

PROJECT REPORT

OBJECT-ORIENTED PROGRAMMING AND DATA BASE

2017/2018

Vegetable Distributor

Authors

David Marques - 70940

Manuel Morais - 78713

Supervisor

Prof. Alexandra MOUTINHO

January 22, 2018



Contents

1	Introduction	1
1.1	Project Description	1
1.2	Implementation	1
2	Queries	4
2.1	Select all the data from a table	4
2.1.1	Relational Algebra	4
2.1.2	MySQL	4
2.1.3	Output	4
2.2	Present the items available for purchase	5
2.2.1	Relational Algebra	5
2.2.2	MySQL	5
2.2.3	Output	5
2.3	Search by Name or Category	6
2.3.1	Relational Algebra	6
2.3.2	MySQL	6
2.3.3	Output	6
2.4	Item Price	7
2.4.1	Relational Algebra	7
2.4.2	MySQL	7
2.4.3	Output	7
2.5	Days until expiration	8
2.5.1	Relational Algebra	8
2.5.2	MySQL	8
2.5.3	Output	8
2.6	Last Order Number	9
2.6.1	Relational Algebra	9
2.6.2	MySQL	9
2.6.3	Output	9
2.7	Replace ID's with information	10
2.7.1	Relational Algebra	10
2.7.2	MySQL	10
2.7.3	Output	10
2.8	Items available for sell	11
2.8.1	Relational Algebra	11
2.8.2	MySQL	11
2.8.3	Output	11
2.9	Warehouse Capacity	12
2.9.1	Relational Algebra	12

2.9.2	MySQL	12
2.9.3	Output	12
2.10	Rank Consumers	13
2.10.1	MySQL	13
2.10.2	Output	13
2.11	Ranking the farms	14
2.11.1	MySQL	14
2.11.2	Output	14
3	Interface	15
3.1	Toolbox Components used	15
3.2	Log in Window	16
3.3	Client Window	19
3.4	Producer Window	23
3.5	Administrator Window	25
4	Submitted Files and <i>GitHub</i> Repository	27
5	Conclusion	27
A	Database Generation	28

List of Figures

1	Entity Relationship Diagram	3
2	Clean Log in Window	16
3	User Selection	17
4	Error in Invalid Limitations	17
5	Successful Register	18
6	Client Window	19
7	Available Vegetable Families	19
8	The Client is searching for Pineapples	20
9	Placing and order	20
10	Selecting the desired delivery date	21
11	Error because the minimum 5 working days constraint was not respected	21
12	Error thrown because the client registered in log in window but was not inserted in the database	22
13	Producer Window Clean	23
14	Inserting a new item	24
15	Insert new item completed	24
16	Administrator Window	25
17	Assigning an item to a Warehouse	26
18	Process Completed	26

List of Tables

1	Contents of Table Warehouse	4
2	Items available for Sale	5
3	All the diferent kinds o	6
4	Price per unit of IDPlant 2	7
5	Days to expiration date	8
6	Last Order ID Number	9
7	List of available Plants with the corresponding information . .	10
8	All the tomatoes currently in store	11
9	Warehouse Capacity List	12
10	Top consumers in the data base	13
11	Farms ranked by supplied products	14

1 Introduction

Data volumes are exploding. More data has been created in the past four years than in the entire previous history in the human history¹. For this reason, it is critical to develop methods to store and access the stored data in most efficient way possible, and the usage of data-bases like *MySQL* is one of the best ways to achieve this.

With this project a demonstration of the *MySQL* database framework capabilities will be presented and discussed.

1.1 Project Description

The business created for this project is a fruit and vegetables distributor. The owner of the farm wishes to sell the products to a client. Instead of seeking the client the farm owner sells the products to a distributor, at a slightly lower price. The distributor will then act as a middle man and sell the farm owner products. The items are stored in a warehouse and the client looking to purchase goods will be able to browse the warehouse's contents.

The following entities were defined:

- **Client:** The client will be looking to purchase good from the warehouse and it will access the system by logging in with credential. The client must register first before using the interface;
- **Orders:** An order made by a client. Can contain one or several items;
- **Plant:** Details about a specific vegetable or fruit, including the name, the type and the maximum time that product can remain in storage;
- **Farm:** Information about a particular farmland;
- **Warehouse:** Warehouse description including the address and maximum capacity.

All the databases are fictions and were artificially created uniquely or the purpose of this project. This system could very easily use real data however.

1.2 Implementation

This business was designed using three software programs:

- Visual Studio 2015;

¹Forbes Magazine

- MySQL Workbench;
- Enterprise Architect.

Enterprise Architect was used to design the ER-Model and Class Diagram, and to create the *MySQL* script that generated the database structure. The script was run in *MySQL Workbench* in order to create the database in the cluster server. To the script was added the code that would read the data from *.csv* files and load them into the database. *MySQL Workbench* was used afterwards to design the queries. Finally, *Visual Studio* was used to design the interface that would interact with the system users.

1 INTRODUCTION

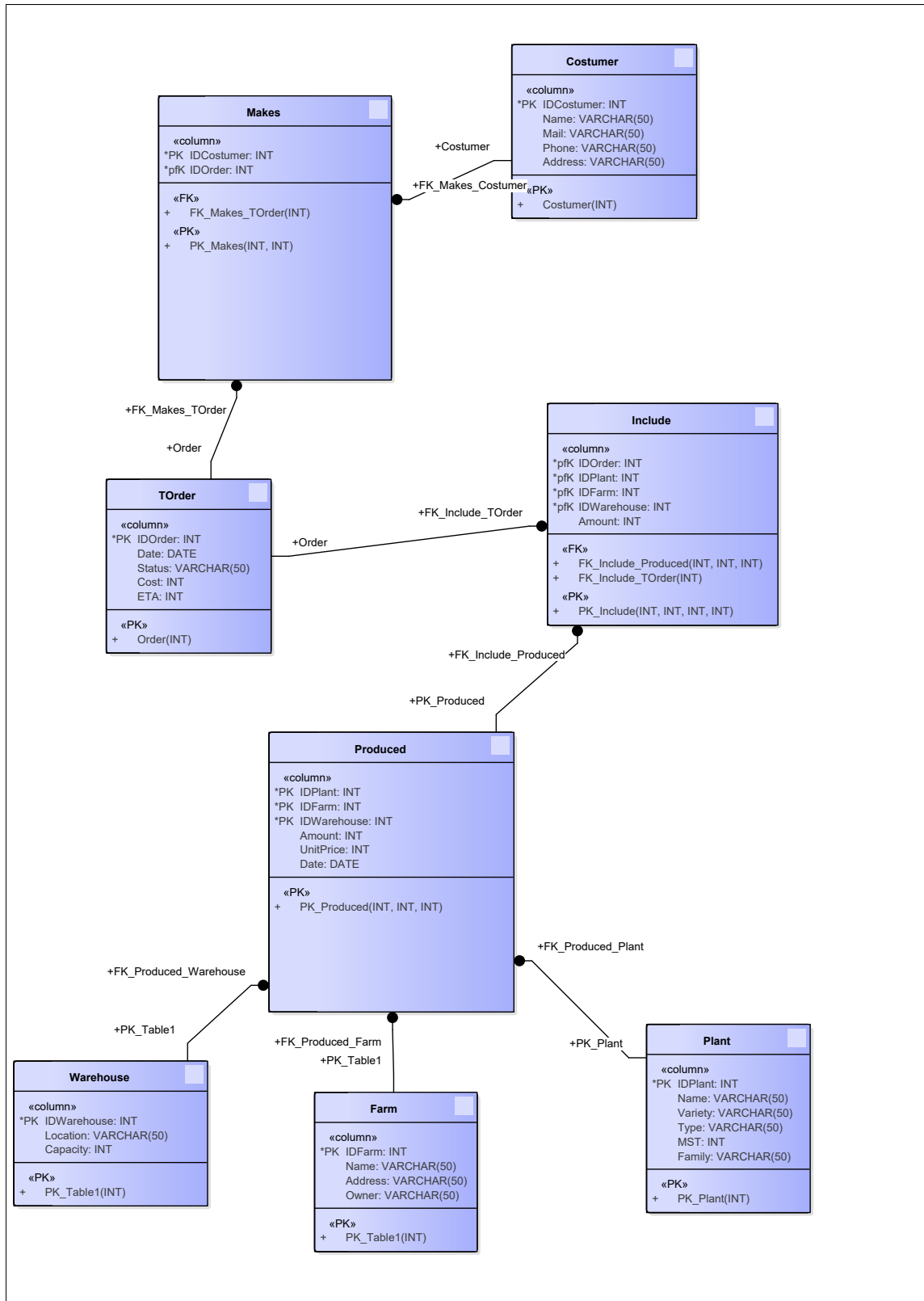


Figure 1: Entity Relationship Diagram

2 Queries

A query is a question, often expressed in a formal way. A database query can be either a select query or an action query. A select query is a data retrieval query, while an action query asks for additional operations on the data, such as insertion, updating or deletion.

In the present section, all the queries used in the project will be presented and explained in detail.

2.1 Select all the data from a table

This query will select the all the data from a table.

2.1.1 Relational Algebra

$$\sigma_{IDWarehouse, Location, Capacity}(Warehouse)$$

2.1.2 MySQL

```
1 select * from Warehouse;
```

2.1.3 Output

ID Warehouse	Location	Capacity
1	Porto	50
2	Algarve	50
3	Lisboa	50

Table 1: Contents of Table Warehouse

2.2 Present the items available for purchase

This query will be used to present to the client the items that are available for purchase.

2.2.1 Relational Algebra

$$\Pi_{Plant.Name}(\sigma(Plant \bowtie Produced))$$

2.2.2 MySQL

```
1 select Plant.Name
2 from Plant
3 Natural Join Produced
4 group by Name
```

2.2.3 Output

Name
Ananas
Lettuce
Potato
Tomato

Table 2: Items available for Sale

2.3 Search by Name or Category

Presents all the items with a specific name or category

2.3.1 Relational Algebra

$$\Pi_{IDPlant, Name, Variety, Type, MST, Family}(\sigma_{Name='Tomato'}(Plant))$$

2.3.2 MySQL

```
1 select * from Plant
2 where Name = 'Tomato'
```

2.3.3 Output

IDPlant	Name	Variety	Type	MST	Family
8	Tomato	Cherry	Bio	30	Fruit
10	Tomato	Cherry	GMO	30	Fruit
12	Tomato	"Black Russian"	GMO	90	Fruit
14	Tomato	"Tini Tiger"	Bio	20	Fruit

Table 3: All the different kinds o

2.4 Item Price

Unit price for a given plant ID

2.4.1 Relational Algebra

$$\Pi_{UnitPrice}(\sigma_{IDPlant=2}(Produced))$$

2.4.2 MySQL

1

2

```
select Unit_Price from Produced
where ID_Plant = 2
```

2.4.3 Output

UnitPrice
7

Table 4: Price per unit of IDPlant 2

2.5 Days until expiration

The number of days until the item surpasses the expiry date. Information useful for the client. This value is acquired by comparing the current date with the number of maximum days an item can remain in storage.

2.5.1 Relational Algebra

$$\Pi_{Plant.MST - datediff(current_date(), Produced.Date)}(\sigma_{IDPlant=1}(Plant \bowtie Produced))$$

2.5.2 MySQL

```
1 select (pp.MST - datediff(current_date(), p.Date))
2 from Produced as p, Plant as pp
3 where p.ID_Plant=1
4 and p.ID_Plant=pp.ID_Plant
```

2.5.3 Output

"Days to Expiration"
11

Table 5: Days to expiration date

2.6 Last Order Number

Return the last order ID number.

2.6.1 Relational Algebra

$$\Pi_{MAX(ID_{Order})}(TOrder)$$

2.6.2 MySQL

```
1 select MAX(ID_Order) from TOrder;
```

2.6.3 Output

MAX(IDOrder)
3

Table 6: Last Order ID Number

2.7 Replace ID's with information

Replaces the ID's of the Plants, Farms and Warehouses with the corresponding names and additional information that will be presented in the administrator menu.

2.7.1 Relational Algebra

$$\Pi_{*1}(Warehouse \bowtie (Farm \bowtie (Plant \bowtie Produced)))$$

*1 Plant.Name, Farm.Name, Warehouse.Location, Produced.Amount, Produced.UnitePrice, Produced.Date

2.7.2 MySQL

```
1 select * from
2 (
3     select p.Name as 'Plant', f.Name as 'Farm', w.Location as
4         ↳ 'Warehouse', pp.Amount as 'Amount', pp.UnitPrice as
5         ↳ 'Unit Price', pp.Date as 'Entry Date'
6     from Plant as p, Farm as f, Warehouse as w, Produced as
7         ↳ pp
8     where p.IDPlant = pp.IDPlant
9     and f.IDFarm = pp.IDFarm
10    and w.IDWarehouse = pp.IDWarehouse) as R
11 where Warehouse != 'New'
```

2.7.3 Output

Plant	Farm	Warehouse	Amount	"Unit Price"	"Entry Date"
Tomato	WesternComboys	" Lisboa"	30	3	2018-01-01
Potato	WesternComboys	" Porto"	5	7	2016-12-06
Tomato	WesternComboys	" Porto"	5	7	2016-12-06

Table 7: List of available Plants with the corresponding information

2.8 Items available for sell

In this example, the query will list all the different tomatoes available for purchase.

2.8.1 Relational Algebra

$$\Pi_{*1}(\sigma_{Plant.Name=Tomato}(Produced \bowtie Plant))$$

*1 Plant.IDPlant, Plant.Name, Plant.Variety, Plant.Type, Plat.MST, Plant.Family

2.8.2 MySQL

```
1 select p.IDPlant as 'id', p.Name, p.Variety, p.Type, p.MST,  
   ↪ p.Family  
2 from Plant as p, Produced as pp  
3 where p.Name = 'Tomato'  
4 and p.IDPlant = pp.IDPlant
```

2.8.3 Output

id	Name	Variety	Type	MST	Family
10	Tomato	Cherry	GMO	30	Fruit
12	Tomato	"Black Russian"	GMO	90	Fruit
8	Tomato	Cherry	Bio	30	Fruit
14	Tomato	"Tini TIger"	Bio	20	Fruit

Table 8: All the tomatoes currently in store

2.9 Warehouse Capacity

Gets the capacity for all the warehouses for the administrator.

2.9.1 Relational Algebra

$$s \leftarrow IDWarehouse, g_{sum}(Amount) \text{ as Stored } (Produced)$$

$$\Pi_{*1}(Warehouse \bowtie s)$$

*1 Warehouse.IDWarehouse, Warehouse.Location,s.Stored, Warehouse.Capacity

2.9.2 MySQL

```
1 select s.IDWarehouse,
   ↪ w.Location,concat(s.Stored/w.Capacity*100,'%') as
   ↪ 'Storage', w.Capacity
2 from (
3     select pp.IDWarehouse, sum(pp.Amount) as 'Stored'
4     from Produced as pp
5     group by pp.IDWarehouse) as s, Warehouse as w
6 where s.IDWarehouse = w.IDWarehouse
```

2.9.3 Output

IDWarehouse	Location	Stored	Capacity
1	" Porto"	10	50
2	" Algarve"	5	50
3	" Lisboa"	30	50

Table 9: Warehouse Capacity List

2.10 Rank Consumers

Lists the consumers, with the respective contact information, by number of orders. This query will help identify the clients that most use the service.

2.10.1 MySQL

```
1  select c.IDCostumer, c.Name as 'Name', r.Consumption,  
   ↪   c.Phone,c.Mail  
2  from (  
3  select w.IDCostumer, s.IDOrder ,s.Cost as 'Consumption'  
4  from TOrder as s  
5  natural join Makes as w  
6  where s.IDOrder = w.IDOrder  
7  group by w.IDCostumer) as r, Costumer as c  
8  where r.IDCostumer = c.IDCostumer  
9  group by r.Consumption  
10 ORDER BY r.consumption desc;
```

2.10.2 Output

IDCostumer	Name	Consumption	Phone	Mail
4	Ruben	80	934562324	rubentecnico@gmail.com
1	Manel	60	91645463324	pingodoce@gmail.com
6	Jose	50	943262324	joseguerra@gmail.com

Table 10: Top consumers in the data base

2.11 Ranking the farms

This query ranks the farms in order of supplied the products.

2.11.1 MySQL

```
1 select f.IDFarm, f.Name as 'Name', sum(p.Amount) as  
   ↪ 'Supplied', f.Owner ,f.Address, p.Date  
2 from Produced as p , Farm as f  
3 where f.IDFarm = p.IDFarm  
4 group by f.IDFarm  
5 order by Supplied desc;
```

2.11.2 Output

IDFarm	Name	Supplied	Owner	Address	Date
2	"Quinta Jose "	58	David	" Algarve"	2018-01-21
1	"Herdade Vale da Rosa"	44	" Manel"	Alentego	2018-01-21
3	WesternComboys	22	Duarte	" Texas"	2018-01-21
4	"Ze Wars"	9	José	" Coucenhira"	2018-01-22

Table 11: Farms ranked by supplied products

3 Interface

To access the interface the user must first log in with valid credentials. The interface is divided in 4 windows, each with a different function:

- Log in Window
- Client Window
- Producer Window
- Administrator Window

The credentials are not stored in the database but rather in a *.txt* file. Storing a log in credentials in plain text is terrible in terms of security but this aspect is not a part of the projects objectives, these concerns were set aside. The decision was made as an excuse to explore and learn how to write and read into files.

3.1 Toolbox Components used

A lot of different toolbox components were used in this part of the work to make sure that the user would easily find his way through the database and find what he was looking for. This included:

- Different Dialog Windows
- Buttons
- Check Boxes
- List Boxes
- Group Boxes
- Radio Buttons
- Static Text
- List Control
- On Change Event Handlers
- Date Picker
- Warning Dialog

Some of this skills developed took some time to

3.2 Log in Window

The log in window appears when the program is run.

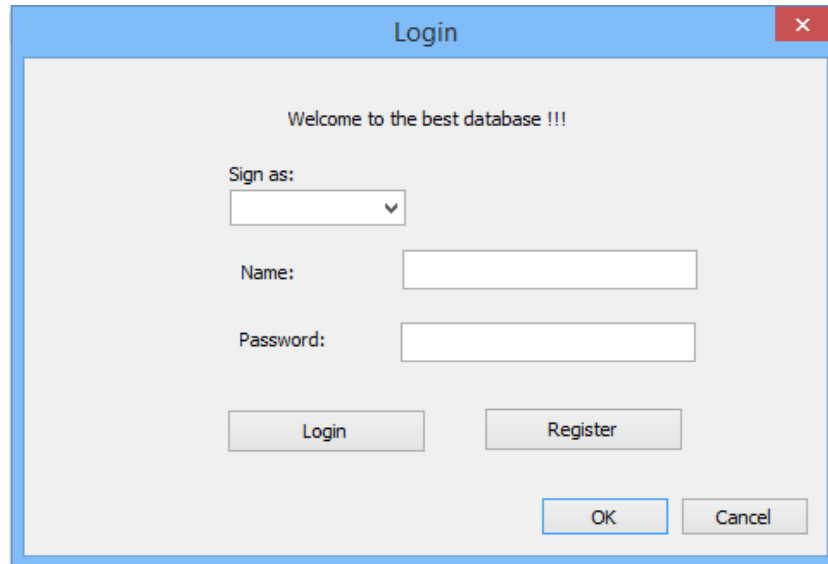


Figure 2: Clean Log in Window

First, the user should select the type of log in from Client, Producer or Administrator and afterwards insert the credentials in the appropriate boxes. If the credentials are not valid, an error is shown.

The user can register as *Client* and *Producer* by first introducing the credentials and then pressing *Register*. This will save the credentials in the text file and will allow the user to browse the application only. To place orders the user must be manually introduced in the database.

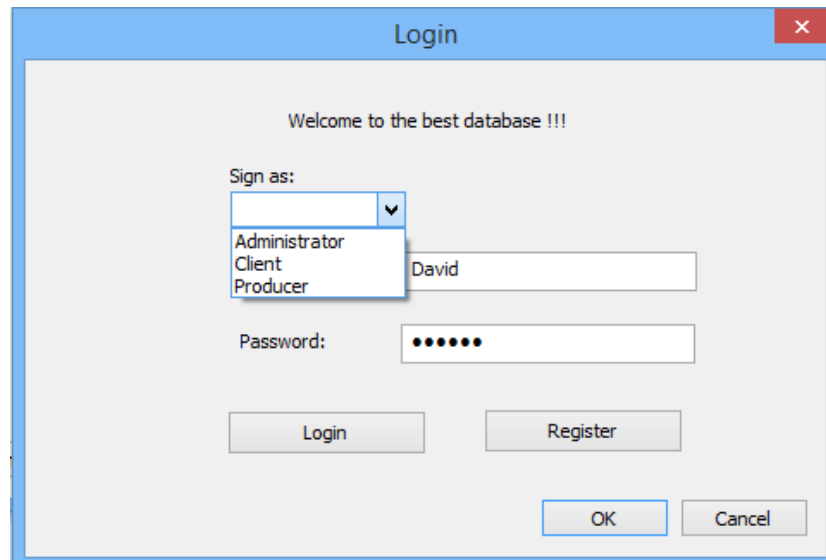


Figure 3: User Selection

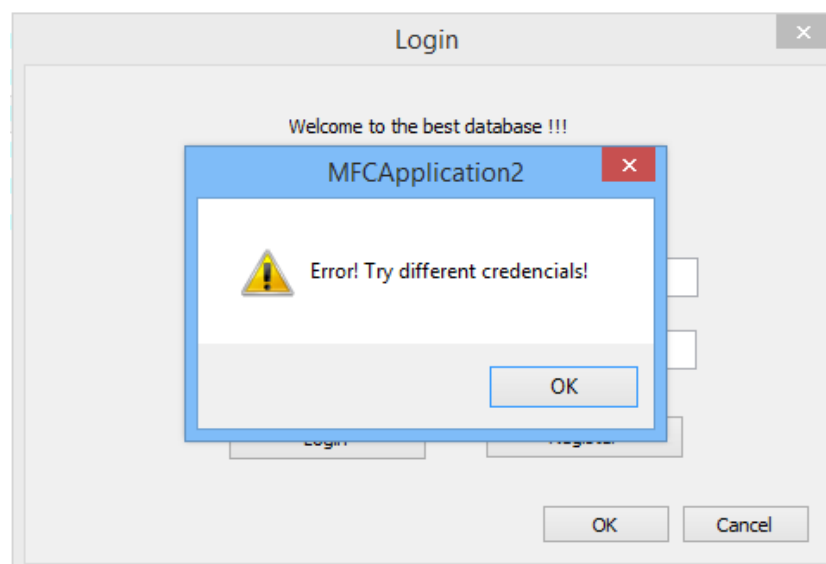


Figure 4: Error in Invalid Limitations

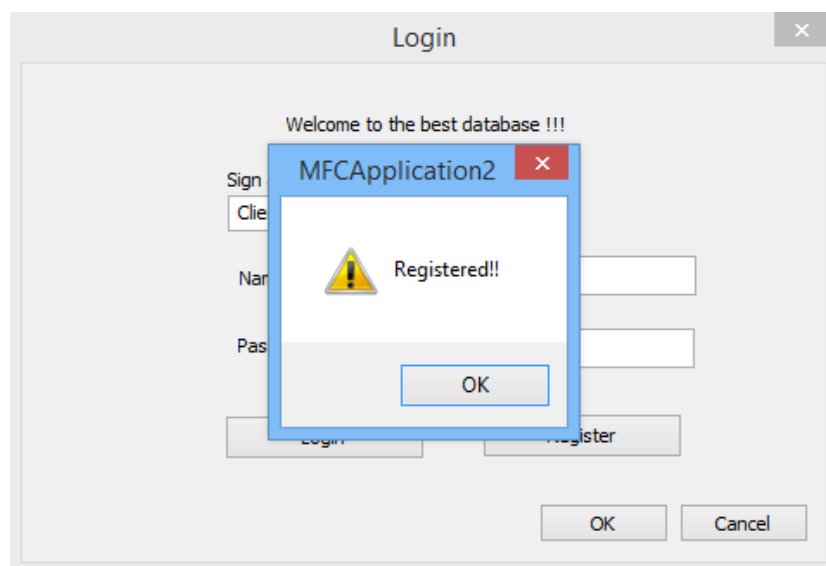


Figure 5: Successful Register

3 INTERFACE

3.3 Client Window

In the Client Window, the user can browse the contents available for purchase and place orders.

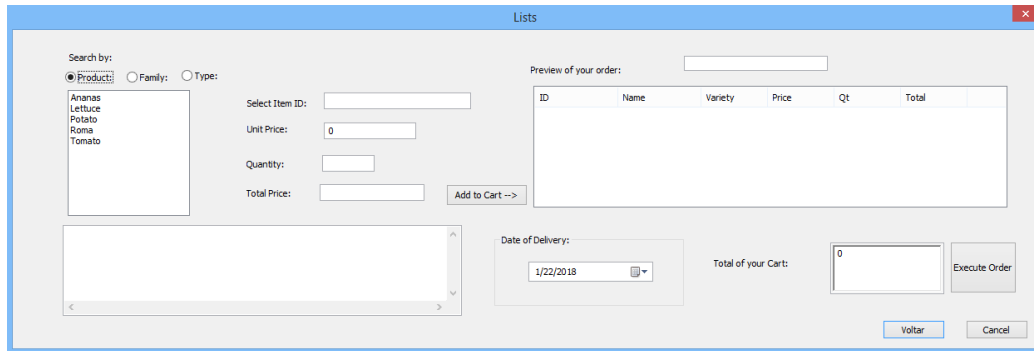


Figure 6: Client Window

In this Window the Client can select the desired items by searching for a specific type, family or name of the vegetable.

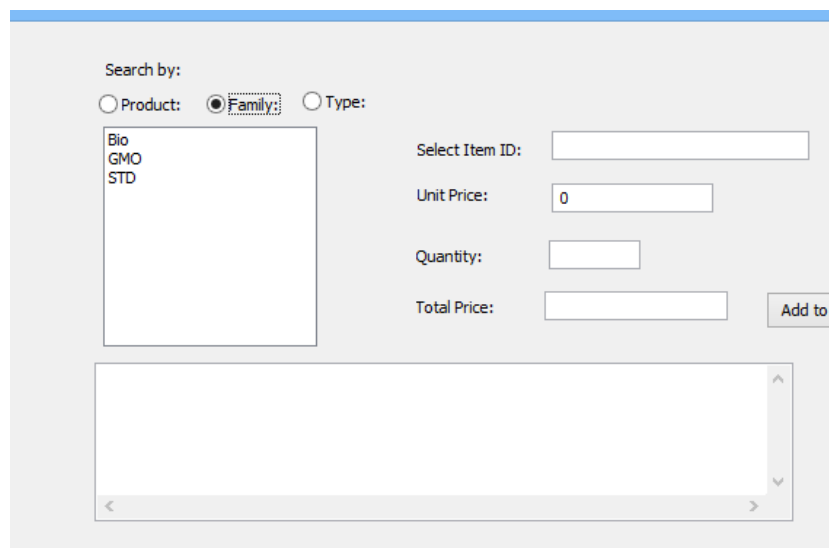


Figure 7: Available Vegetable Families

After selecting the desired Family, Type or Product the Client can see the available products for purchase in the lower window.

To place an order, it is necessary to insert the desired item ID in the box, as well as choosing the desired amount. The Price is inserted automatically. After the selection of the item it is necessary to click the *Add to Cart* button.

3 INTERFACE

Search by:
☒ Product: ☐ Family: ☐ Type:

Ananas
Lettuce
Potato
Roma
Tomato

Select Item ID:
Unit Price:
Quantity:
Total Price:

ID	Name	Variety	Type	MST	Far
2	Ananas	Amazon	Bio	70	Fruit
4	Ananas	Camp	Bio	50	Fruit
6	Ananas	Camp	GMO	50	Fruit

Figure 8: The Client is searching for Pineapples

Several items can be placed in the cart and the total price will be showed in the bottom right corner window.

Search by:
☒ Product: ☐ Family: ☐ Type:

Ananas
Lettuce
Potato
Roma
Tomato

Select Item ID:
Unit Price:
Quantity:
Total Price:

ID	Name	Variety	Type	MST	Far
22	Potato	Sweet	STD	30	Vegi
24	Potato	Amorosa	Bio	12	Vegi
24	Potato	Amorosa	Bio	12	Vegi
26	Potato	Arizona	STD	50	Vegi
26	Potato	Arizona	STD	50	Vegi

Figure 9: Placing and order

After all the items are inserted into the basked, the client must choose

3 INTERFACE

the desired date for delivery. This can be done using the calendar imbued in the window. There must be a minimum of 5 working days between the order time and the delivery time, as this is the time required for delivery. If this condition is not fulfilled and error is shown.

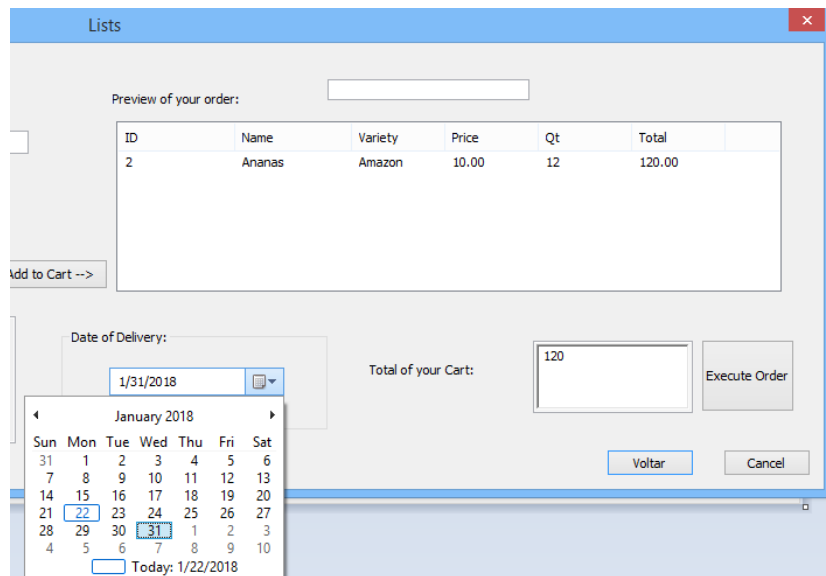


Figure 10: Selecting the desired delivery date

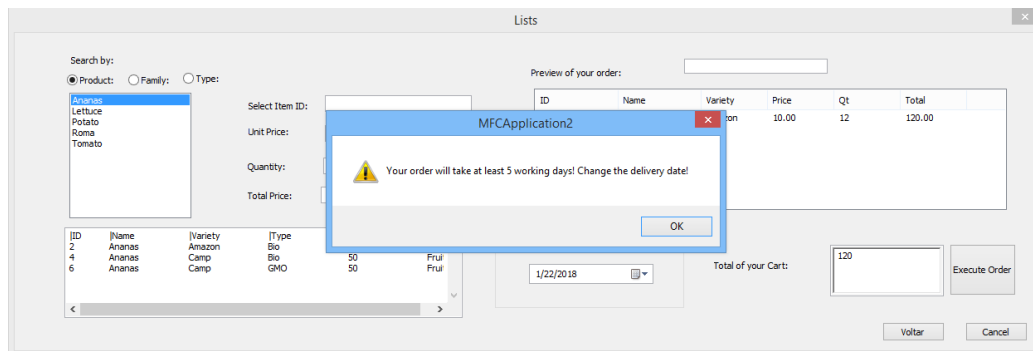


Figure 11: Error because the minimum 5 working days constraint was not respected

If the user registered via the log in window, the newly registered credentials can be used to browse the contents of the store. However, if the client tries to place an order and error will be thrown, as that client will need to be inserted manually in the database.

3 INTERFACE

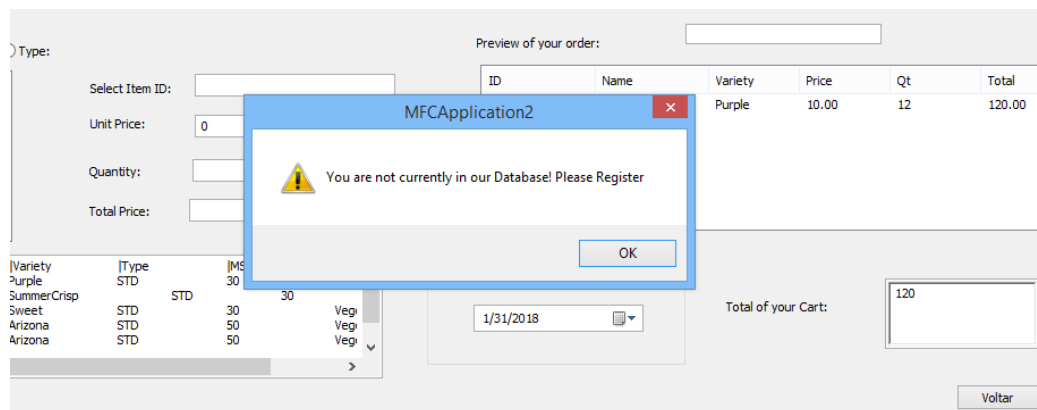
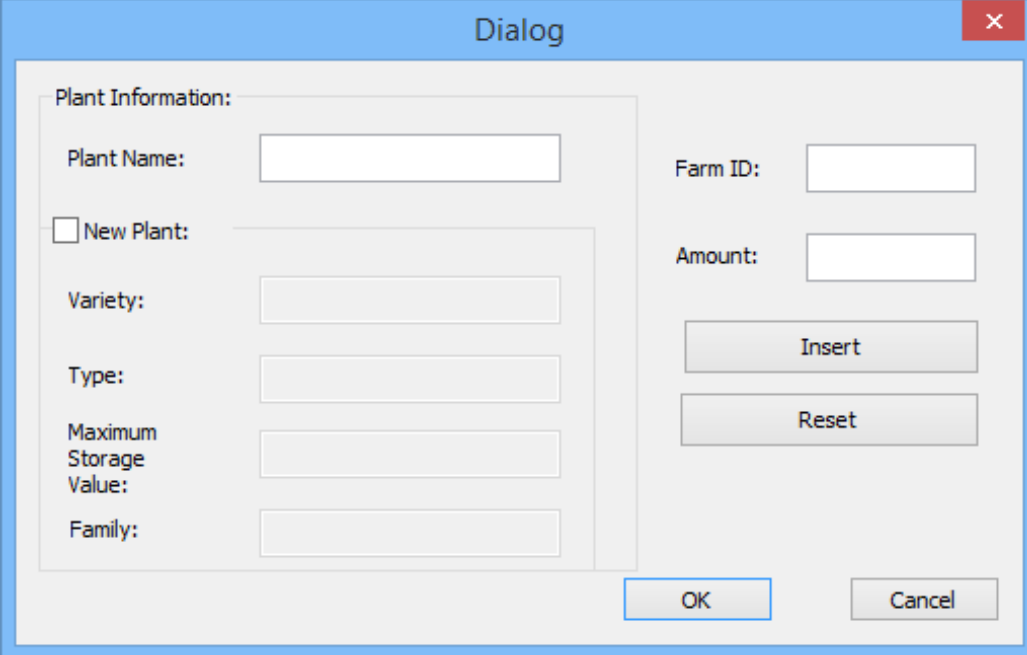


Figure 12: Error thrown because the client registered in log in window but was not inserted in the database

3.4 Producer Window

In the Producer window, new plants can be added in the database.



The image shows a software dialog box titled "Dialog" with a standard Windows-style title bar (blue background, red close button). The dialog contains a form for adding a new plant. On the left, under the heading "Plant Information:", there is a "Plant Name:" label followed by a text input field. Below this is a checkbox labeled "New Plant:". To the right of the checkbox is a section containing five labels with corresponding text input fields: "Variety:", "Type:", "Maximum Storage Value:", and "Family:". To the right of the "Plant Information:" section, there are two more labels with text input fields: "Farm ID:" and "Amount:". Below these fields are two buttons: "Insert" and "Reset". At the bottom right of the dialog are two buttons: "OK" and "Cancel".

Figure 13: Producer Window Clean

To insert a new item, the producer must check the new item box and then specify all the fields to describe the item. Finally, the Farm ID that represents the farm where the item was grown should be inserted as well as the amount.

After all the fields in the window are filled, the Producer should click the insert button to insert the new item in the database. No confirmation window is shown.

Dialog

Plant Information:

Plant Name: Bananas

☒ New Plant:

Variety: Madeira

Type: Bio

Maximum Storage Value: 60

Family: Fruit

Farm ID:

Amount:

Insert

Reset

OK Cancel

Figure 14: Inserting a new item

Dialog

Plant Information:

Plant Name: Bananas

☒ New Plant:

Variety: Madeira

Type: Bio

Maximum Storage Value: 60

Family: Fruit

Farm ID: 1

Amount: 12

Insert

Reset

OK Cancel

Figure 15: Insert new item completed

3.5 Administrator Window

In the administrator window, the newly added items by the Producer are listed in the top window and the status of each warehouse is displayed in the bottom window:

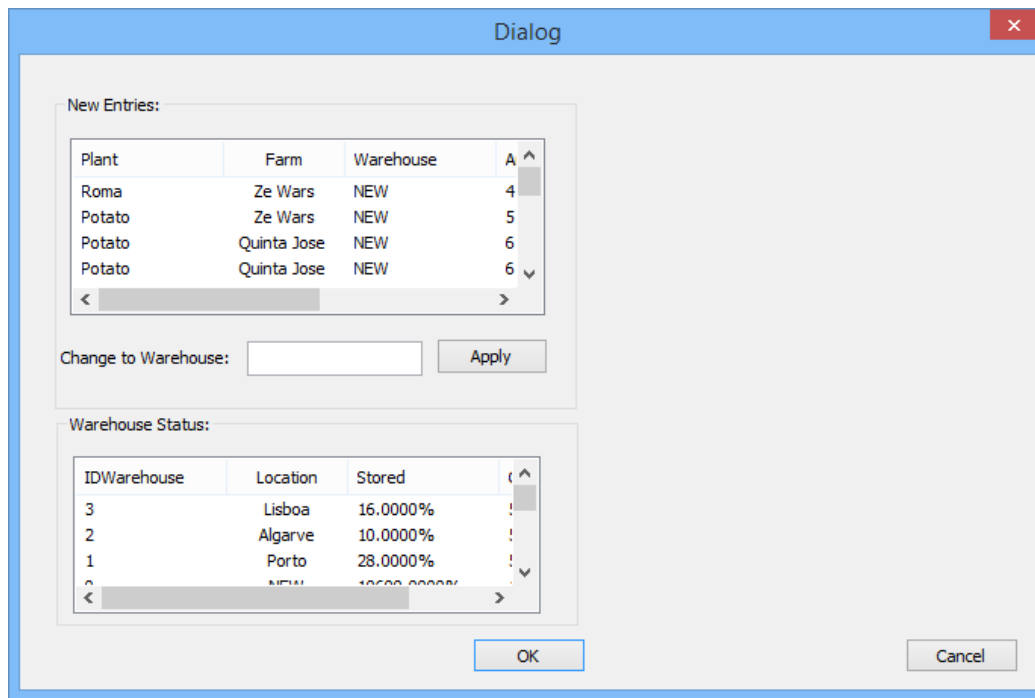


Figure 16: Administrator Window

To assign a product to a Warehouse the Administrator should insert the desired Warehouse ID Number in the appropriate window, then select the item to be assigned and then click the Change Warehouse button.

3 INTERFACE

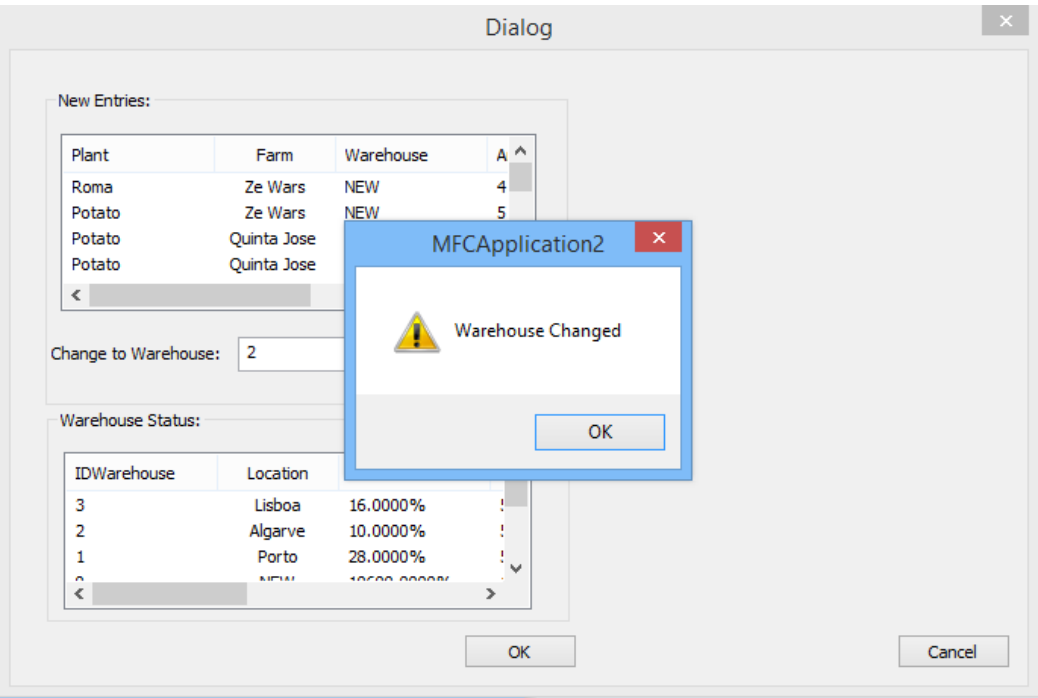


Figure 17: Assigning an item to a Warehouse

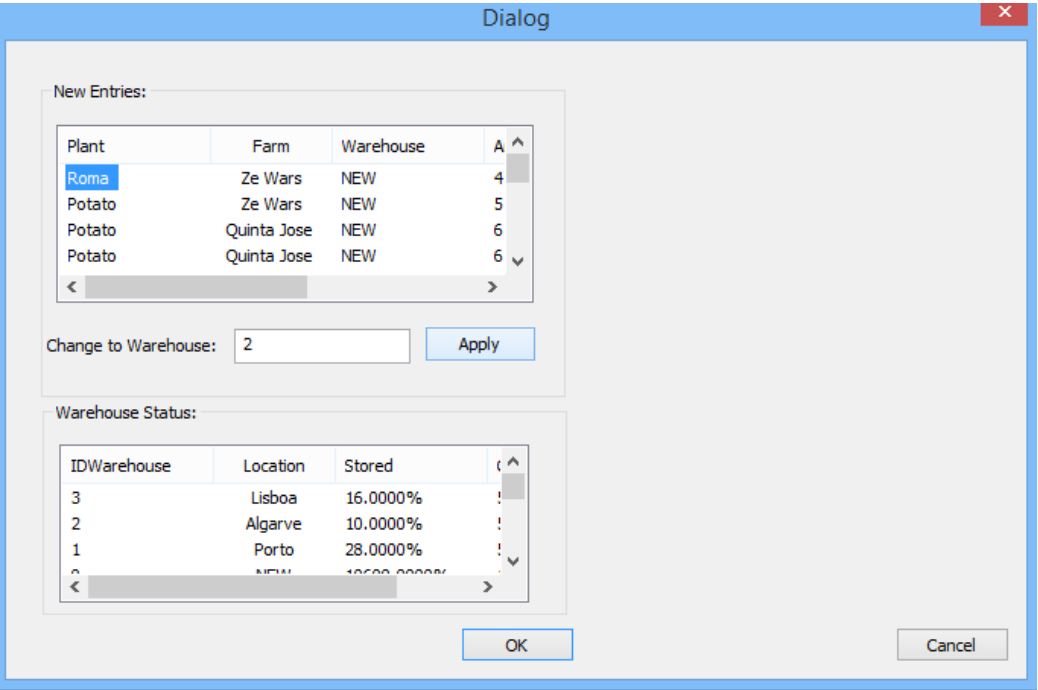


Figure 18: Process Completed

4 Submitted Files and *GitHub* Repository

The project was developed using *git* for version control. The folder submitted by e-mail only contains the release version and the report, as the size limit made it impossible to send the entire project by email. The totality of the project files can be downloaded in the *GitHub* repository page at:

GITHUB REPO LINK

5 Conclusion

By developing this project, the basic knowledge for working and designing databases as well as designing an interface to interact it was acquired, and for this reason the project is considered a success.

Designing and implementing the *queries* was straightforward as the language is fairly intuitive. This process was further simplified by writing the *query* in relational algebra beforehand.

The most challenging part of the project was by far the interface design. Using the *MFC* to design the dialog boxes was not always a smooth process as some toolbox components were somewhat hard to implement. The query-interface interaction was hard to implement at times.

Overall, the project provided the opportunity to learn about a very important topic and to use tools that will definitely be very useful in the future.

A Database Generation

The following script was used to generate the database structure. This script was generated by *Enterprise Architect*

```
1  SET FOREIGN_KEY_CHECKS=0 ;
2
3  /* Create Tables */
4
5  CREATE TABLE `Warehouse` (
6      `IDWarehouse` INT NOT NULL ,
7      `Location` VARCHAR(50) ,
8      `Capacity` INT NOT NULL ,
9      CONSTRAINT `PK_Table1` PRIMARY KEY (`IDWarehouse`));
10
11  CREATE TABLE `TOrder` (
12      `IDOrder` INT NOT NULL auto_increment,
13      `Date` DATE ,
14      `Status` VARCHAR(50) ,
15      `Cost` INT ,
16      `ETA` INT ,
17      CONSTRAINT `Order` PRIMARY KEY (`IDOrder`));
18
19  CREATE TABLE `Produced` (
20      `IDPlant` INT NOT NULL,
21      `IDFarm` INT NOT NULL,
22      `IDWarehouse` INT NOT NULL,
23      `Amount` INT ,
24      `UnitPrice` INT ,
25      `Date` DATE ,
26      CONSTRAINT `PK_Produced` PRIMARY KEY
27      → (`IDPlant`,`IDFarm`,`IDWarehouse`));
28
29  CREATE TABLE `Plant` (
30      `IDPlant` INT NOT NULL auto_increment,
31      `Name` VARCHAR(50) ,
32      `Variety` VARCHAR(50) ,
```

A DATABASE GENERATION

```
32         `Type` VARCHAR(50)           ,
33         `MST` INT                     ,
34         `Family` VARCHAR(50)         ,
35         CONSTRAINT `PK_Plant` PRIMARY KEY (`IDPlant`));
36
37 CREATE TABLE `Makes` (
38     `IDCostumer` INT NOT NULL auto_increment,
39     `IDOrder` INT NOT NULL,
40     CONSTRAINT `PK_Makes` PRIMARY KEY
41     ↪ (`IDCostumer`, `IDOrder`));
42
43 CREATE TABLE `Include` (
44     `IDOrder` INT NOT NULL,
45     `IDPlant` INT NOT NULL,
46     `IDFarm` INT NOT NULL,
47     `IDWarehouse` INT NOT NULL,
48     `Amount` INT                     ,
49     CONSTRAINT `PK_Include` PRIMARY KEY
50     ↪ (`IDOrder`, `IDPlant`, `IDFarm`, `IDWarehouse`));
51
52 CREATE TABLE `Farm` (
53     `IDFarm` INT NOT NULL auto_increment,
54     `Name` VARCHAR(50)           ,
55     `Address` VARCHAR(50)         ,
56     `Owner` VARCHAR(50)           ,
57     CONSTRAINT `PK_Table1` PRIMARY KEY (`IDFarm`));
58
59 CREATE TABLE `Costumer` (
60     `IDCostumer` INT NOT NULL auto_increment,
61     `Name` VARCHAR(50)           ,
62     `Mail` VARCHAR(50)           ,
63     `Phone` VARCHAR(50)           ,
64     `Address` VARCHAR(50)         ,
65     CONSTRAINT `Costumer` PRIMARY KEY(`IDCostumer`));
66
67 /* Create Foreign Key Constraints */
```

A DATABASE GENERATION

```
66
67 ALTER TABLE `Produced`
68   ADD CONSTRAINT `FK_Produced_Farm`
69     FOREIGN KEY (`IDFarm`) REFERENCES `Farm` (`IDFarm`) ON
      ↳ DELETE Restrict ON UPDATE Restrict;
70
71 ALTER TABLE `Produced`
72   ADD CONSTRAINT `FK_Produced_Plant`
73     FOREIGN KEY (`IDPlant`) REFERENCES `Plant` (`IDPlant`) ON
      ↳ DELETE Restrict ON UPDATE Restrict;
74
75 ALTER TABLE `Produced`
76   ADD CONSTRAINT `FK_Produced_Warehouse`
77     FOREIGN KEY (`IDWarehouse`) REFERENCES `Warehouse`
      ↳ (`IDWarehouse`) ON DELETE Restrict ON UPDATE
      ↳ Restrict;
78
79 ALTER TABLE `Makes`
80   ADD CONSTRAINT `FK_Makes_Costumer`
81     FOREIGN KEY (`IDCostumer`) REFERENCES `Costumer`
      ↳ (`IDCostumer`) ON DELETE Restrict ON UPDATE Restrict;
82
83 ALTER TABLE `Makes`
84   ADD CONSTRAINT `FK_Makes_TOrder`
85     FOREIGN KEY (`IDOrder`) REFERENCES `TOrder` (`IDOrder`)
      ↳ ON DELETE Restrict ON UPDATE Restrict;
86
87 ALTER TABLE `Include`
88   ADD CONSTRAINT `FK_Include_Produced`
89     FOREIGN KEY (`IDPlant`,`IDFarm`,`IDWarehouse`) REFERENCES
      ↳ `Produced` (`IDPlant`,`IDFarm`,`IDWarehouse`) ON
      ↳ DELETE Restrict ON UPDATE Restrict;
90
91 ALTER TABLE `Include`
92   ADD CONSTRAINT `FK_Include_TOrder`
```

A DATABASE GENERATION

```
93      FOREIGN KEY (`IDOrder`) REFERENCES `TOrder` (`IDOrder`)
      ↪ ON DELETE Restrict ON UPDATE Restrict;
94
95 SET FOREIGN_KEY_CHECKS=1 ;
```

The following code was added in order to populate the tables.

```
1  load data local infile 'Path\Warehouse.csv'
2  into table Warehouse
3  fields terminated by ','
4  lines terminated by '\r\n'
5  IGNORE 1 ROWS;
6
7  load data local infile 'Path\Costumer.csv'
8  into table Costumer
9  fields terminated by ','
10 lines terminated by '\r\n'
11 IGNORE 1 ROWS;
12
13 load data local infile 'Path\Plant.csv'
14 into table Plant
15 fields terminated by ','
16 lines terminated by '\r\n'
17 IGNORE 1 ROWS;
18
19 load data local infile 'Path\Order.csv'
20 into table TOrder
21 fields terminated by ','
22 lines terminated by '\r\n'
23 IGNORE 1 ROWS;
24
25 load data local infile 'Path\Farm.csv'
26 into table Farm
27 fields terminated by ','
28 lines terminated by '\r\n'
```

A DATABASE GENERATION

```
29  IGNORE 1 ROWs;
30
31  load data local infile 'Path\r_makes.csv'
32  into table Makes
33  fields terminated by ','
34  lines terminated by '\r\n'
35  IGNORE 1 ROWs;
36
37  load data local infile 'Path\r_Produced.csv'
38  into table Produced
39  fields terminated by ','
40  lines terminated by '\r\n'
41  IGNORE 1 ROWs;
42
43  load data local infile 'Path\r_include.csv'
44  into table Include
45  fields terminated by ','
46  lines terminated by '\r\n'
47  IGNORE 1 ROWs;
```