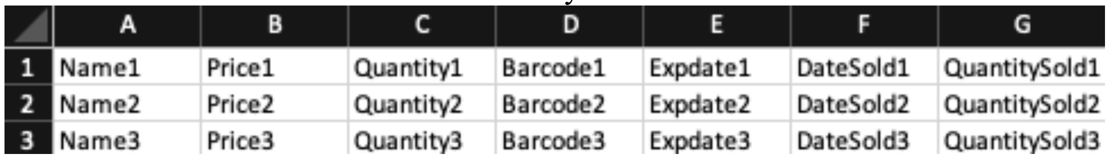
7. Files

products.csv file



	A	B	C	D	E	F
1	Name1	Price1	Quantity1	Barcode1	Expdate1	DateAdded1
2	Name2	Price2	Quantity2	Barcode2	Expdate2	DateAdded2
3	Name3	Price3	Quantity3	Barcode3	Expdate3	DateAdded3

saleshistory.csv file



	A	B	C	D	E	F	G
1	Name1	Price1	Quantity1	Barcode1	Expdate1	DateSold1	QuantitySold1
2	Name2	Price2	Quantity2	Barcode2	Expdate2	DateSold2	QuantitySold2
3	Name3	Price3	Quantity3	Barcode3	Expdate3	DateSold3	QuantitySold3

8. External Libraries

Library	**Use**
JavaFX	Java framework for creating the Graphical User Interface
PDFjet	An open source library for generating and compiling a PDF from a given code input

9. Test plan

Criterion	**Test**
A point-of-sale system linked with a bar scanner, with options to add more or remove a product, an option to use keyboard or GUI numpad to enter a barcode, ability to add and remove a discount, and a calculator with the subtotal, tax, discount, and total price	Check if the point-of-sale system has every required function, and check if the bar scanner works fully with the application Check if the calculator works correctly, and if the numpad responds fast
A point-of-sale change system with a change calculator	Check if the change system gives correct calculations of the change. Check if the Finish button correctly updates the products' quantities
Storing a list of store's commodities	Check if the list can hold every product and display it in a clear, visible manner
Option to add and remove the products and their variables on the list	Check whether the add, remove, and edit buttons change file data appropriately
Displaying sorted list of commodities, together with its name, price, quantity, and expiration date	Check if the client can sort the list by own preference
Changing quantity of the products according to their purchases and supply	Check whether the products file gets updated with sold product. Check whether edit product allows for increasing the quantities of a product
Simple graphical user interface	Use of basic UI creating principles and final interview with the client

Notifying about upcoming expiration dates and low quantities of a good	Check whether a main window notification comes up with an upcoming expiration date and low quantity of a good
Displaying sorted list of the sold products, together with their name, barcode, price, date they were sold, and quantity sold	Check if the products are sorted according to the date they were sold on default, and if the client can sort the list by own preference
A cumulative sales report print file in a given time period	Test of the PDFjet library generator with a products list.
Ability to find the most appropriate price for a product from a given input	Check whether the appropriate price mechanism performs well and produces a readable graph