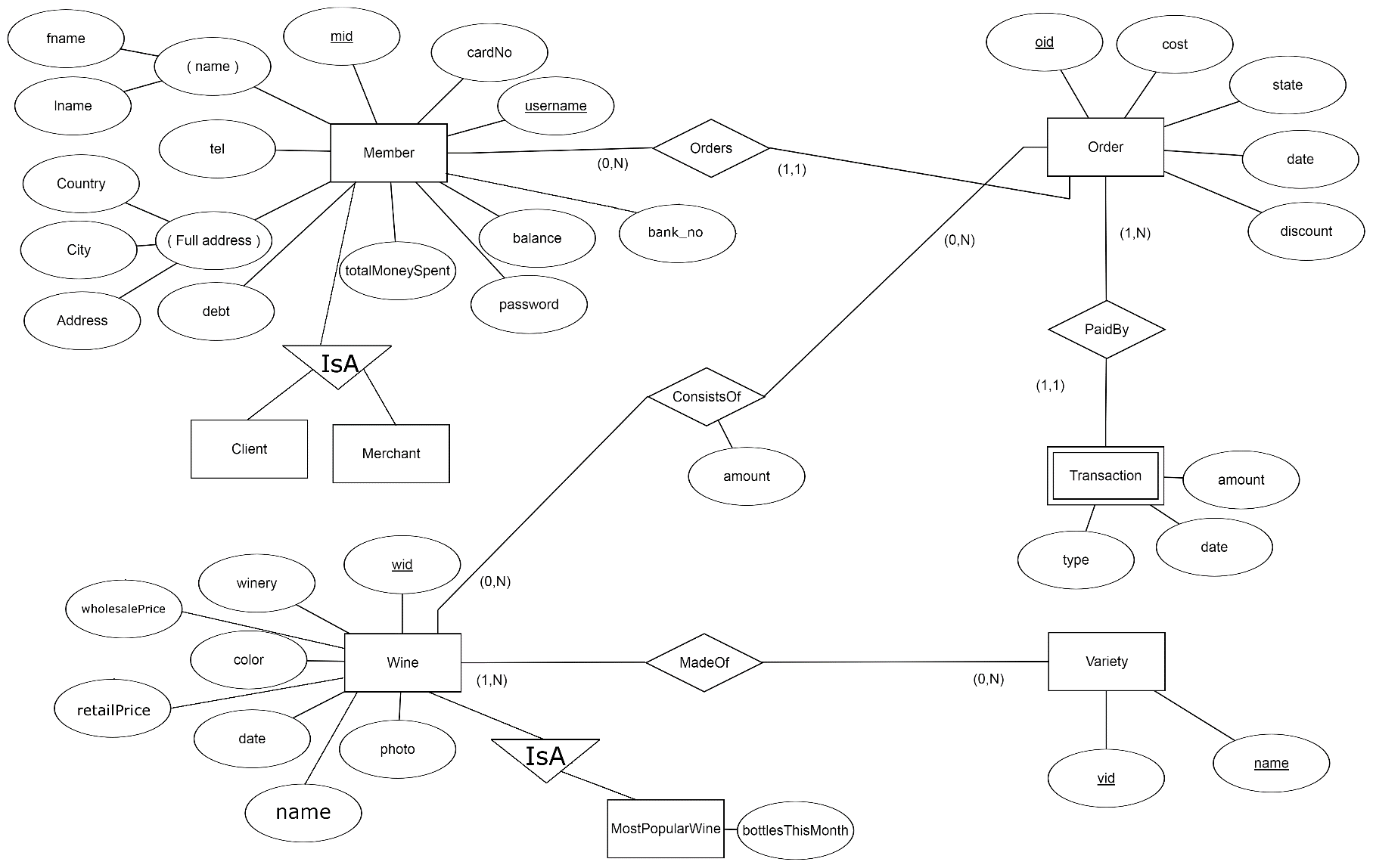
ΗΥ-360 Αρχεία και Βάσεις Δεδομένων   
Αναφορά Ομαδικής Εργασίας  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Wines of Crete E-SHOP

* Γιορταμής Μάνος
  + - csd3575
* Λιβανίδης Αναστάσης
  + - csd3646
* Λυδάκης Γιώργος
  + - csd3581

Περιεχόμενα

* Ενότητα 1: Μοντέλο Οντοτήτων Σχέσεων
  + Κεφάλαιο 1.1: Πλήρες Διάγραμμα Οντοτήτων-Σχέσεων
  + Κεφάλαιο 1.2: Γνωρίσματα
    - 1.2.1: Γνωρίσματα Οντοτήτων
    - 1.2.2: Γνωρίσματα Σχέσεων
  + Κεφάλαιο 1.3: Πρωτεύοντα Κλειδιά
  + Κεφάλαιο 1.4: Επεξηγήσεις
    - 1.4.1: Επεξηγήσεις Οντοτήτων
    - 1.4.2: Επεξηγήσεις Σχέσεων
  + Κεφάλαιο 1.5: Περιορισμοί Πληθικότητας
* Ενότητα 2: Σχεσιακό Μοντέλο Δεδομένων
  + Κεφάλαιο 2.1: Μετάφραση Μοντέλου σε Σχεσιακό Μοντέλο Δεδομένων
* Ενότητα 3: Γλώσσα Ορισμού Δεδομένων
  + Κεφάλαιο 3.1: Εντολές Γλώσσας Ορισμού Δεδομένων για τις Σχέσεις
  + Κεφάλαιο 3.2: Περιορισμοί Ακεραιότας
* Ενότητα 4: Κανονικές Μορφές
  + Κεφάλαιο 4.1: Συναρτησιακές Εξαρτήσεις
  + Κεφάλαιο 4.2: Καθορισμός Κλειδιών βάσει Συναρτησιακών Εξαρτήσεων
  + Κεφάλαιο 4.3: Μετατροπή του Μοντέλου σε Τρίτη Κανονική Μορφή με διατήρηση των συναρτησιακών εξαρτήσεων και χωρίς απώλεια πληροφορίας
* Ενότητα 5: S.Q.L.
  + Κεφάλαιο 5.1: Περιγραφή των Ερωτήσεων προς τη Βάση Δεδομένων
    - 5.1.1: MemberDB
    - 5.1.2: OrderDB
    - 5.1.3: OrderConsistsOfWineDB
    - 5.1.4: WineDB
    - 5.1.5: VarietyDB
    - 5.1.6: TransactionsDB
* Ενότητα 6: Εγχειρίδιο Χρήσης του Συστήματος
  + Κεφάλαιο 6.1: Επιτρεπτές λειτουργίες χωρίς σύνδεση (login)
    - 6.1.1: Δημοφιλή Κρασιά (Trending)
    - 6.1.2: Καλοί και Κακοί Πελάτες (Reliable and Unreliable Members)
  + Κεφάλαιο 6.2: Εγγραφή Χρήστη (Register)
    - 6.2.1: Όνομα Χρήστη (username)
    - 6.2.2: Κωδικός Πρόσβασης (password)
    - 6.2.3: Ονοματεπώνυμο (first/last name)
    - 6.2.4: Τηλέφωνο (telephone)
    - 6.2.5: Αριθμός Πιστωτικής Κάρτας (card number)
    - 6.2.6: Διεύθυνση (country/city/address)
    - 6.2.7: Έλεγχος Εμπόρων (are you a merchant?)
  + Κεφάλαιο 6.3: Λειτουργίες σε σύνδεση (logged in)
    - 6.3.1: Περί session και cookie
    - 6.3.2: E-SHOP
    - 6.3.3: Καλάθι Αγορών
    - 6.3.4: Πληροφορίες Λογαριασμού
      * 6.3.4.1: Γενικές Πληροφορίες (Account Details)
      * 6.3.4.2: Ιστορικό Παραγγελιών (Order History)
      * 6.3.4.3: Ιστορικό Δοσοληψιών (Transaction History)
    - 6.3.5: Υπόλοιπο Λογαριασμού (Balance)
    - 6.3.6: Κλείσιμο Λογαριασμού
* Ενότητα 7: Ενδεικτικά Αποτελέσματα
* Ενότητα 8: Περιορισμοί Υλοποίησης και Δυνατότητες Βελτίωσης

Ενότητα 1: Μοντέλο Οντοτήτων Σχέσεων

 Κεφάλαιο 1.1: Πλήρες Διάγραμμα Οντοτήτων-Σχέσεων

Κεφάλαιο 1.2: Γνωρίσματα

1.2.1: Γνωρίσματα Οντοτήτων

**Member:**

-**mid**, εσωτερικό κλειδί (αύξων αριθμός) της βάσης για την δεικτοδότηση των μελών  
-**name**, που αποτελείται από το μικρό όνομα (**fname**) και το επίθετο (**lname**) του μέλους  
-**tel**, τηλέφωνο μέλους  
-**full address**, διέυθυνση του μέλους που αποτελείται από την χώρα (**country**), πόλη (**city**) και διεύθυνση εντός της πόλης (**address**)  
-**debt**, χρήματα που ο χρήστης χρωστάει στο κατάστημα  
-**totalMoneySpent**, σύνολο χρημάτων που ο χρήστης έχει ξοδέψει σε παραγγελίες  
-**password**, ο κωδικός εισόδου του χρήστη στο site  
-**balance,** χρήματα που ο χρήστης έχει αυτή τη στιγμή στο λογαριασμό του  
-**username,** όνομα του χρήστη στο site, χρησιμοποιείται στο login  
-**cardNo,** αριθμός πιστωτικής κάρτας που χρησιμοποιεί ο χρήστης  
-**bankNo**, αριθμός τραπεζικού λογαριασμού που καταθέτουμε τα χρήματα από επιστροφές παραγγελιών

**Client:**

-**mid,** το ίδιο εσωτερικό κλειδί που χρησιμοποιείται στη **Member**.

**Merchant:**

-**mid,** το ίδιο εσωτερικό κλειδί που χρησιμοποιείται στη **Member.**

Οι οντότητες αυτές συνδέονται με την **Member** με τη σχέση **IsA**, αφού κάθε ιδιώτης και κάθε έμπορος είναι και μέλος. Εν προκειμένω η σχέση **IsA** χαρακτηρίζεται από **Αποκλειστικότητα (Disjointness) και Πληρότητα (Completeness)**, δηλαδή ένα στιγμιότυπο ανήκει σε ΑΚΡΙΒΩΣ μια από τις **Client, Merchant** και αυτό διότι ένα μέλος δεν μπορεί συγχρόνως να είναι έμπορος και ιδιώτης, ενώ πρέπει οπωσδήποτε να είναι κάτι από τα δύο (δηλ. δεν μπορεί να υπάρξει στιγμιότυπο της **Member** που δεν ανήκει σε κάποια από τις **Client, Merchant**).

**Order**

-**oid**, εσωτερικό κλειδί (αύξων αριθμός) της βάσης για την δεικτοδότηση των παραγγελιών  
-**cost**, κόστος της παραγγελίας αυτής σε ευρώ  
**-state,** η κατάσταση στην οποία βρίσκεται η παραγγελία. Μπορεί να πάρει τις εξής τιμές: **AWAITING\_PAYMENT**, αν δεν έχει εξοφληθεί πλήρως ακόμη  
**FULLY\_PAID**, αν έχει εξοφληθεί πλήρως αλλά ο χρήστης ακόμη δεν την έχει λάβει  
**DELIVERED**, αν ο χρήστης έχει λάβει την παραγγελία αυτή (αυτό το δηλώνει μέσω κατάλληλου κουμπιού στο site, βλ. Εγχειρίδιο Συστήματος)  
**-date**, η ημερομηνία (χρόνος, μήνας, μέρα, ώρα, λεπτό, δευτερόλεπτο) κατά την οποία καταχωρήθηκε η παραγγελία  
-**discount**, το ποσοστό έκπτωσης που δικαιούται αυτός που έκανε την παραγγελία. Εκπτώσεις δικαιούνται μόνο οι έμποροι, και παίρνουν 5% έκπτωση αν η παραγγελία είναι πάνω από 280 ευρώ, ή 10% αν είναι πάνω από 400 ευρώ

**Wine**

**-wid**, εσωτερικό κλειδί (αύξων αριθμός) της βάσης για την δεικτοδότηση των κρασιών  
**-winery,** το όνομα του οινοποιείου που παράγει αυτό το κρασί  
**-wholesalePrice,** τιμή χονδρικής πώλησης του κρασιού  
**-retailPrice,** τιμή λιανικής πώλησης του κρασιού  
-**color,** το χρώμα του κρασιού  
**-date,** το έτος παραγωγής του κρασιού  
**-name,** το όνομα του κρασιού  
**-photo,** μια φωτογραφία του κρασιού Αποθηκεύεται στη βάση ως url που δείχνει σε μια εικόνα αποθηκευμένη στο μηχάνημα όπου είναι hosted το site

**MostPopularWine**

**-bottlesThisMonth,** ο αριθμός των μπουκαλιών του κρασιού που πουλήθηκαν τον εκάστοτε μήνα

**Variety**

**-vid,** εσωτερικό κλειδί (αύξων αριθμός) της βάσης για την δεικτοδότηση των ποικιλιών σταφυλιού  
**-name,** όνομα της ποικιλίας

**Transaction**

**-type,** το είδος της δοσοληψίας. Παίρνει τις εξής τιμές: **PAYMENT,** αν πρόκεται για πληρωμή ενός μέλους προς το κατάστημα **REFUND,** αν πρόκειται για επιστροφή χρημάτων από το κατάστημα προς κάποιο μέλος **-date,** ημερομηνία (χρόνος, μήνας, μέρα, ώρα, λεπτό, δευτερόλεπτο) κατά την οποία έγινε η δοσοληψία  
**-amount,** το χρηματικό ποσό της δοσοληψίας

1.2.2: Γνωρίσματα Σχέσεων

**ConsistsOf**

**-amount,** το πλήθος των μπουκαλιών ενός συγκεκριμένου είδους κρασιού που υπάρχουν σε μια παραγγελία

Κεφάλαιο 1.3: Πρωτεύοντα Κλειδιά

-**Member**, πρωτεύον κλειδί **mid**

-**Client**, πρωτεύον κλειδί **mid**

-**Merchant**, πρωτεύον κλειδί **mid**

-**Order**, πρωτεύον κλειδί **oid**

-**Wine**, πρωτεύον κλειδί **wid**

-**Transaction**, ως ασθενής οντότητα δεν έχει πρωτεύον κλειδί, αλλά έχει ως μερικό αναγνωριστικό το συνδυασμό του γνωρίσματος **date** και του κλειδιού **oid** της **Order.**

-**Variety**, πρωτεύον κλειδί **vid**

-**MostPopularWine**, πρωτεύον κλειδί **wid**

**-OrderConsistsOfWine**, πρωτεύον κλειδί το σύνολο (**oid, wid**)

**-WineMadeOfVariety**, πρωτεύον κλειδί το σύνολο (**wid, vid**) (τετριμμένο).

Κεφάλαιο 1.4: Επεξηγήσεις

1.4.1: Επεξηγήσεις Οντοτήτων

-**Member**, μέλος του ηλεκτρονικού καταστήματος

-**Client**, μέλος του ηλεκτρονικού καταστήματος που είναι ιδιώτης

-**Merchant**, μέλος του ηλεκτρονικού καταστήματος που είναι έμπορος

-**Order**, παραγγελία ενός χρήστη

-**Wine**, κρασί που διατίθεται στο κατάστημα

-**Transaction**, δοσοληψία του χρήστη (πληρωμή ή επιστροφή χρημάτων) με το κατάστημα. Πρόκειται περί ασθενούς οντότητας εξαρτώμενης από την **Order**, μιας και μια δοσοληψία έχει νόημα μόνο σε σχέση με την παραγγελία με την οποία σχετίζεται.

-**Variety**, ποικιλία σταφυλιού από την οποία παράγονται κρασιά

-**MostPopularWine**, κρασί το οποίο είχε μη μηδενικές πωλήσεις τον αμέσως προηγούμενο μήνα από τον τρέχοντα

1.4.2: Επεξηγήσεις Σχέσεων

**-Orders,** σχέση που συνδέει τις οντότητες **Member** και **Order**. Χρησιμοποιείται για να εκφράσει την σχέση ενός μέλους με τις παραγγελίες του.

**-ConsistsOf,** σχέση που συνδέει τις οντότητες **Wine** και **Order.** Χρησιμοποιείται για να εκφράσει την σχέση μιας παραγγελίας με τα κρασιά που την αποτελούν.

**-PaidBy,** σχέση που συνδέει τις οντότητες **Order** και **Transaction.** Χρησιμοποιείται για να εκφράσει την σχέση μιας παραγγελίας με τις δοσοληψίες σχετικές με αυτήν (δηλαδή δοσοληψίες που την εξοφλούν ή που επιστρέφουν μέρη της στο κατάστημα).

**-MadeOf,** σχέση που συνδέει τις οντότητες **Wine** και **Variety.** Χρησιμοποιείται για να εκφράσει την σχέση ενός κρασιού με τις ποικιλίες σταφυλιού από τις οποίες παράγεται.

Κεφάλαιο 1.5: Περιορισμοί Πληθικότητας

**-Orders:** Ένα μέλος μπορεί ανά πάσα στιγμή να έχει καμία ή περισσότερες παραγγελίες στο λογαριασμό του, άρα η **Member** έχει προαιρετική συμμετοχή και max-card(Member, Orders) = N. Μία παραγγελία σχετίζεται με ακριβώς ένα μέλος, άρα η **Order** έχει υποχρεωτική συμμετοχή και max-card(Order, Orders) = 1.

**-ConsistsOf:** Ένα κρασί μπορεί να ανήκει σε καμία ή περισσότερες παραγγελίες, άρα η **Wine** έχει προαιρετική συμμετοχή και max-card(Wine, ConsistsOf) = N. Μία παραγγελία αποτελείται από κανένα ή περισσότερα κρασιά, άρα η **Order** έχει προαιρετική συμμετοχή και max-card(Order, ConsistsOf) = N. Ο λόγος που έχει προαιρετική συμμετοχή είναι ότι κάναμε τη σχεδιαστική επιλογή να μην διαγράφουμε ποτέ παραγγελίες από το σύστημα, ώστε ο πελάτης να βλέπει και τις παραγγελίες που έχει ακυρώσει πλήρως. Άρα, μπορεί μια παραγγελία να καταλήξει να είναι «άδεια» από κρασιά.

**-PaidBy:** Μία παραγγελία συνδέεται με τουλάχιστον μία δοσοληψία και ίσως με περισσότερες, διότι κάνουμε τη σχεδιαστική επιλογή να συνδέουμε πάντα μια παραγγελία με μία «αρχική δοσοληψία», όταν η παραγγελία καταχωρείται στο σύστημα. Αυτό γίνεται ακόμα και αν ο πελάτης δεν πληρώνει τίποτα όταν παραγγέλνει, οπότε η δοσοληψία καταχωρείται με μηδενικό **amount.** Άρα η **Order** έχει υποχρεωτική συμμετοχή στην **PaidBy** και max-card(Order, PaidBy) = N. Μία δοσοληψία συνδέεται με ακριβώς μία παραγγελία, άρα έχει υποχρεωτική συμμετοχή και max-card(Transaction, PaidBy) = 1.

**-MadeOf:** Ένα κρασί παράγεται από τουλάχιστον μία παραγγελία και πιθανώς από περισσότερες, άρα έχει υποχρεωτική συμμετοχή και max-card(Wine, MadeOf) = N. Μία ποικιλία μπορεί να μην συνδέεται ακόμη με κάποιο κρασί, σε περίπτωση που δεν έχει εισαχθεί στη βάση κρασί παραγόμενο από αυτή, ενώ μπορεί να παράγει ένα η περισσότερα κρασιά. Άρα έχει προαιρετική συμμετοχή και max-card(Variety, MadeOf) = N.

Ενότητα 2: Σχεσιακό Μοντέλο Δεδομένων

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| mid | username | password | fname | lname | tel | address | city | country | debt | totalMoneySpent | cardno | balance |

Member

bank\_no

Σχόλια: Απλή Μετάφραση των Γνωρισμάτων του E.R.

Client Merchant

|  |
| --- |
| mid |

|  |
| --- |
| mid |

Σχόλια: Οντότητες που σχετίζονται με ISA στο E.R. με την Member. Περιέχουν μόνο το ξένο κλειδί mid, το οποίο αποτελεί κλειδί και για κάθε μια από αυτές.

Wine

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| wid | retailPrice | wholesalePrice | winery | name | color | date | photo |

Σχόλια: Απλή Μετάφραση των Γνωρισμάτων του E.R.

MostPopularWines

|  |  |
| --- | --- |
| wid | bottlesThisMonth |

Σχόλια: Οντότητα που σχετίζεται με ISA στο E.R. με την Wine. Περιέχει το ξένο κλειδί wid (που είναι ταυτόχρονα και δικό της κλειδί) μαζί με το γνώρισμά της bottlesThisMonth

Ο πίνακας αυτός «καθαρίζεται» αυτόματα από τη βάση στην αρχή κάθε μήνα και γεμίζει εκ νέου με πλειάδες που αντιστοιχούν τα κρασιά με τις πωλήσεις τους τον μήνα που μόλις τελείωσε. Αυτό το πετυχαίνουμε μέσω του event system της phpmyadmin ορίζοντας το ακόλουθο event:

CREATE DEFINER=`root`@`localhost` EVENT `update\_most\_popular\_wines` ON SCHEDULE EVERY 1 MONTH STARTS '2018-01-01 00:00:00' ON COMPLETION NOT PRESERVE ENABLE DO BEGIN

DELETE

FROM

mostpopularwines;

INSERT INTO mostpopularwines(

SELECT

wine.wid,

SUM(amount)

FROM

wine,

orderconsistsofwine,

order\_t

WHERE

order\_t.oid = orderconsistsofwine.oid and wine.wid = orderconsistsofwine.wid and (MONTH(CURDATE()) - MONTH(order\_t.date) = 1 or MONTH(CURRENT\_DATE) - MONTH(order\_t.date) = -11)

GROUP BY

orderconsistsofwine.wid

);

END

Αυτό τρέχει στην αρχή του κάθε μήνα και για κάθε **wid** βρίσκει όλες τις παραγγελίες που έγιναν τον προηγούμενο μήνα και που το περιλαμβάνουν. Αθροίζει όλες τις πωλήσεις του κάθε **wid** και τις καταχωρεί στον πίνακα **mostPopularWines**.

Variety

|  |  |
| --- | --- |
| vid | name |

Σχόλια: Απλή Μετάφραση των Γνωρισμάτων του E.R.

WineMadeOfVariety

|  |  |
| --- | --- |
| wid | vid |

Σχόλια: Προκύπτει απ’ την N-N σχέση MadeOf του E.R.

Order

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| oid | mid | date | cost | discount | state |

Σχόλια: Απλή μετάφραση των γνωρισμάτων του E.R. ενώ το ξένο κλειδί mid προκύπτει απ’ την πληθικότητα της σχέσης Orders, που επιβάλει πως κάθε παραγγελία σχετίζεται με ακριβώς ένα μέλος.

OrderConsistsOfWine

|  |  |  |
| --- | --- | --- |
| oid | wid | amount |

Σχόλια: Προκύπτει απ’ την N-N σχέση ConsistsOf του E.R.

Transactions

|  |  |  |  |
| --- | --- | --- | --- |
| oid | type | amount | date |

Σχόλια: Απλή μετάφραση της ασθενούς οντότητας Transactions, με oid ως ξένο κλειδί απ’ την Order. Η paidBy του ER υλοποιείται με την ύπαρξη του oid στην Transactions (εφόσον κάθε transaction συνδέεται με ακριβώς μία order, σχέση 1-1).

Ενότητα 3: Γλώσσα Ορισμού Δεδομένων

Κεφάλαιο 3.1: Εντολές Ορισμού Δεδομένων για τις Σχέσεις

Εντολές για το ορισμό σχέσεων και τον ορισμό κλειδιών στις σχέσεις:

CREATE TABLE `member` (

`mid` int(11) NOT NULL,

`username` varchar(30) NOT NULL,

`password` varchar(50) NOT NULL,

`fname` varchar(30) NOT NULL,

`lname` varchar(30) NOT NULL,

`tel` varchar(10) NOT NULL,

`address` varchar(30) NOT NULL,

`city` varchar(30) NOT NULL,

`country` varchar(30) NOT NULL,

`debt` decimal(6,2) NOT NULL,

`totalMoneySpent` decimal(6,2) NOT NULL,

`cardno` varchar(19) NOT NULL,

`balance` decimal(6,2) NOT NULL

);

CREATE TABLE `client` (

`mid` int(11) NOT NULL

);

-- RELATIONSHIPS FOR TABLE `client`:

-- `mid`

-- `member` -> `mid`

--

CREATE TABLE `merchant` (

`mid` int(11) NOT NULL

);

-- RELATIONSHIPS FOR TABLE `merchant`:

-- `mid`

-- `member` -> `mid`

--

CREATE TABLE `wine` (

`wid` int(11) NOT NULL,

`retailPrice` decimal(6,2) NOT NULL,

`wholesalePrice` decimal(6,2) NOT NULL,

`winery` varchar(50) NOT NULL,

`name` varchar(30) NOT NULL,

`color` varchar(30) NOT NULL,

`date` varchar(10) NOT NULL,

`photo` varchar(100) NOT NULL

);

CREATE TABLE `mostpopularwines` (

`wid` int(11) NOT NULL,

`bottlesThisMonth` int(11) NOT NULL

);

-- RELATIONSHIPS FOR TABLE `mostpopularwines`:

-- `wid`

-- `wine` -> `wid`

--

CREATE TABLE `variety` (

`vid` int(11) NOT NULL,

`name` varchar(30) CHARACTER SET utf8 NOT NULL

);

CREATE TABLE `winemadeofvariety` (

`wid` int(11) NOT NULL,

`vid` int(11) NOT NULL

);

-- RELATIONSHIPS FOR TABLE `winemadeofvariety`:

-- `vid`

-- `variety` -> `vid`

-- `wid`

-- `wine` -> `wid`

--

CREATE TABLE `order\_t` (

`oid` int(11) NOT NULL,

`mid` int(11) DEFAULT NULL,

`date` datetime NOT NULL,

`cost` decimal(6,2) NOT NULL,

`discount` int(11) NOT NULL,

`state` varchar(20) NOT NULL

);

-- RELATIONSHIPS FOR TABLE `order\_t`:

-- `mid`

-- `member` -> `mid`

--

CREATE TABLE `orderconsistsofwine` (

`oid` int(11) NOT NULL,

`wid` int(11) NOT NULL,

`amount` int(11) NOT NULL

);

-- RELATIONSHIPS FOR TABLE `orderconsistsofwine`:

-- `wid`

-- `wine` -> `wid`

-- `oid`

-- `order\_t` -> `oid`

--

CREATE TABLE `transactions` (

`oid` int(11) NOT NULL,

`type` varchar(20) NOT NULL,

`amount` decimal(6,2) NOT NULL,

`date` datetime NOT NULL

);

-- RELATIONSHIPS FOR TABLE `transactions`:

-- `oid`

-- `order\_t` -> `oid`

--

ALTER TABLE `client`

ADD PRIMARY KEY (`mid`);

--

-- Indexes for table `member`

--

ALTER TABLE `member`

ADD PRIMARY KEY (`mid`);

--

-- Indexes for table `merchant`

--

ALTER TABLE `merchant`

ADD PRIMARY KEY (`mid`);

--

-- Indexes for table `mostpopularwines`

--

ALTER TABLE `mostpopularwines`

ADD PRIMARY KEY (`wid`);

--

-- Indexes for table `orderconsistsofwine`

--

ALTER TABLE `orderconsistsofwine`

ADD PRIMARY KEY (`oid`,`wid`),

ADD KEY `FK\_order` (`oid`),

ADD KEY `FK\_Wine` (`wid`);

--

-- Indexes for table `order\_t`

--

ALTER TABLE `order\_t`

ADD PRIMARY KEY (`oid`),

ADD KEY `FK\_Order\_t\_Member` (`mid`);

--

-- Indexes for table `transactions`

--

ALTER TABLE `transactions`

ADD PRIMARY KEY (`oid`,`date`),

ADD KEY `FK\_Payment\_Order` (`oid`);

--

-- Indexes for table `variety`

--

ALTER TABLE `variety`

ADD PRIMARY KEY (`vid`),

ADD UNIQUE KEY `name` (`name`);

--

-- Indexes for table `wine`

--

ALTER TABLE `wine`

ADD PRIMARY KEY (`wid`);

--

-- Indexes for table `winemadeofvariety`

--

ALTER TABLE `winemadeofvariety`

ADD PRIMARY KEY (`wid`,`vid`),

ADD KEY `FK\_Variety` (`vid`),

ADD KEY `FK\_WineMadeOfVariety\_Wine` (`wid`);

Εντολές για εισαγωγή κρασιών στη βάση

Σημείωση: Η επιλογή των κρασιών έγινε τυχαία και για σκοπούς δοκιμής και επίδειξης. Καμιά επιλογή δεν εκφράζει τις προσωπικές απόψεις των σχεδιαστών ή οποιαδήποτε προκατάληψη γενικότερα.

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(1, '10.00', '8.00', 'Δουλουφάκης Οινοποιείο', 'CHARDONNAY Λευκός Ξηρός οίνος', 'Λευκό', '2015', 'images/1.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(2, '8.00', '6.00', 'Δουλουφάκης Οινοποιείο', 'FEMINA Λευκός Ξηρός οίνος', 'Λευκό', '2013', 'images/2.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(3, '7.50', '5.50', 'ALEXAKIS WINERY', 'Malvasia Aromatica', 'Λευκό', '2012', 'images/3.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(4, '8.90', '6.90', 'ALEXAKIS WINERY', 'Moscato Sweetheart', 'Λευκό', '2016', 'images/4.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(5, '12.00', '10.00', 'Μανουσάκη Οινοποιία', 'Nostos Mourvèdre', 'Κόκκινο', '2015', 'images/5.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(6, '13.00', '11.00', 'Αμπελώνες Καραβιτάκη', 'Ελιά Καραβιτάκ', 'Κόκκινο', '2017', 'images/6.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(7, '12.30', '10.30', 'Μανουσάκη Οινοποιία', '2 Mazi Red', 'Κόκκινο', '2015', 'images/7.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(8, '8.60', '6.60', 'Μανουσάκη Οινοποιία', '2 Mazi Rosé', 'Ροζέ', '2016', 'images/8.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(9, '7.80', '5.80', 'Μανουσάκη Οινοποιία', '2 Mazi White', 'Λευκό', '2005', 'images/9.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(10, '6.50', '4.50', 'ΜΙΝΩΣ Κρασιά Κρήτης ΑΕ – Οινοποιείο Μηλιαράκη', '35o - 25o Λευκό', 'Λευκό', '2007', 'images/10.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(12, '8.70', '6.70', 'ΜΙΝΩΣ Κρασιά Κρήτης ΑΕ – Οινοποιείο Μηλιαράκη', '35ο - 25ο Ερυθρό', 'Κόκκινο', '2014', 'images/11.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(14, '13.00', '11.00', 'ΜΙΝΩΣ Κρασιά Κρήτης ΑΕ – Οινοποιείο Μηλιαράκη', '35ο - 25ο Ροζέ', 'Ροζέ', '2013', 'images/12.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(15, '15.00', '13.00', 'Οινοποιείο Αφοί Σπ. Μαραγκάκη', '8η Τέχνη', 'Λευκό', '2016', 'images/13.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(16, '9.00', '7.00', 'Ρους Οινοποιία Ταμιωλακη', 'Ahinos', 'Λευκό', '2002', 'images/14.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(17, '13.30', '11.30', 'Δουλουφάκης Οινοποιείο', 'ALARGO Ερυθρός Ξηρός οίνος', 'Βαθύ κόκκινο χρώμα', '2009', 'images/15.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(18, '12.80', '10.80', 'ALEXAKIS WINERY', 'Artis ', 'Κόκκινο', '2015', 'images/16.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(19, '12.50', '10.50', 'ALEXAKIS WINERY', 'Artis', 'Λευκό', '2015', 'images/17.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(20, '14.00', '12.00', 'Κτήμα Μιχαλάκη', 'CABERNET SAUVIGNON', 'Κόκκινο', '2017', 'images/18.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(21, '9.00', '7.00', 'Οινοποιείο Ευφροσύνη', 'CABERNET SAUVIGNON “INDIGO”', 'Κόκκινο', '2014', 'images/19.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(22, '14.20', '12.20', 'Ιδαια Οινοποιητική ', 'CABERNET SAUVIGNON, ΙΔΑΙΑ', 'Κόκκινο', '2013', 'images/20.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(23, '7.50', '5.50', 'Κτήμα Μιχαλάκη', 'CHARDONNAY', 'Λευκό', '2014', 'images/21.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(24, '8.20', '6.20', 'Λυραράκης - ΓΕΑ ΑΕ', 'Λυραράκης Δαφνί \"Ψαράδες', 'Λευκό', '2015', 'images/22.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(25, '15.00', '13.00', 'Δουλουφάκης Οινοποιείο', 'DAFNIOS ερυθρός', 'Κόκκινο', '2017', 'images/23.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(26, '14.00', '12.00', 'Δουλουφάκης Οινοποιείο', 'DAFNIOS λευκός', 'Λευκό', '2012', 'images/24.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(27, '14.70', '12.70', 'Silva Δασκαλάκη', 'Emphasis Ερυθρό', 'Κόκκινο', '2013', 'images/25.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(28, '16.00', '14.00', 'Silva Δασκαλάκη', 'Emphasis Λευκό', 'Λευκό', '2013', 'images/26.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(29, '13.70', '11.70', 'Κτήμα Μιχαλάκη', 'LATO ΡΟΖΕ', 'Ροζέ', '2016', 'images/27.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`) VALUES(30, '15.70', '13.70', 'Κτήμα Μιχαλάκη', 'LE MANOIR ΡΟΖΕ ΜΙΧΑΛΑΚΗ', 'Ροζέ', '2011', 'images/28.jpg');

INSERT INTO `wine` (`wid`, `retailPrice`, `wholesalePrice`, `winery`, `name`, `color`, `date`, `photo`)VALUES(31, '12.50', '10.50', 'Δουλουφάκης Οινοποιείο', 'ENOTRIA ροζέ', 'Ροζέ', '2002', 'images/29.jpg');

Εντολές για εισαγωγή ποικιλιών στη βάση:

INSERT INTO `variety` (`vid`, `name`) VALUES(12, 'Cabernet Sauvignon');

INSERT INTO `variety` (`vid`, `name`) VALUES(13, 'Chardonnay');

INSERT INTO `variety` (`vid`, `name`) VALUES(14, 'Merlot');

INSERT INTO `variety` (`vid`, `name`) VALUES(18, 'Roussanne');

INSERT INTO `variety` (`vid`, `name`) VALUES(17, 'Sangiovese');

INSERT INTO `variety` (`vid`, `name`) VALUES(15, 'Sauvignon Blanc');

INSERT INTO `variety` (`vid`, `name`) VALUES(16, 'Syrah');

INSERT INTO `variety` (`vid`, `name`) VALUES(1, 'Βηλάνα');

INSERT INTO `variety` (`vid`, `name`) VALUES(2, 'Βιδιανό');

INSERT INTO `variety` (`vid`, `name`) VALUES(3, 'Δαφνί');

INSERT INTO `variety` (`vid`, `name`) VALUES(4, 'Θραψαθήρι');

INSERT INTO `variety` (`vid`, `name`) VALUES(5, 'Κοτσιφάλι');

INSERT INTO `variety` (`vid`, `name`) VALUES(6, 'Λιάτικο');

INSERT INTO `variety` (`vid`, `name`) VALUES(7, 'Μαλβαζία');

INSERT INTO `variety` (`vid`, `name`) VALUES(8, 'Μαντηλάρι');

INSERT INTO `variety` (`vid`, `name`) VALUES(9, 'Μοσχάτο Λευκό');

INSERT INTO `variety` (`vid`, `name`) VALUES(10, 'Πλυτό');

INSERT INTO `variety` (`vid`, `name`) VALUES(11, 'Ρωμέικο');

Εντολές για εισαγωγή σχέσης κρασιών-ποικιλιών στη βάση:

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(1, 13);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(2, 7);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(3, 7);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(4, 7);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(4, 9);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(5, 5);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(6, 16);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(7, 8);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(7, 11);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(8, 8);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(8, 11);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(9, 1);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(9, 18);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(10, 1);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(10, 4);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(12, 5);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(12, 8);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(12, 16);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(14, 5);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(14, 16);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(15, 2);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(16, 15);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(17, 16);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(18, 5);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(19, 2);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(19, 9);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(20, 12);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(21, 12);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(22, 12);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(23, 12);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(24, 3);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(25, 6);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(26, 2);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(27, 8);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(27, 12);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(28, 10);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(28, 15);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(29, 5);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(29, 8);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(30, 5);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(30, 8);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(31, 5);

INSERT INTO `winemadeofvariety` (`wid`, `vid`) VALUES(31, 16);

Κεφάλαιο 3.2: Περιορισμοί Ακεραιότητας

Στα παρακάτω επιλέξαμε να μην αλλάζουμε ποτέ το πρωτεύον κλειδί των πλειάδων, μιας και θεωρήσαμε ότι δεν υπάρχει λόγος να γίνεται αυτό, και απλά θα επέφερε επιπλοκές.

**Member**

**-**Όταν δημιουργείται ένα νέο μέλος, πρέπει να έχει μηδενικό **balance, debt, totalMoneySpent**. Επίσης, πρέπει το τηλέφωνό του, **tel**, να είναι ακριβώς δέκα ψηφία. Στη βάση αυτό ελέγχεται με το εξής trigger:

CREATE TRIGGER `member\_domain\_constraints` BEFORE INSERT ON `member`

FOR EACH ROW BEGIN

IF (NEW.tel REGEXP '[0-9]{10}') = 0 THEN

signal sqlstate '45000' set message\_text = 'Invalid phone number!';

END IF;

IF NEW.debt != 0 THEN

SET NEW.debt = 0;

END IF;

IF NEW.totalMoneySpent != 0 THEN

SET NEW.totalMoneySpent = 0;

END IF;

IF NEW.balance != 0 THEN

SET NEW.balance = 0;

END IF;

END

, το οποίο διασφαλίζει αν δοθούν μη μηδενικά **balance, debt, totalMoneySpent,** στη βάση θα εισαχθούν μηδενικά τέτοια γνωρίσματα, ενώ αν δοθεί άκυρο τηλέφωνο ειδοποιεί κατάλληλα με μήνυμα λάθους.

- Όταν γίνονται update τα στοιχεία ενός μέλους, δεν πρέπει ποτέ να αλλάξει το **mid,** όντας πρωτεύον κλειδί, ούτε το **username**, όντας επίσης αναγνωριστικό, ενώ δεν πρέπει να δοθεί άκυρο τηλέφωνο, ή αρνητικά **debt, balance, totalMoneySpent.** Στη βάση αυτό ελέγχεται με το εξής trigger:

CREATE TRIGGER `member\_update\_constraints` BEFORE UPDATE ON `member`

FOR EACH ROW BEGIN

IF (NEW.tel REGEXP '[0-9]{10}') = 0 THEN

signal sqlstate '45000' set message\_text = 'Invalid phone number!';

END IF;

IF NEW.debt < 0 THEN

signal sqlstate '45000' set message\_text = 'Invalid debt!';

END IF;

IF NEW.totalMoneySpent < 0 THEN

signal sqlstate '45000' set message\_text = 'Invalid total money spent for member!';

END IF;

IF NEW.mid != OLD.mid THEN

signal sqlstate '45000' set message\_text = 'Member id cannot be changed';

END IF;

IF NEW.username != OLD.username THEN

signal sqlstate '45000' set message\_text = 'Username cannot be changed';

END IF;

IF NEW.balance < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative balance!';

END IF;

END

-Ένα μέλος μπορεί να σβηστεί μόνο εφόσον δεν έχει χρέος και έχει παραλάβει όλες τις παραγγελίες του. Αυτό ελέγχεται με το εξής trigger:

CREATE TRIGGER `member\_delete\_constraints` BEFORE DELETE ON `member`

FOR EACH ROW BEGIN

IF OLD.debt != 0 THEN

signal sqlstate '45000' set message\_text = 'Cannot delete a member with debt';

END IF;

IF EXISTS(SELECT state FROM order\_t o WHERE o.mid = OLD.mid AND o.state != 'DELIVERED') THEN

signal sqlstate '45000' set message\_text = 'Cannot delete a member with undelivered orders';

END IF;

END

-Όταν εισάγεται ένα μέλος στον **Member**, πρέπει να εισαχθεί το **mid** σε ΑΚΡΙΒΩΣ έναν από τους πίνακες **Client, Merchant** (βλέπε σχέση IsA). Ο έλεγχος αυτός επιλέξαμε να γίνεται server-side, δηλαδή ο χρήστης της DB να εισάγει το μέλος στον πίνακα που αυτός αποφασίζει.

**Client**

**­**-Άπαξ και εισαχθεί ένα μέλος στον **Client**, δεν θα πρέπει ποτέ να αλλάζει το **mid** του, ως πρωτεύον κλειδί. Αυτό ελέγχεται με το εξής trigger:

CREATE TRIGGER `client\_update\_constraints` BEFORE UPDATE ON `client`

FOR EACH ROW BEGIN

IF NEW.mid != OLD.mid THEN

signal sqlstate '45000' set message\_text = 'Member id cannot be changed';

END IF;

END

-Το **mid,** εκτός απο κλειδί της **Client**, είναι και ξένο κλειδί που αναφέρεται στην **Member,** οπότε κάθε φορά που διαγράφεται ένα στοιχείο της **Member** πρέπει να διαγράφεται και το αντίστοιχο στοιχείο της **Client**, αν υπάρχει. Αυτό είναι ένας περιορισμός αναφοράς, και ελέγχεται μέσω χρήσης foreign key constraint στην phpmyadmin, που ορίζει ότι κατά τη διαγραφή στοιχείου από την **Member,** θα ακολουθείται cascaded deletion policy.

ALTER TABLE `client`

ADD CONSTRAINT `FK\_Client\_Member` FOREIGN KEY (`mid`) REFERENCES `member` (`mid`) ON DELETE CASCADE ON UPDATE CASCADE;

**Merchant**

Τα πράγματα εδώ είναι ακριβώς αντίστοιχα με τον **Client.**

ALTER TABLE `merchant`

ADD CONSTRAINT `FK\_Merchant\_Member` FOREIGN KEY (`mid`) REFERENCES `member` (`mid`) ON DELETE CASCADE ON UPDATE CASCADE;

**Order**

-Δεν επιτρέπεται ποτέ μια **Order** να έχει αρνητικό κόστος, **state** που δεν ανήκει στο {FULLY\_PAID, DELIVERED, AWAITING\_PAYMENT} ή **discount** που δεν ανήκει στο {0, 5, 10}. Επίσης δεν επιτρέπεται να αλλάζει το **oid** ή το **mid** μετά τη δημιουργία της παραγγελίας, ούτε να γίνεται παραγγελία αν υπάρχουν απλήρωτες παραγγελίες που έγιναν πάνω από 10 μέρες πριν. Τέλος, το ποσοστό έκπτωσης καθορίζεται στη βάση και ελέγχεται αν ο χρήστης διαθέτει το απαιτούμενο balance. Αυτά ελέγχονται με τα εξής δύο triggers:

CREATE TRIGGER `order\_t\_update\_constraints` BEFORE UPDATE ON `order\_t`

FOR EACH ROW BEGIN

IF NEW.oid != OLD.oid THEN

signal sqlstate '45000' set message\_text = 'Cannot change order id after creation of order';

END IF;

IF NEW.mid != OLD.mid THEN

signal sqlstate '45000' set message\_text = 'Cannot change member id referred to by order after creation of order';

END IF;

IF NEW.state != 'DELIVERED' AND NEW.state != 'AWAITING\_PAYMENT' AND NEW.state != 'FULLY\_PAID' THEN

signal sqlstate '45000' set message\_text = 'Invalid order state';

END IF;

IF NEW.cost < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative order cost';

END IF;

IF NEW.discount != 0 AND NEW.discount != 5 AND NEW.discount != 10 THEN

signal sqlstate '45000' set message\_text = 'Invalid order discount';

END IF;

END

CREATE TRIGGER `order\_t\_create\_constraints` BEFORE INSERT ON `order\_t`

FOR EACH ROW BEGIN

IF NEW.cost < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative order cost!';

END IF;

IF NEW.discount != 0 AND NEW.discount != 5 AND NEW.discount != 10 THEN

signal sqlstate '45000' set message\_text = 'Invalid discount in order!';

END IF;

IF NEW.state != 'FULLY\_PAID' AND NEW.state != 'AWAITING\_PAYMENT' THEN

signal sqlstate '45000' set message\_text = 'Invalid order state!';

END IF;

IF EXISTS(SELECT \* FROM order\_t o WHERE DATEDIFF(o.date, NEW.date)> 10 AND o.mid = NEW.mid AND o.state = 'AWAITING\_PAYMENT') THEN

signal sqlstate '45000' set message\_text = 'Member cannot order because of unpaid orders made more than 10 days ago!';

END IF;

UPDATE member SET debt = debt + NEW.cost;

IF NEW.mid in (SELECT mid FROM merchant) THEN

IF NEW.cost > 400 THEN

SET NEW.cost = NEW.cost\*90/100;

SET NEW.discount = 10;

ELSEIF NEW.cost > 280 THEN

SET NEW.cost = NEW.cost\*95/100;

SET NEW.discount = 5;

ELSE

SET NEW.discount = 0;

END IF;

END IF;

END

- Η **Order** έχει ξένο κλειδί το **mid**, άρα έχουμε περιορισμό αναφοράς (foreign key constraint στην phpmyadmin), και επιλέγουμε την εξής πολιτική: όταν ένα μέλος διαγράφεται, οι παραγγελίες του παραμένουν στο σύστημα ώστε να είναι διαθέσιμες οι πληροφορίες σχετικά με το πόσα κρασιά κάθε είδους έχουν πουληθεί αυτόν τον μήνα. Άρα, θέτουμε το **mid** της **Order** ως null.

ALTER TABLE `order\_t`

ADD CONSTRAINT `FK\_Order\_t\_Member` FOREIGN KEY (`mid`) REFERENCES `member` (`mid`) ON DELETE SET NULL;

**Wine**

-Ανά πάσα στιγμή απαγορεύεται ένα κρασί να έχει αρνητική τιμή λιανικής ή χονδρικής πώλησης, ενώ το **wid** δεν επιτρέπεται να αλλάζει. Υπάρχουν τα εξής δύο triggers στη βάση:

CREATE TRIGGER `wine\_domain\_constraints` BEFORE INSERT ON `wine`

FOR EACH ROW BEGIN

IF NEW.retailPrice <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative retail price!';

END IF;

IF NEW.wholesalePrice <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative wholesale price!';

END IF;

END

CREATE TRIGGER `wine\_update\_constraints` BEFORE UPDATE ON `wine`

FOR EACH ROW BEGIN

IF NEW.retailPrice <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative retail price!';

END IF;

IF NEW.wholesalePrice <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative wholesale price!';

END IF;

IF NEW.wid != OLD.wid THEN

signal sqlstate '45000' set message\_text = 'Wine id cannot be changed';

END IF;

END

**MostPopularWine**

-Το **bottlesThisMonth** δεν επιτρέπεται να είναι αρνητικό, ούτε και να αλλάζει μετά τη δημιουργία της πλειάδας. Υπάρχουν στη βάση τα εξής δύο triggers:

CREATE TRIGGER `mostpopularwines\_domain\_constraints` BEFORE INSERT ON `mostpopularwines`

FOR EACH ROW BEGIN

IF NEW.bottlesThisMonth <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative amount of bottles!';

END IF;

END

CREATE TRIGGER `mostpopularwines\_update\_constraints` BEFORE UPDATE ON `mostpopularwines`

FOR EACH ROW BEGIN

IF NEW.bottlesThisMonth != OLD.bottlesThisMonth THEN

signal sqlstate '45000' set message\_text = 'Can''t change number of bottles in mostpopularwines after creation';

END IF;

END

-Επίσης, το **wid** είναι και ξένο κλειδί στην **mostPopularWine**, άρα έχουμε έναν περιορισμό αναφοράς στον πίνακα **Wine.** Επιλέξαμε να μην σβήνεται ποτέ ένα κρασί από τον πίνακα **Wine** άπαξ και εισαχθεί στη βάση**,** άρα ακολουθήσαμε restriction policy στη διαγραφή.

ALTER TABLE `mostpopularwines`

ADD CONSTRAINT `FK\_MostPopularWines` FOREIGN KEY (`wid`) REFERENCES `wine` (`wid`);

**Variety**

Δεν έχει περιορισμούς ακεραιότητας.

**Transaction**

-Το **type** δεν μπορεί να πάρει τιμή διαφορετική των **PAYMENT, REFUND**, ενώ το **amount** δεν μπορεί να γίνει ποτέ αρνητικό. Επίσης, δεν πρέπει ποτέ μια νέα δοσοληψία μιας παραγγελίας να έχει ημερομηνία παλαιότερη από την ίδια την παραγγελία ή να ξεπερνάει το απλήρωτο/πληρωμένο μέρος της παραγγελίας αν είναι payment/refund αντίστοιχα. Το εξής trigger υπάρχουν στη βάση:

CREATE TRIGGER `transactions\_domain\_constraints` BEFORE INSERT ON `transactions`

FOR EACH ROW BEGIN

IF NEW.amount < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative transaction amount!';

END IF;

IF NEW.type != 'PAYMENT' AND NEW.type != 'REFUND' THEN

signal sqlstate '45000' set message\_text = 'Invalid transaction state!';

END IF;

IF (NEW.type = 'PAYMENT') THEN

IF ( (SELECT sum(balance) FROM member WHERE member.mid in (SELECT mid FROM order\_t WHERE order\_t.oid = NEW.oid) ) < NEW.amount ) THEN

signal sqlstate '45000' set message\_text = 'Insufficient balance for transaction!';

END IF;

END IF;

IF (NEW.type = 'REFUND' AND (SELECT sum(amount) FROM transactions t WHERE t.oid = NEW.oid AND type = 'PAYMENT') < NEW.amount) THEN

signal sqlstate '45000' set message\_text = 'Can''t refund more than paid part of order!';

END IF;

IF (NEW.type = 'PAYMENT' AND ( (SELECT sum(cost) FROM order\_t o WHERE o.oid = NEW.oid) - (SELECT sum(amount) FROM transactions t WHERE t.oid = NEW.oid AND t.type ='PAYMENT') + (SELECT sum(amount) FROM transactions t WHERE t.oid = NEW.oid AND t.type ='REFUND') ) < NEW.amount) THEN

signal sqlstate '45000' set message\_text = 'Attempted to pay more than remaining cost of order!';

END IF;

IF (NEW.type = 'PAYMENT') THEN

UPDATE member SET balance = balance - NEW.amount, debt = debt - NEW.amount, totalMoneySpent = totalMoneySpent + NEW.amount WHERE mid in (SELECT mid FROM order\_t WHERE oid = NEW.oid );

END IF;

IF (NEW.type = 'REFUND') THEN

UPDATE member SET balance = balance + NEW.amount, totalMoneySpent = totalMoneySpent - NEW.amount WHERE mid in (SELECT mid FROM order\_t WHERE oid = NEW.oid);

END IF;

END

-Το **oid** είναι ξένο κλειδί, άρα έχουμε έναν περιορισμό αναφοράς. Επιλέγουμε cascaded deletion policy, αφού όταν διαγράφεται μια παραγγελία θέλουμε να διαγράφονται και όλα τα σχετικά transactions.

ALTER TABLE `transactions`

ADD CONSTRAINT `FK\_Payment\_Order` FOREIGN KEY (`oid`) REFERENCES `order\_t` (`oid`) ON DELETE CASCADE;

**OrderConsistsOfWine**

**-**Το γνώρισμα **amount** δεν επιτρέπεται να πάρει αρνητική τιμή. Επίσης, δεν επιτρέπεται να αλλάζει η τιμή των **oid, wid** μετά τη δημιουργία μιας πλειάδας. Υπάρχουν τα εξής triggers:

CREATE TRIGGER `orderconsistsofwine\_domain\_constraints` BEFORE INSERT ON `orderconsistsofwine`

FOR EACH ROW BEGIN

IF NEW.amount <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative amount of bottles!';

END IF;

END

CREATE TRIGGER `orderconsistsofwine\_update\_constraints` BEFORE UPDATE ON `orderconsistsofwine`

FOR EACH ROW BEGIN

IF NEW.amount < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative amount of bottles!';

END IF;

IF NEW.amount > OLD.amount THEN

signal sqlstate '45000' set message\_text = 'Cannot increase order''s bottles after submission';

END IF;

IF NEW.oid != OLD.oid THEN

signal sqlstate '45000' set message\_text = 'Cannot change order id after creation of order';

END IF;

IF NEW.oid != OLD.oid THEN

signal sqlstate '45000' set message\_text = 'Cannot change order id after creation of order';

END IF;

IF NEW.wid != OLD.wid THEN

signal sqlstate '45000' set message\_text = 'Cannot change id of wine referred by order after creation of order';

END IF;

END

-Υπάρχουν επίσης δύο ξένα κλειδιά, **oid & wid**, άρα δύο περιορισμοί αναφοράς. Όταν διαγράφεται μια παραγγελία διαγράφονται και όλα τα σχετικά **OrderConsistsOfWine** άρα έχουμε cascaded deletion policy, ενώ εφόσον έχουμε την παραδοχή ότι κρασιά δεν θα διαγράφονται, έχουμε restrict deletion policy για το **wid.**

ALTER TABLE `orderconsistsofwine`

ADD CONSTRAINT `FK\_Wine` FOREIGN KEY (`wid`) REFERENCES `wine` (`wid`),

ADD CONSTRAINT `FK\_order` FOREIGN KEY (`oid`) REFERENCES `order\_t` (`oid`) ON DELETE CASCADE;

**WineMadeOfVariety**

-Έχουμε δύο ξένα κλειδιά, **wid & oid**, άρα δύο περιορισμούς αναφοράς. Επειδή ποικιλίες και κρασιά θεωρούμε ότι δεν θα διαγράφονται, βάζουμε restrict deletion policy.

ALTER TABLE `winemadeofvariety`

ADD CONSTRAINT `FK\_Variety` FOREIGN KEY (`vid`) REFERENCES `variety` (`vid`),

ADD CONSTRAINT `FK\_WineMadeOfVariety\_Wine` FOREIGN KEY (`wid`) REFERENCES `wine` (`wid`);

Ενότητα 4: Κανονικές Μορφές

Κεφάλαιο 4.1: Συναρτησιακές Εξαρτήσεις

**Member  
mid -> username, fname, lname, tel, address, city, country, debt, cardno, balance, password, totalMoneySpent**

**username->mid, fname, lname, tel, address, city, country, debt, cardno, balance, password, totalMoneySpent**

(αφού **mid, username** κλειδιά)

**MostPopularWines**

**wid->bottlesThisMonth**

(αφού **wid** κλειδί)

**OrderConsistsOfWine**

**oid wid -> amount**

(αφού {**oid, wid**} κλειδί)

**Order\_t**

**oid -> mid, date, cost, state**

(αφού **oid** κλειδί)

**Transaction**

**oid date -> type, amount**

αφού στην **Transaction** ο συνδυασμός των **oid, date** προέκυψε ως μερικό αναγνωριστικό (και επομένως δρα τρόπον τινά ως κλειδί)

**Variety**

**vid->name**

**name->vid**

αφού και τα 2 γνωρίσματα είναι κλειδιά

**Wine**

**wid -> retailPrice, wholesalePrice, winery, name, color, date, photo**

**retailPrice, wholesalePrice, winery, name, color, date -> wid, photo**

(αφού **wid** κλειδί και αφού δεν μπορούν να υπάρχουν στη βάση δύο κρασιά που διαφέρουν μόνο κατά το **wid**)

**WineMadeOfVariety**

Δεν υπάρχουν μη τετριμμένες συναρτησιακές εξαρτήσεις (αφού η σχέση έχει ως γνωρίσματα μόνο τα **wid, vid** των οποίων το σύνολο αποτελεί και το πρωτεύον κλειδί της σχέσης)

Κεφάλαιο 4.2: Καθορισμός Κλειδιών βάσει Συναρτησιακών Εξαρτήσεων

Λόγω της απλότητας των συναρτησιακών εξαρτήσεων, τα μοναδικά υποψήφια κλειδιά είναι τα όσα έχουν αναφερθεί στην ενότητα «Πρωτεύοντα κλειδιά», ενώ στο **Wine** υποψήφιο κλειδί είναι και το σύνολο { **retailPrice, wholesalePrice, winery, name, color, date**}. Επειδή λοιπόν δεν υπάρχουν άλλες (ουσιώδεις) συναρτησιακές εξαρτήσεις, ο αλγόριθμος εύρεσης υποψήφιων κλειδιών δεν δίνει άλλα κλειδιά εκτός των προαναφερθέντων.

Κεφάλαιο 4.3: Μετατροπή του Μοντέλου σε Τρίτη Κανονική Μορφή με διατήρηση των συναρτησιακών εξαρτήσεων και χωρίς απώλεια πληροφορίας

Μια σχέση R με σύνολο συναρτησιακών εξαρτήσεων F είναι σε 3η κανονική μορφή αν για κάθε μη τετριμμένη συναρτησιακή εξάρτηση Χ->Α που ανήκει στο F+ ισχύει ότι:

Χ είναι υπερκλειδί της R, ή

Α είναι ένα πρωτεύον γνώρισμα της R

Για κάθε μία από τις σχέσεις στη βάση μας, οι συναρτησιακές εξαρτήσεις που υπάρχουν είναι αρκούντως απλές και λίγες ώστε το κλείσιμο του εκάστοτε συνόλου τους να είναι το ίδιο το σύνολο. Επειδή λοιπόν αυτό το σύνολο είναι μόνο εξαρτήσεις της μορφής Χ->Α όπου Χ κλειδί της R, το Χ είναι και υπερκλειδί, άρα ικανοποιείται η πρώτη από τις προαναφερθείσες προϋποθέσεις ώστε η R να είναι σε 3η κανονική μορφή.

Άρα, όλες οι σχέσεις της βάσης είναι σε 3η κανονική μορφή, οπότε και το σχήμα της βάσης δεδομένων μας είναι ήδη σε 3η κανονική μορφή και δεν χρειάζεται αλλαγές.

Ενότητα 5: S.Q.L.

Κεφάλαιο 5.1: Περιγραφή των Ερωτήσεων προς τη Βάση Δεδομένων

Στον κώδικά μας έχουμε ενσωματώσει τον κώδικα των ερωτήσεων προς τη βάση σε ξεχωριστά αρχεία, ανάλογα με τη σχέση που απευθύνονται.

5.1.1: MemberDB

Get All Members (basically never used except for testing)

SELECT \* FROM Member;

Get Bad Members

SELECT \* FROM Member WHERE debt > 0 ORDER BY debt DESC;

Get Good Members

SELECT \* FROM Member WHERE debt = 0 ORDER BY totalMoneySpent DESC;

Get Member by username

SELECT \* FROM Member WHERE username = '<username>';

Get Member by username and password (used for logging in)

SELECT mid FROM Member WHERE username = '<username>' AND password = '<password>';

Get mid from username (useful when we need to pass user info as argument and only have username)

SELECT mid FROM Member WHERE username = '<username>';

Get an order and its cost from particular member id

SELECT oid,cost from Order\_t WHERE mid = '<mid>';

Check if a username is taken in registration (if empty result it’s not taken)

SELECT username FROM Member WHERE username = '<username'>;

Check if a user is a merchant (by username)

SELECT Member.mid FROM Member, Merchant WHERE Member.mid = Merchant.mid AND Member.username = '<username>';

Get member balance (to check if the member has enough money to make an order)

SELECT balance FROM Member WHERE mid = '<mid>';

Check if there is an order by a member that hasn’t been fully paid and is more than 10 days old

SELECT o.oid FROM Order\_t o where o.mid = '<mid>' and TIMESTAMPDIFF(DAY, o.date, '<date>') >= 10;

Get Member debt (to display it, or check if the member is ready to delete)

SELECT debt FROM Member where mid = '<mid>';

Get state of orders of a member (check if they’re delivered, or display it)

SELECT state FROM Order\_t where mid = '<mid>';

Insert a new member in the database

INSERT INTO Member(username, password, fname, lname, tel, address, city, country, debt, cardno, balance) VALUES ('<username>', '<password>', '<fname>', '<lname>', '<telephone>', '<address>', '<city>', '<country>', 0, '<cardno>', 0);

Update a member in the database

UPDATE Member SET password = '<password>', fname = '<fname>', lname = '<lname>', tel = '<telephone>', address = '<address>', city = '<city>', country = '<country>', debt = '<debt>', cardno = '<cardno>' WHERE username = '<username>';

Delete a member from the database

DELETE FROM Member WHERE username = '<username>';

Update total money spent in a member (when paying or being refunded an order)

UPDATE Member SET totalMoneySpent = totalMoneySpent <+/-> <amount> WHERE mid = '<mid>';

Update member balance

UPDATE Member SET balance = balance <+/-> <amount> WHERE mid = '<mid>';

5.1.2: OrderDB

Get all orders (only used for testing)

SELECT \* FROM Order\_t;

Get all orders of a member (to display them or examine them)

SELECT \* FROM Order\_t where mid = '<member\_id>';

Get all orders at a particular date (only used for testing)

SELECT \* FROM Order\_t where date = '<date>';

Get all orders on a particular state (only used for testing)

SELECT \* FROM Order\_t where state = '<state>';

Get orders in a price interval [min, max] (only used for testing)

SELECT \* FROM Order\_t where cost >= '<min>' and cost <= '<max>';

Get an order by its id (used in many places where a particular order needs to be examined)

SELECT \* FROM Order\_t WHERE oid = '<order\_id>';

Get the last inserted order (used to return an order id when inserting it)

SELECT oid FROM Order\_t WHERE oid = LAST\_INSERT\_ID();

Get order cost (to display or update it)

SELECT cost FROM Order\_t WHERE oid = '<oid>';

Get discount of an order (to display or apply it to total cost)

SELECT discount FROM Order\_t where oid = '<oid>';

Get the cost and member id of an order (to update a member debt or balance after making a transaction)

SELECT cost, mid FROM Order\_t WHERE oid = '<oid>';

Add a new order

INSERT INTO Order\_t(mid,date,cost,state) VALUES('<member\_id>','<date>', '<cost>', '<state>');

Update the state of an order

UPDATE Order\_t SET state = '<state>' WHERE oid = '<oid>';

Update order cost

UPDATE Order\_t SET cost = '<cost>' WHERE oid = '<oid>';

Update order discount

UPDATE Order\_t SET discount = '<discount>' WHERE oid = '<oid>';

5.1.3: OrderConsistsOfWineDB

Get amounts and wines in an order

SELECT wid, amount FROM OrderConsistsOfWine WHERE oid = '<oid>'

Get amount of a wine in an order

SELECT amount FROM OrderConsistsOfWine WHERE oid = '<oid>' AND wid = '<wid>'

Add a new wine in an order

INSERT INTO OrderConsistsOfWine(oid,wid,amount) VALUES('<oid>',’<wid>', '<amount>');

Update a wine amount in an order

UPDATE OrderConsistsOfWine SET amount = <newamount> WHERE oid = '<oid>' AND wid = '<wid>';

Delete a wine from an order

DELETE FROM OrderConsistsOfWine WHERE oid = '<oid>' AND wid = '<wid>';

5.1.4: WineDB

Get wine id after inserting a wine

SELECT wid FROM wine WHERE wid = LAST\_INSERT\_ID();

Get all wines (only used for testing)

SELECT \* FROM Wine;

Get a wine by id (used in basket to display wine info, in orders, in filtering, etc…)

SELECT \* FROM Wine where wid = <wid>;

Get a wine by name (only used for testing)

SELECT \* FROM Wine where name = '<wineName>;

Get a wine by color (only used for testing)

SELECT \* FROM Wine where color = '<wineColor>'

Get a wine by winery (only used for testing)

SELECT \* FROM Wine where winery = '<winery>';

Get wines in price interval (only used for testing)

SELECT \* FROM Wine where retailPrice >= '<lowerBound>' and retailPrice <= '<upperBound>';

SELECT \* FROM Wine where wholesalePrice >= '<lowerBound>' and wholesalePrice <= '<upperBound>';

Get wines in date interval (only used for testing)

SELECT \* FROM Wine where date >= '<lowerBound>' and date <= '<upperBound>';

Get wines by variety (only used for testing)  
SELECT w.wid, w.price, w.winery, w.name, w.color, w.date, w.photo FROM Wine w, Variety v, WineMadeOfVariety wv where w.wid = wv.wid and v.vid = wv.vid and v.name = '<wineVariety>';

Get varieties of wine (only used for testing)  
SELECT v.vid, v.name FROM Variety v, WineMadeOfVariety wv WHERE wv.wid = ‘<wineId>’ and v.vid = wv.vid;

Get wines join varieties (used in filtering)  
SELECT v.vid, v.name, wv.wid FROM Variety v, WineMadeOfVariety wv WHERE v.vid = wv.vid;

Get best wines by variety (used in filtering best wines)  
SELECT w.\*, p.bottlesThisMonth FROM Wine w, mostpopularwines p, Variety v, winemadeofvariety wv where w.wid = p.wid and w.wid = wv.wid and wv.vid = v.vid and v.name = '<wineVariety>' ORDER BY p.bottlesThisMonth DESC LIMIT 10;

Get best wines by Winery (used in filtering best wines)  
SELECT w.\*, p.bottlesThisMonth FROM Wine w, mostpopularwines p where w.wid = p.wid and w.winery = '<winery>' ORDER BY p.bottlesThisMonth DESC LIMIT 10;

Get best wines by color (used in filtering best wines)  
SELECT w.\*, p.bottlesThisMonth FROM Wine w, mostpopularwines p where w.wid = p.wid and w.color = '<wineColor>' ORDER BY p.bottlesThisMonth DESC LIMIT 10;

Get best wines by date (used in filtering best wines)  
SELECT w.\*, p.bottlesThisMonth FROM Wine w, mostpopularwines p where w.wid = p.wid and w.date = '<date>' ORDER BY p.bottlesThisMonth DESC LIMIT 10;

Get best wines with no filter (used in best wine display)  
SELECT w.\*, p.bottlesThisMonth FROM Wine w, mostpopularwines p where w.wid = p.wid ORDER BY p.bottlesThisMonth DESC LIMIT 10;

Get wines containing a substring in their name

SELECT \* FROM Wine WHERE name LIKE '%<substr>%';

Get all wines that fulfill the filtering (used in filtering)  
SELECT DISTINCT Wine.wid, Wine.color, Wine.date, Wine.photo, Wine.name, Wine.winery, Wine.retailPrice, Wine.wholesalePrice FROM Wine, WineMadeOfVariety, Variety WHERE Wine.name LIKE '%<substr>%’

In this query we add according to the chosen filters:  
-and color = '<color>'

-and winery = '<winery>'  
-and Variety.name = '<variety>' and WineMadeOfVariety.wid = Wine.wid and WineMadeOfVariety.vid = Variety.vid

-and wine.date >= '<dateLower>'

-and wine.date <= '<dateUpper>'

-and wine.<retail/wholesaleprice> >= '<priceLower>'

-and wine.<retail/wholesaleprice> <= '<priceUpper>'

At the end we add the ;

Add a new wine

INSERT INTO wine(retailPrice, wholesalePrice, winery, name, color, date, photo) VALUES ('<retailPrice>', '<wholesalePrice>', '<winery>', '<name>', '<color>', '<date>', '<photo>');

Add wine variety

INSERT INTO winemadeofvariety(wid,vid) VALUES('<wid>','<vid>');

5.1.5: VarietyDB

Get variety by name (used when getting wines of a variety)

SELECT \* FROM variety WHERE name = '<name>';

Add a new variety

INSERT INTO variety(name) VALUES('<name>);

5.1.6: TransactionsDB

Get sum of payments in an order (to find how much of this order has been paid)

SELECT sum(amount) as total FROM transactions where oid = '<oid>' and type = 'PAYMENT';

Get sum of refunds in an order (to find how much of this order has been refunded)

SELECT sum(amount) as total FROM transactions where oid = '<oid>' and type = 'REFUND';

Get all transactions (used only for testing)

SELECT \* FROM transactions;

Get transactions by order id (to display transactions of an order)

SELECT \* FROM transactions where oid = '<oid>';

Get transactions by date interval (to filter transactions in an interval)

SELECT \* FROM transactions where date >= '<dateMin>' and date <= <dateMax>;

Get transactions by order and date (only used for testing)

SELECT \* FROM transactions where oid = '<oid>' and date = '<date>';

Get transactions in price interval (only used for testing)

SELECT \* FROM transactions where amount >= '<min>' and cost <= '<max>';

Get transactions of a member (used to display transactions of a member)

SELECT transactions.oid, type, amount, transactions.date FROM transactions, order\_t WHERE mid = '<mid>' AND order\_t.oid = transactions.oid;

Get transactions of a member in a date interval (used for filtering transactions of a member by date and time)

SELECT transactions.oid, type, amount, transactions.date FROM transactions, order\_t WHERE mid = '<mid>' AND order\_t.oid = transactions.oid AND transactions.date >= '<dateMin>' AND transactions.date <= '<dateMax>';

Add a new transaction

INSERT INTO transactions(oid,type,amount,date) VALUES('<oid>','<type>','<amount>', '<date>');

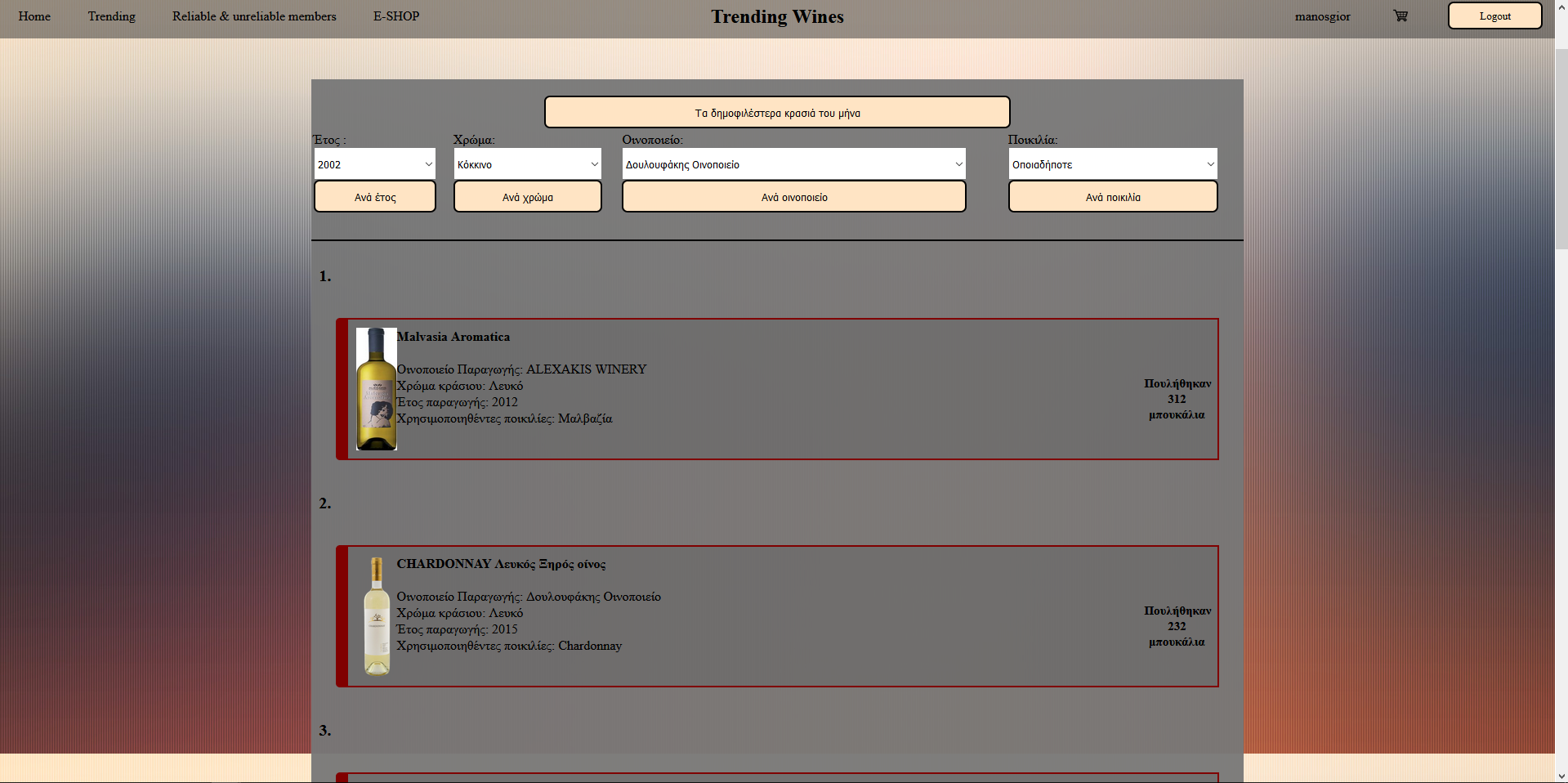
Ενότητα 6: Εγχειρίδιο Χρήσης του Συστήματος

Κεφάλαιο 6.1: Επιτρεπτές λειτουργίες χωρίς σύνδεση (login)

6.1.1: Δημοφιλή Κρασιά (Trending)

Το ηλεκτρονικό κατάστημα εκδίδει τα δημοφιλέστερα κρασιά του προηγούμενου μήνα, δηλαδή τα 10 κρασιά με τις περισσότερες πωλήσεις στον μήνα που πέρασε.

Ο επισκέπτης του ηλεκτρονικού καταστήματος έχει την δυνατότητα να δει αυτή την λίστα πατώντας το κουμπί “**Trending**” από το κεντρικό μενού της ιστοσελίδας. Επίσης μπορεί να επιλέξει φίλτρα με βάση κάποιο κριτήριο όπως είναι το χρώμα των κρασιών, η ποικιλία από την οποία παράχθηκαν, το οινοποιείο από το οποίο παρασκευάστηκαν καθώς και η χρονολογία στην οποία ωρίμασαν και τέθηκαν έτοιμα προς πώληση.

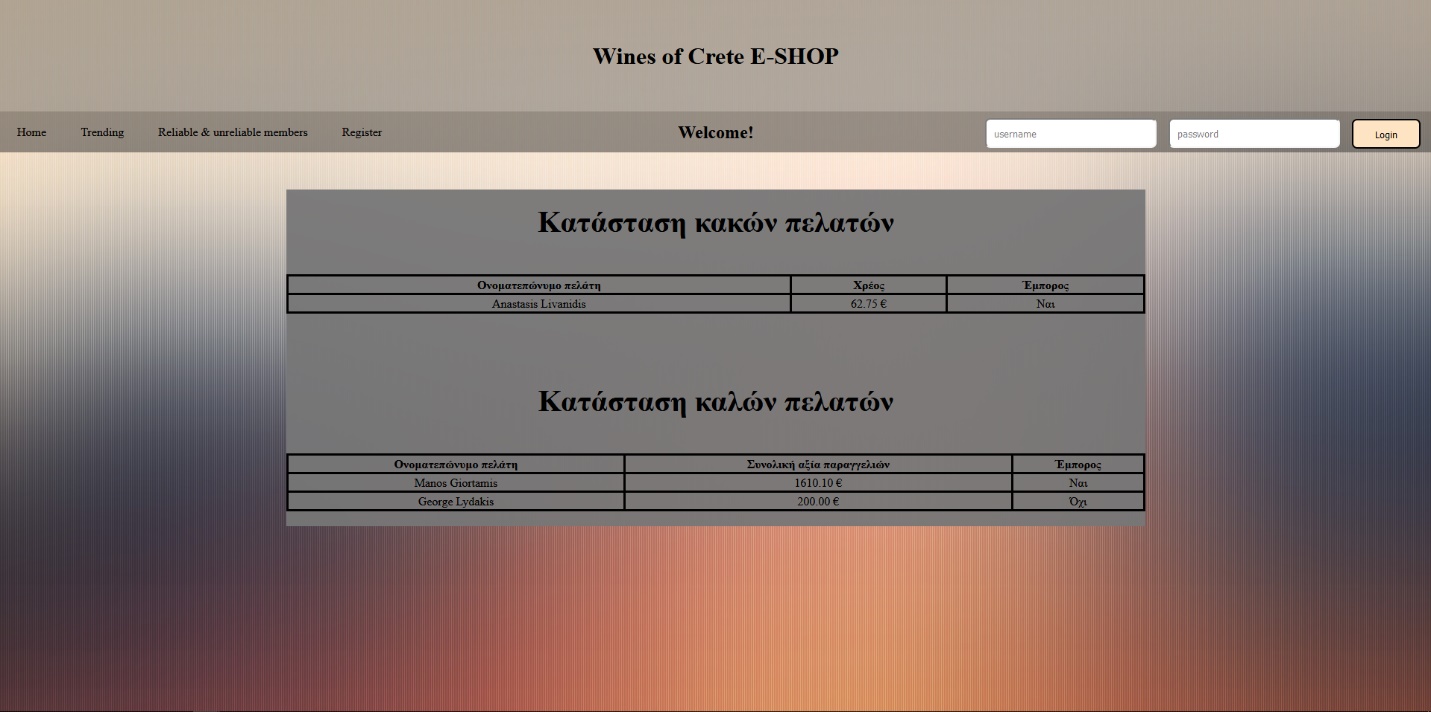


6.1.2: Κατάσταση Καλών και Κακών Πελατών (Reliable & unreliable members)

Το ηλεκτρονικό κατάστημα εκδίδει μια κατάσταση καλών πελατών, δηλαδή πελατών που δεν έχουν χρέος, ταξινομημένη σύμφωνα με την συνολική αξία των παραγγελιών του κάθε πελάτη.

Αντίστοιχα, εκδίδει την κατάσταση των κακών πελατών, δηλαδή πελατών που έχουν χρέος προς το ηλεκτρονικό κατάστημα, ταξινομημένη σύμφωνα με το χρέος τους.

Ο επισκέπτης του ηλεκτρονικού καταστήματος έχει πρόσβαση στις 2 καταστάσεις πατώντας το κουμπί “**Reliable & unreliable members** ” από το κεντρικό μενού της ιστοσελίδας.



Κεφάλαιο 6.2: Εγγραφή Χρήστη (Register)

6.2.1: Όνομα Χρήστη (username)

Το όνομα χρήστη αποτελεί ένα από τα δύο πεδία που χρειάζονται για την σύνδεση (login) στο ηλεκτρονικό κατάστημα. Ο χρήστης πρέπει να το θυμάται για να μπορέσει να συνδεθεί μετά την εγγραφή του. Το όνομα χρήστη πρέπει να είναι μοναδικό, δηλαδή δεν πρέπει να ταυτίζεται με το όνομα χρήστη κάποιου άλλου πελάτη. Επίσης πρέπει να αποτελείται από 8 έως 30 χαρακτήρες με επιτρεπτούς χαρακτήρες μόνο αυτούς του λατινικού αλφαβήτου (a-z, A-Z).

6.2.2: Κωδικός πρόσβασης (password)

Ο κωδικός πρόσβασης αποτελεί το δεύτερο πεδίο που χρειάζεται για την σύνδεση στο ηλεκτρονικό κατάστημα. Όπως και με το username, ο χρήστης πρέπει να θυμάται τον κωδικό του ώστε να μπορεί να συνδέεται στον λογαριασμό του στο ηλεκτρονικό κατάστημα. Ο κωδικός πρόσβασης πρέπει να αποτελείται από 8 έως 30 χαρακτήρες, με υποχρεωτική παρουσία τουλάχιστον ενός χαρακτήρα του λατινικού αλφαβήτου, τουλάχιστον ενός αριθμού (0-9) και τουλάχιστον ενός ειδικού χαρακτήρα (!@#$ κ.λ.π.).

6.2.3: Ονοματεπώνυμο (First & Last name)

Το ονοματεπώνυμο του χρήστη. 4 έως 30 χαρακτήρες.

6.2.4: Τηλέφωνο (Telephone)

Το τηλέφωνο του χρήστη. Πρέπει να είναι ακριβώς 10 χαρακτήρες από το 0 έως το 9, όπως ορίζεται για ένα έγκυρο τηλεφωνικό αριθμό στην Ελλάδα.

6.2.5: Αριθμός Πιστωτικής Κάρτας (card number)

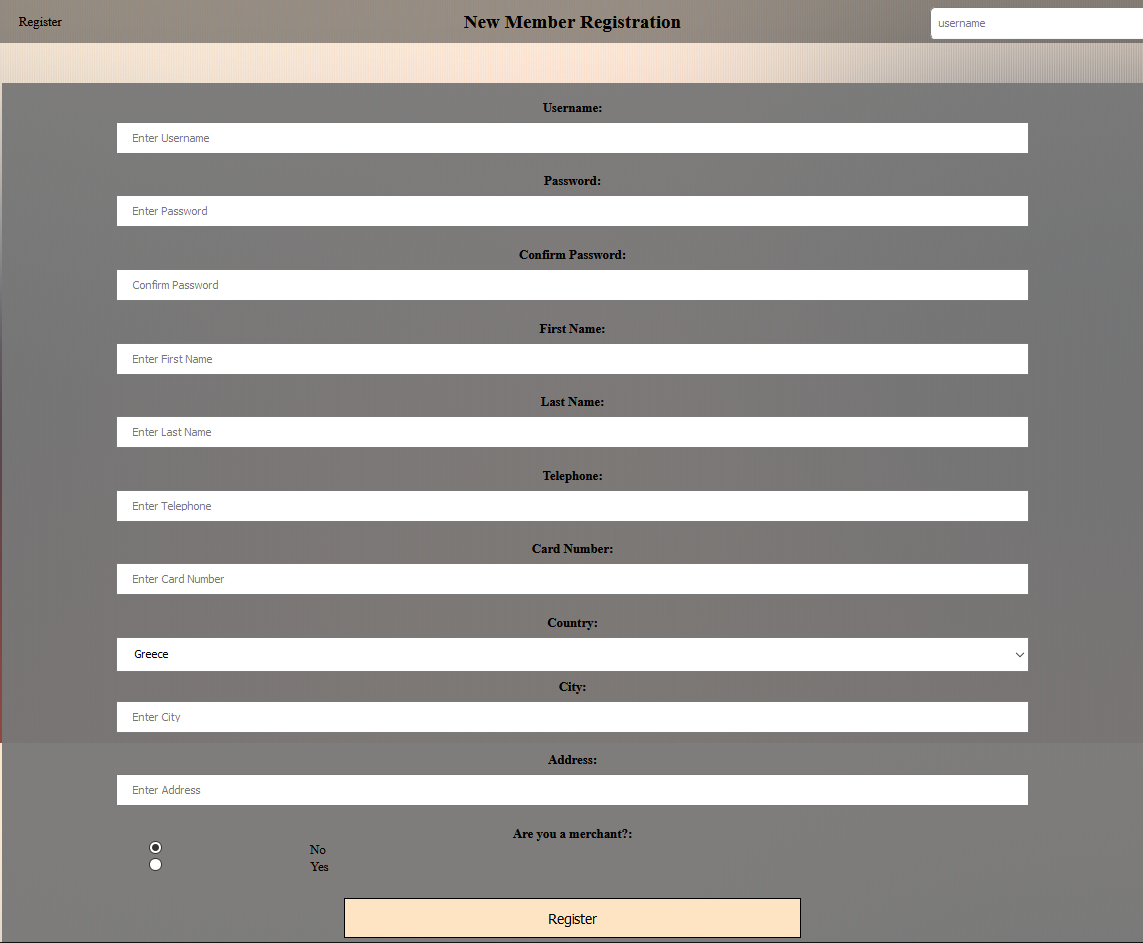
Ο αριθμός της πιστωτικής κάρτας του χρήστη. Πρέπει να είναι της μορφής xxxx-xxxx-xxxx-xxxx δηλαδή ακριβώς 16 χαρακτήρες από το 0 έως το 9, χωρισμένοι ανά 4 με παύλα (“-“), όπως ορίζεται για ένα έγκυρο αριθμό πιστωτικής κάρτας διεθνώς.

6.2.6: Διεύθυνση (country/city/address)

Η χώρα, πόλη και διεύθυνση του χρήστη αντίστοιχα.

6.2.7: Έλεγχος Εμπόρων (Are you a merchant ?)

Ο χρήστης επιλέγει “Yes” αν είναι έμπορος, διαφορετικά “No”.



Κεφάλαιο 6.3: Λειτουργίες σε σύνδεση (logged in)

6.3.1: Περί session και cookie

Στο session κρατείται πληροφορία όπως η κατάσταση σύνδεσης του χρήστη (δηλαδή αν έχει κάνει login στην ιστοσελίδα) καθώς και το καλάθι του, με σκοπό να μην χρειάζεται επανασύνδεση ή γέμισμα του καλαθιού από την αρχή σε περίπτωση που ο χρήστης αλλάξει σελίδα, κλείσει την καρτέλα ή ακόμα και τον browser. Οι πληροφορίες αυτές διατηρούνται για 30 μέρες στον browser. Ο χρήστης πρέπει να έχει ενεργοποιήσει τα cookies στον browser του για να έχει αυτή λειτουργία.

6.3.2: E-SHOP

Πατώντας το κουμπί “**E-SHOP**” στο κεντρικό μενού της ιστοσελίδας ο χρήστης επισκέπτεται το ηλεκτρονικό κατάστημα. Εκεί βρίσκονται όλα τα κρασιά που είναι διαθέσιμα προς πώληση. Βλέποντας κάποιο που επιθυμεί να αγοράσει, αρκεί να πληκτρολογήσει την ποσότητα που θέλει στο πεδίο «ποσότητα» και μετά να πατήσει «**Προσθήκη στο καλάθι**».

Το e-shop παρέχει την δυνατότητα αναζήτησης στα κρασιά με βάση ενός ή περισσότερων φίλτρων αναζήτησης.

**Όνομα :** Γίνεται αναζήτηση με βάση το όνομα του κρασιού που πληκτρολογεί ο χρήστης.

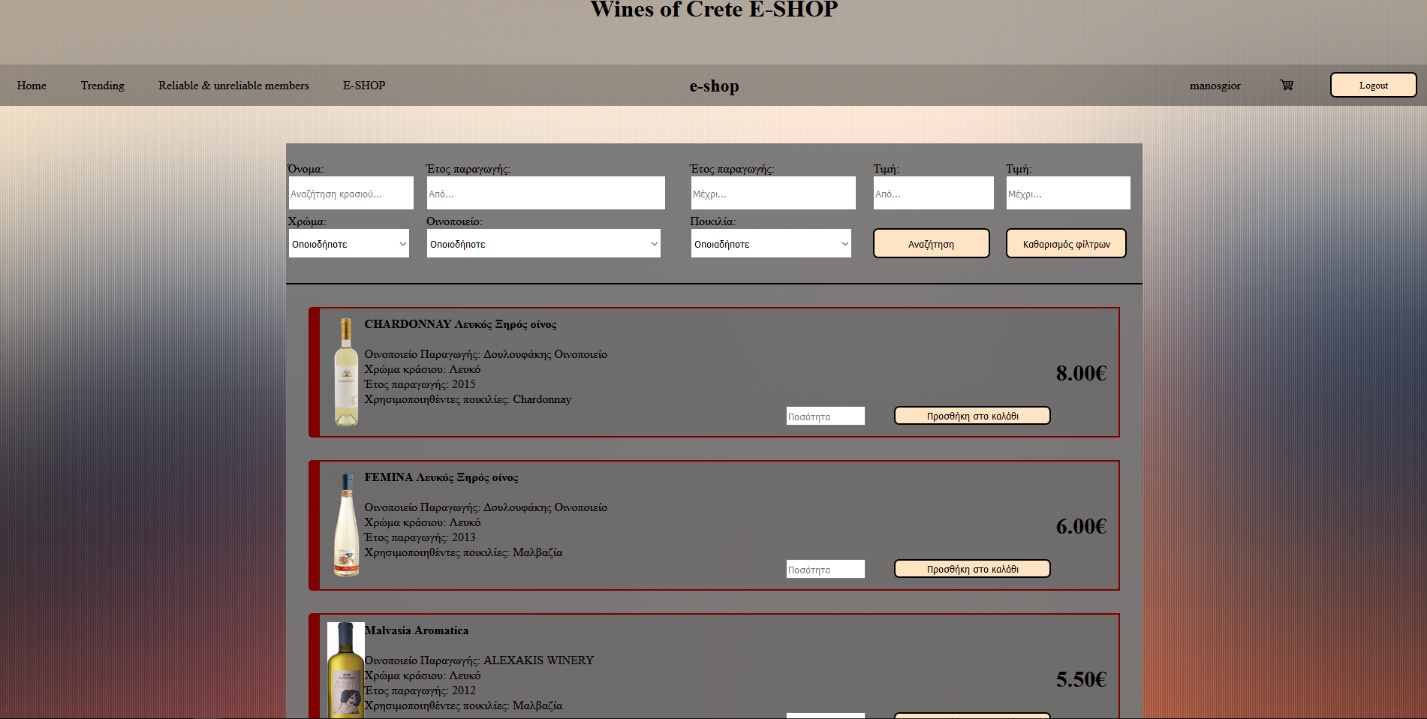
**Έτος παραγωγής :** Γίνεται αναζήτηση με βάση το εύρος του έτους παραγωγής που επιλέγει ο χρήστης.

**Τιμή :** Αντίστοιχα, με βάση το εύρος της τιμής που επιλέγει ο χρήστης.

**Χρώμα :** Αντίστοιχα, με βάση το χρώμα του κρασιού που επιλέγει ο χρήστης.

**Ποικιλία :** Τέλος, με βάση την ποικιλία που χρησιμοποιήθηκε για να παραχθεί το κρασί.

Αφού επιλέξει τα φίλτρα που θέλει, ο χρήστης αρκεί να πατήσει «**Αναζήτηση**». Εάν θέλει να αφαιρέσει τα φίλτρα για να δει όλα τα κρασιά ή να προσθέσει φίλτρα εκ νέου, αρκεί να πατήσει «**Καθαρισμός φίλτρων**».



6.3.3: Καλάθι Αγορών

Στο καλάθι αγορών αποθηκεύονται τα προϊόντα που επέλεξε να αγοράσει ο χρήστης. Στο καλάθι εμφανίζεται η λίστα με τα προϊόντα. Για κάθε προϊόν εμφανίζεται η ποσότητα που επιλέχθηκε, η τιμή και το όνομα του προϊόντος. Εάν θέλει ο χρήστης να αλλάξει την ποσότητα ενός κρασιού, πρέπει να πληκτρολογήσει την νέα ποσότητα στο άσπρο πλαίσιο κάτω από το όνομα του κρασιού και κατόπιν να πατήσει το κουμπί “**Change**”. Εάν θέλει να αφαιρέσει ένα κρασί από το καλάθι, πρέπει να πατήσει το κόκκινο “**X**” δίπλα από το κουμπί “**Change**”. Εάν θέλει να αφαιρέσει όλα τα προϊόντα από το καλάθι, πρέπει να πατήσει το κουμπί “**Clear**” στην κάτω δεξιά γωνία του καλαθιού.

Τέλος, εάν θέλει να ολοκληρώσει την παραγγελία του, πρέπει να πληκτρολογήσει μια έγκυρη τιμή στο άσπρο πλαίσιο “Pay” κάτω από το “Total: x €” και κατόπιν να πατήσει το κουμπί “**Buy**”.

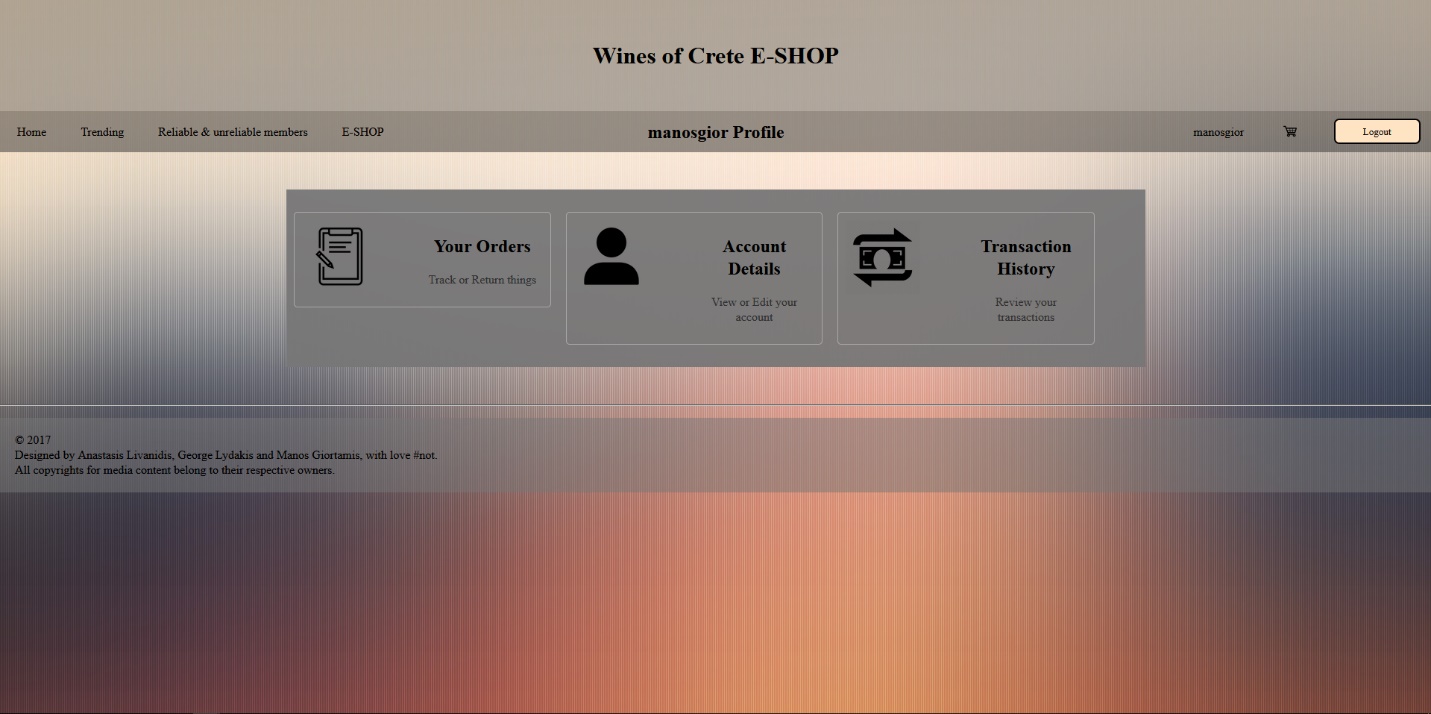
Φυσικά, για να μπορέσει να ολοκληρώσει την παραγγελία του, ο χρήστης πρέπει να έχει αρκετό υπόλοιπο στον λογαριασμό του, καθώς και να μην έχει αφήσει (μερικώς) απλήρωτες παραγγελίες που είναι τουλάχιστον 10 ημέρες πριν την ημερομηνία της τρέχουσας παραγγελίας.

Εάν ο χρήστης είναι έμπορος, πρέπει η παραγγελία του να περιέχει τουλάχιστον 6 μπουκάλια από τουλάχιστον 3 διαφορετικά κρασιά.



6.3.4: Πληροφορίες Λογαριασμού

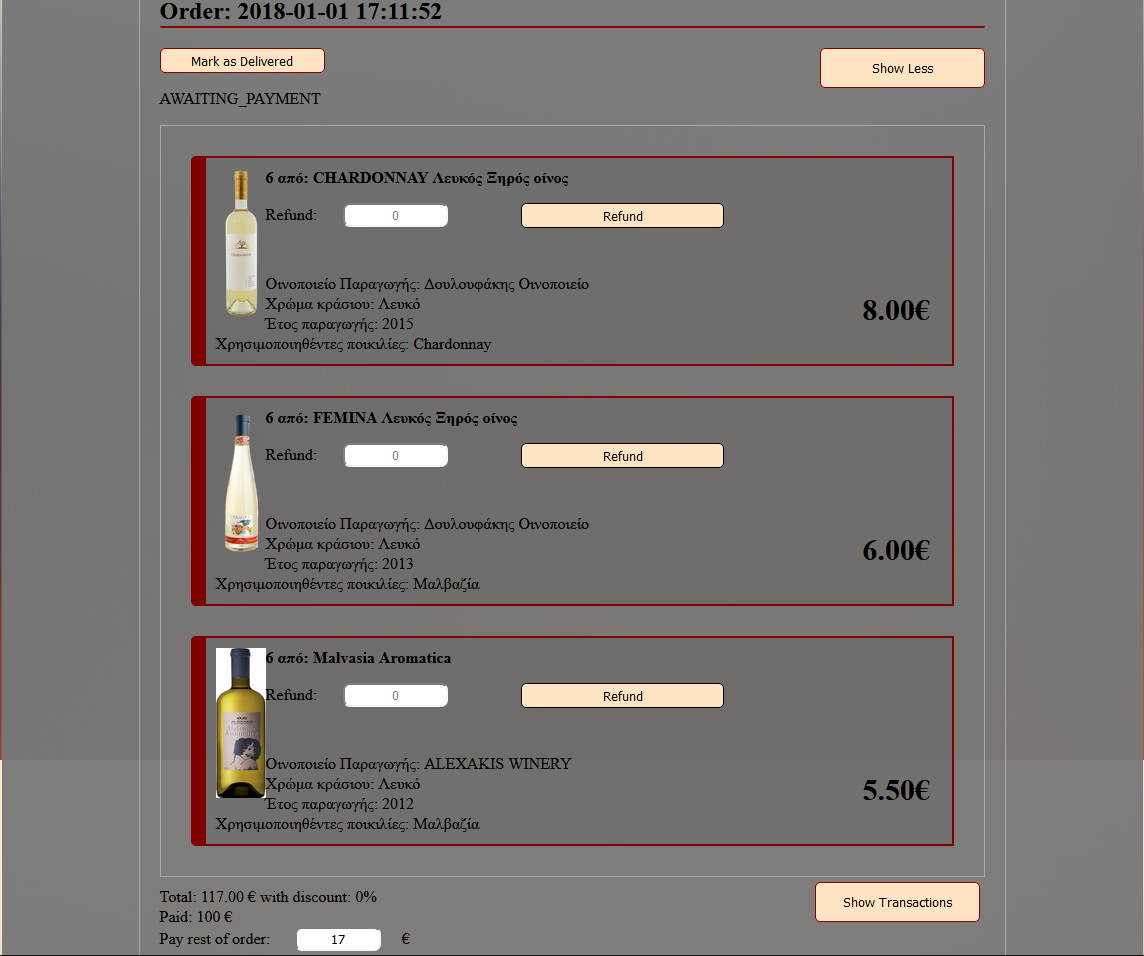
Για να εμφανιστεί το μενού με τις πληροφορίες λογαριασμού, πρέπει να πατηθεί το **username** του χρήστη στο κεντρικό μενού της ιστοσελίδας. Η σελίδα που θα εμφανιστεί είναι η εξής :



6.3.4.1 Ιστορικό Παραγγελιών (Your orders)

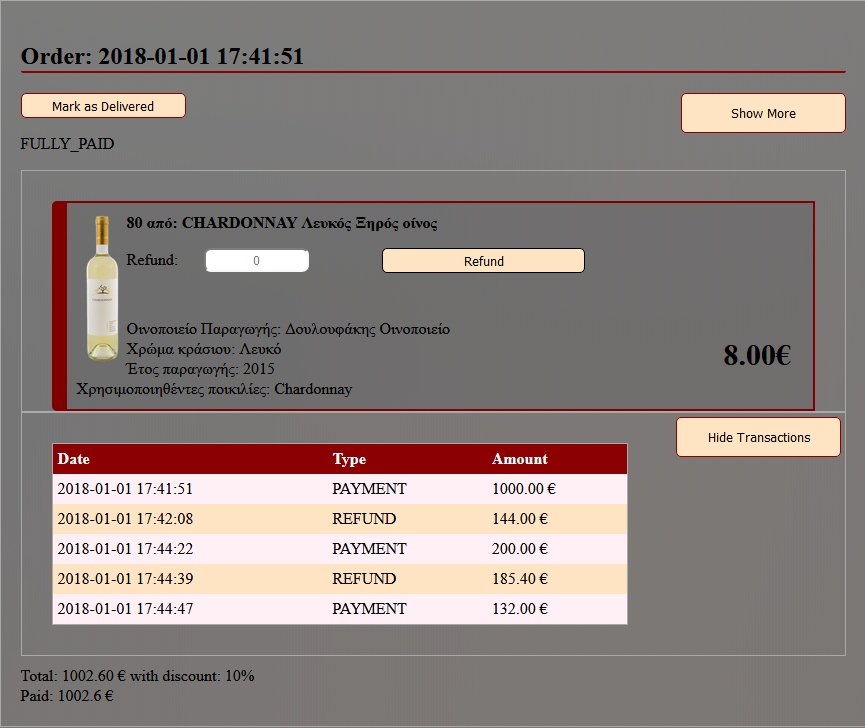
Σε αυτή την σελίδα εμφανίζεται το ιστορικό των παραγγελιών του χρήστη. Για κάθε παραγγελία που έχει κάνει, εμφανίζεται μια καταχώρηση που περιέχει την ημερομηνία και την ώρα της παραγγελίας, την κατάσταση της (“AWAITING\_PAYMENT”, ”FULLY\_PAID”,”DELIVERED”), μία λίστα με τα προϊόντα τα οποία περιέχει, το συνολικό κόστος της παραγγελίας, το αν και πόση έκπτωση έχει, καθώς και το ποσό που έχει εξοφλήσει ο χρήστης. Πατώντας το κουμπί “**Show More**” εμφανίζονται όλα τα προϊόντα της παραγγελίας καθώς και η ποσότητα από το κάθε ένα. Αντίστοιχα με το “**Show Less**” κρύβονται και εμφανίζεται μόνο το πρώτο.

Ο χρήστης έχει την δυνατότητα να επιστέψει προϊόντα πίσω στο κατάστημα. Για να το κάνει αυτό, επιλέγει την ποσότητα που θέλει να επιστρέψει πληκτρολογώντας το νούμερο στο άσπρο πλαίσιο δίπλα από το κουμπί “**Refund**” και έπειτα πατάει το προαναφερθέν κουμπί. Προφανώς δεν μπορεί να επιστρέψει περισσότερα μπουκάλια από όσα είχε αγοράσει, δεν μπορεί η αξία των επιστρεφόμενων μπουκαλιών να είναι μεγαλύτερη από το ποσό που έχει πληρώσει μέχρι τώρα για την παραγγελία, και αν είναι έμπορος, δεν μπορεί η παραγγελία μετά την επιστροφή να έχει λιγότερα από 6 μπουκάλια από λιγότερα από 3 διαφορετικά κρασιά.



Πατώντας το κουμπί “**Show Transactions**” εμφανίζονται όλες οι δοσοληψίες της συγκεκριμένης παραγγελίας. Πιο συγκεκριμένα εμφανίζεται η ακριβής ημερομηνία και ώρα της δοσοληψίας, ο τύπος της και το πόσο της.

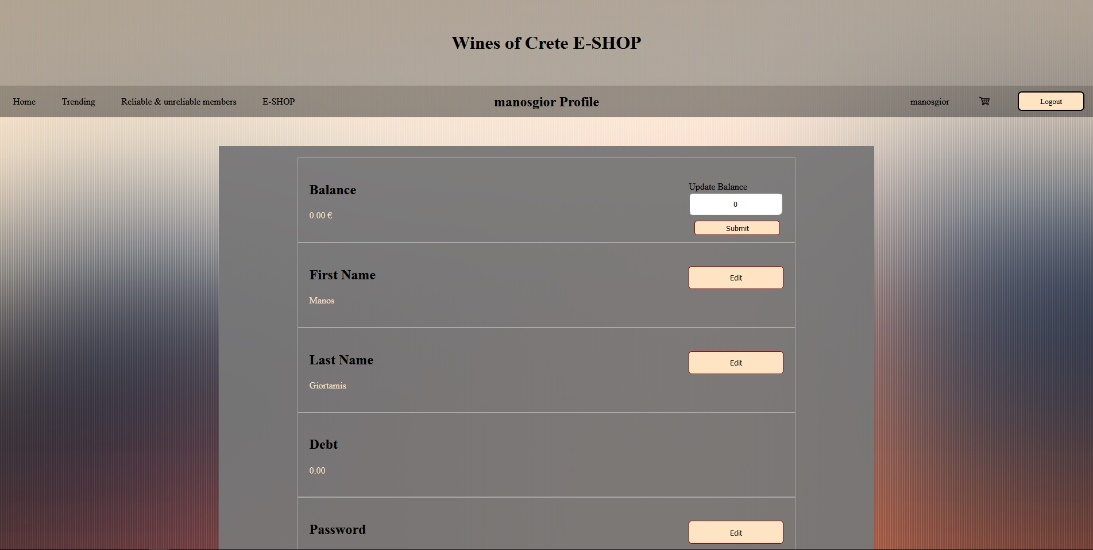
Τέλος, πατώντας το κουμπί “Mark as Delivered” ο χρήστης έχει την δυνατότητα να ενημερώσει το κατάστημα για την παραλαβή της παραγγελίας του. Δηλαδή, μετά το πάτημα του κουμπιού, η κατάσταση της παραγγελίας γίνεται “DELIVERED” και το κατάστημα θεωρεί ότι ο χρήστης έλαβε την παραγγελία του.



6.3.4.2 Γενικές Πληροφορίες (Account Details)

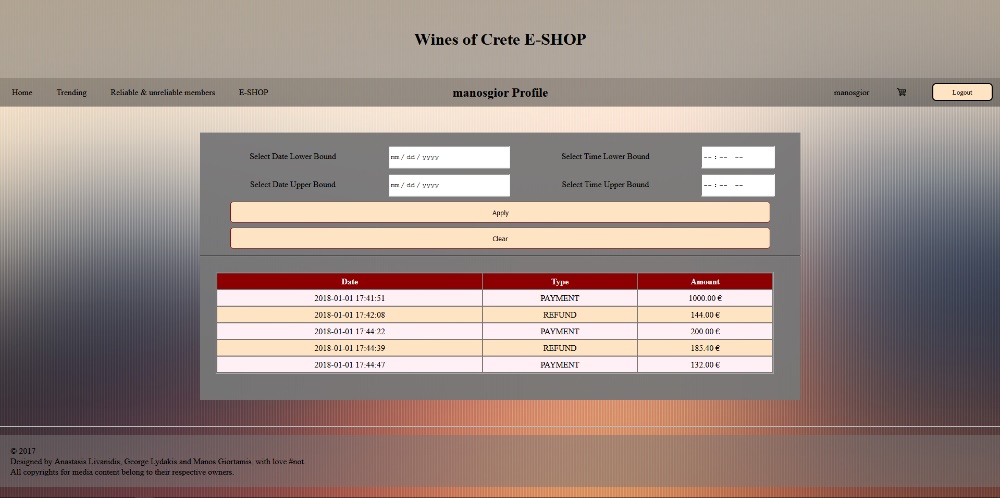
Σε αυτή την σελίδα εμφανίζονται οι πληροφορίες του χρήστη. Οι πληροφορίες είναι αυτές που εισήχθησαν από την χρήστη κατά την εγγραφή του. Επιπλέον, ο χρήστης βλέπει το υπόλοιπο του λογαριασμού του στο πεδίο “Balance” έχοντας την δυνατότητα να το αυξήσει πληκτρολογώντας το ποσό που θέλει να προσθέσει στο πλαίσιο “Update Balance” και πατώντας το κουμπί “**Submit**”.

Επίσης ο χρήστης έχει την δυνατότητα να αλλάξει τις πληροφορίες του πατώντας το κουμπί “**Edit**” δίπλα από την πληροφορία την οποία θέλει να αλλάξει.



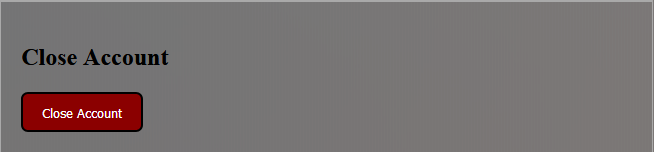
6.3.4.3 Ιστορικό Δοσοληψιών (Transaction History)

Σε αυτή την σελίδα εμφανίζεται το ιστορικό όλων των δοσοληψιών του χρήστη για όλες τις παραγγελίες που έχει κάνει. Ο χρήστης μπορεί να αναζητήσει δοσοληψίες επιλέγοντας ένα εύρος ημερομηνίας ή/και ώρας. Πατώντας πάνω στο άσπρο πλαίσιο “mm/dd/yyyy” εμφανίζεται ένα ημερολόγιο από το οποίο μπορεί ο χρήστης να επιλέξει την ακριβή ημερομηνία που θέλει. Πατώντας στο “Select Time Lower/Upper Bound” μπορεί να πληκτρολογήσει την ακριβή ώρα που θέλει. Τέλος, πατώντας το κουμπί “**Apply**” εφαρμόζει το φίλτρο που έθεσε και εν τέλει βλέπει τις δοσοληψίες της επιθυμητής χρονικής περιόδου. Πατώντας το κουμπί “**Clear**” καθαρίζει το φίλτρο επομένως εμφανίζονται πάλι όλες οι δοσοληψίες για όλες τις παραγγελίες.



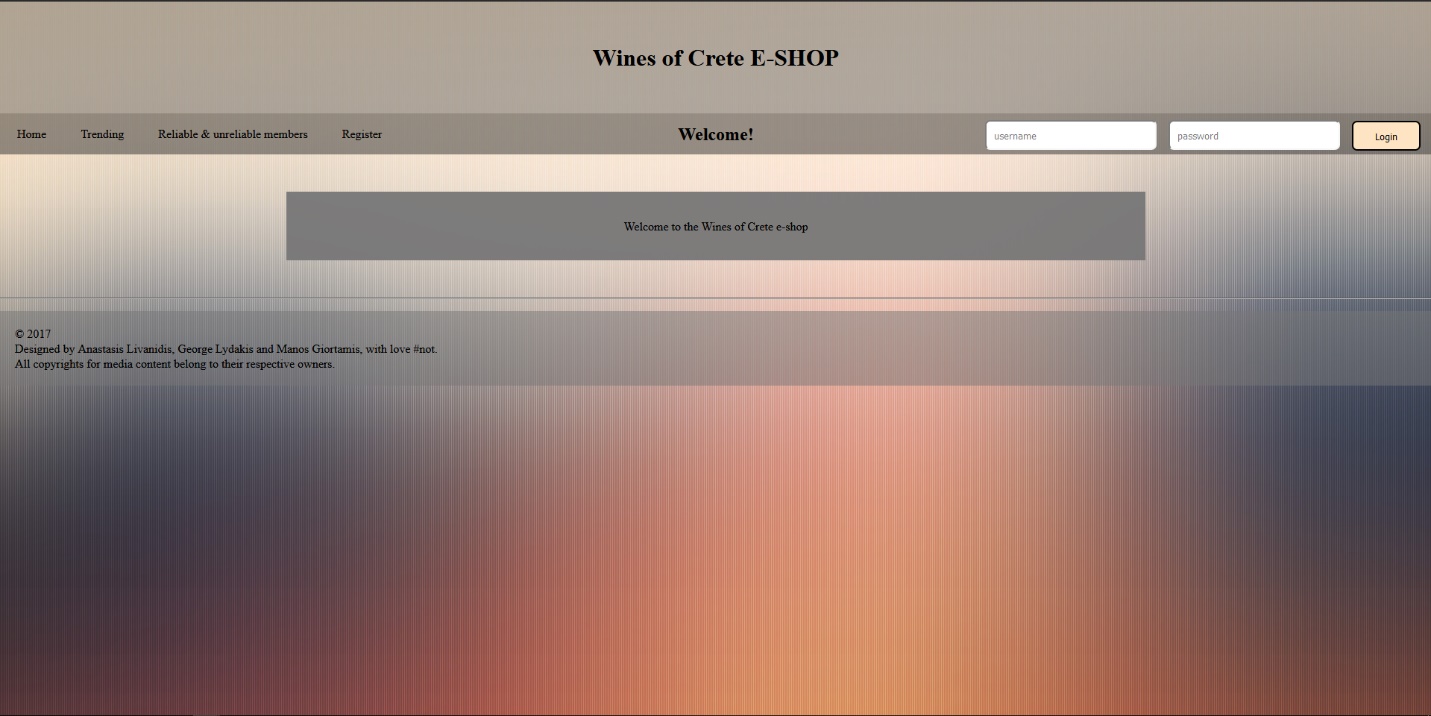
6.3.5: Κλείσιμο Λογαριασμού

Στην σελίδα «Πληροφορίες Λογαριασμού» (“Account Details”), το τελευταίο πεδίο είναι αυτό που λέγεται “Close Account”. Από εκεί μπορεί ο χρήστης να διαγράψει μόνιμα τον λογαριασμό του. Για να το κάνει αυτό, πρέπει να πατήσει το κόκκινο κουμπί “**Close Account**”. Για να διαγραφεί επιτυχώς ο λογαριασμός, απαραίτητη προϋπόθεση είναι να μην χρωστάει στο κατάστημα ο χρήστης και η κατάσταση όλων των παραγγελιών του να είναι “DELIVERED”.

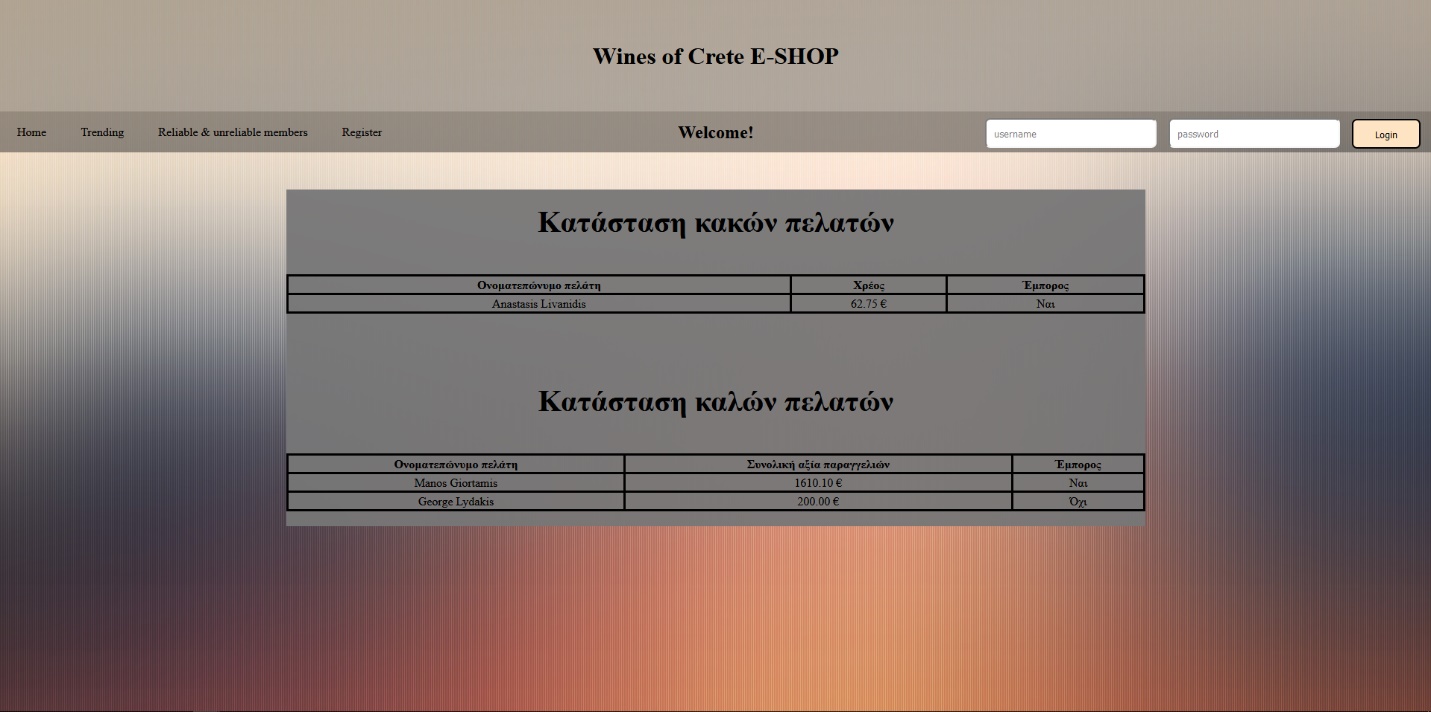


Ενότητα 7: Ενδεικτικά αποτελέσματα

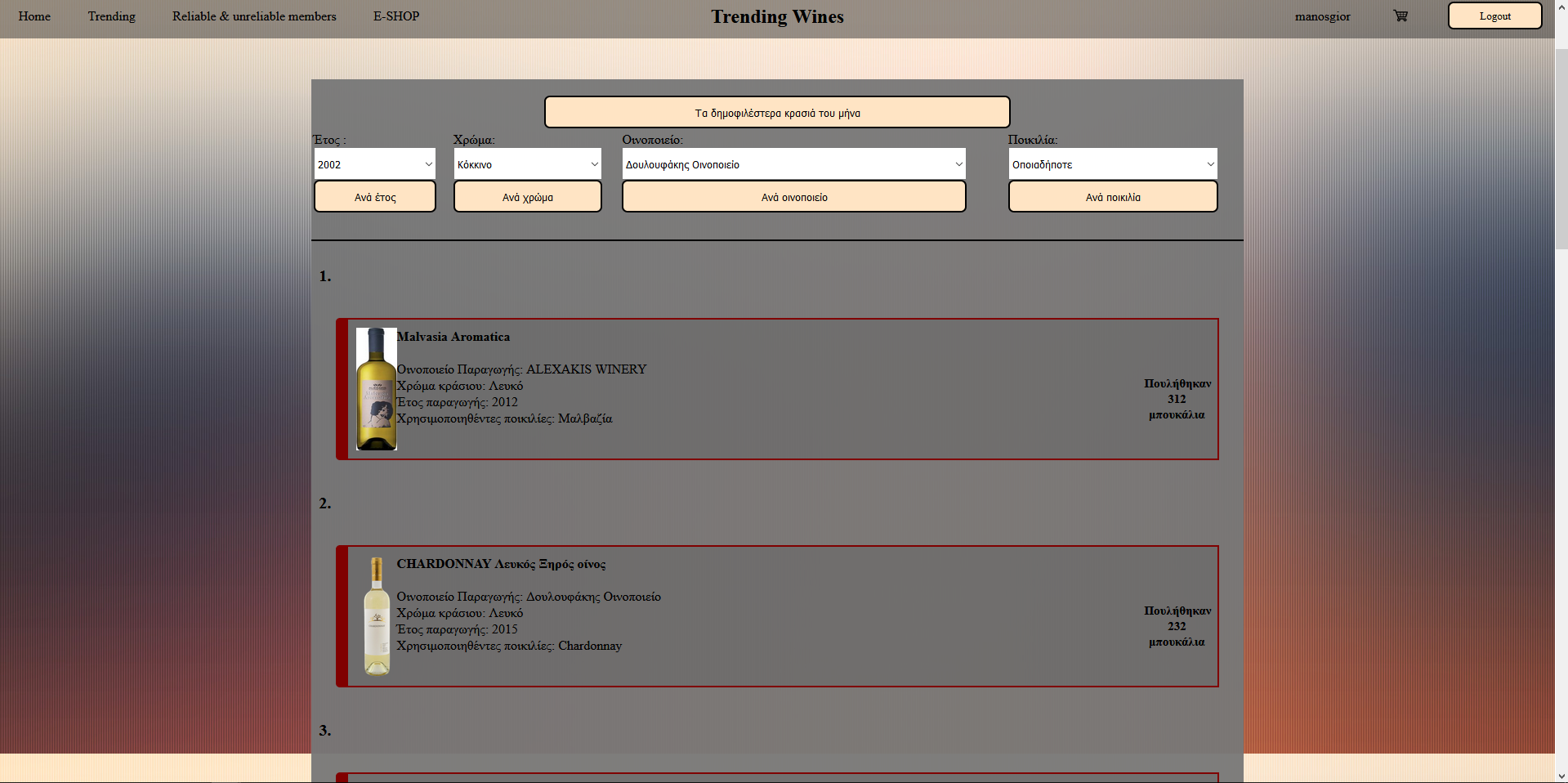
Η Αρχική σελίδα του καταστήματος :



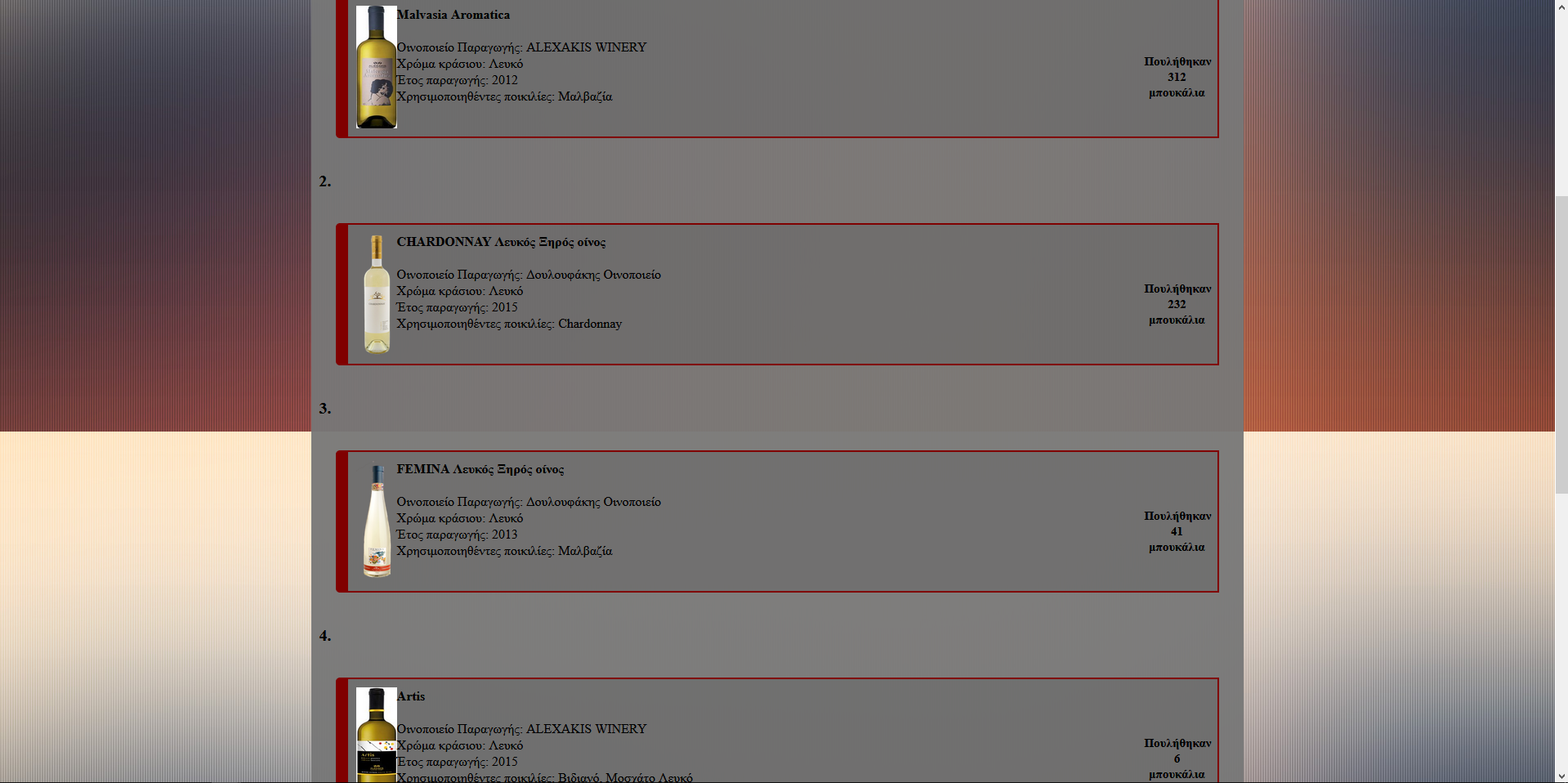
Η σελίδα Καλών/Κακών πελατών του καταστήματος :



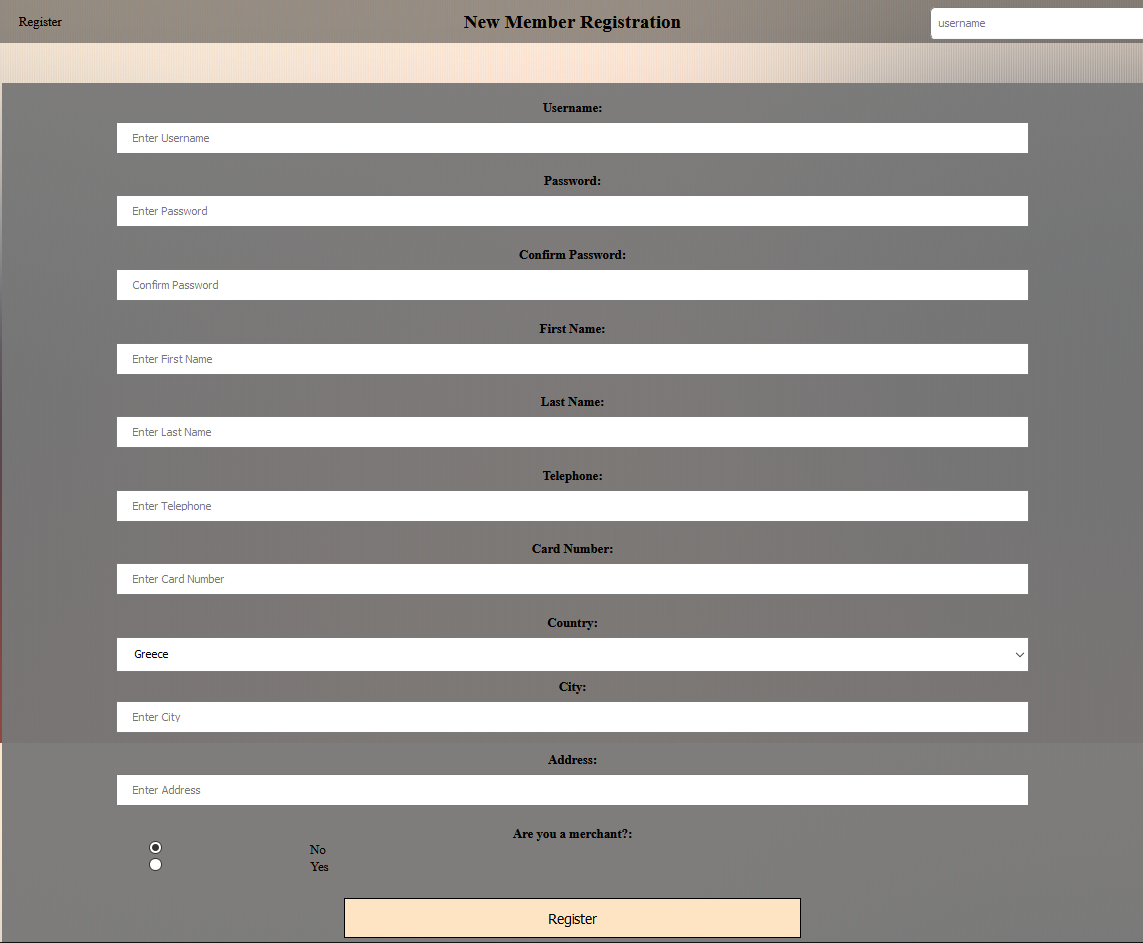
Η σελίδα των Δημοφιλέστερων κρασιών:

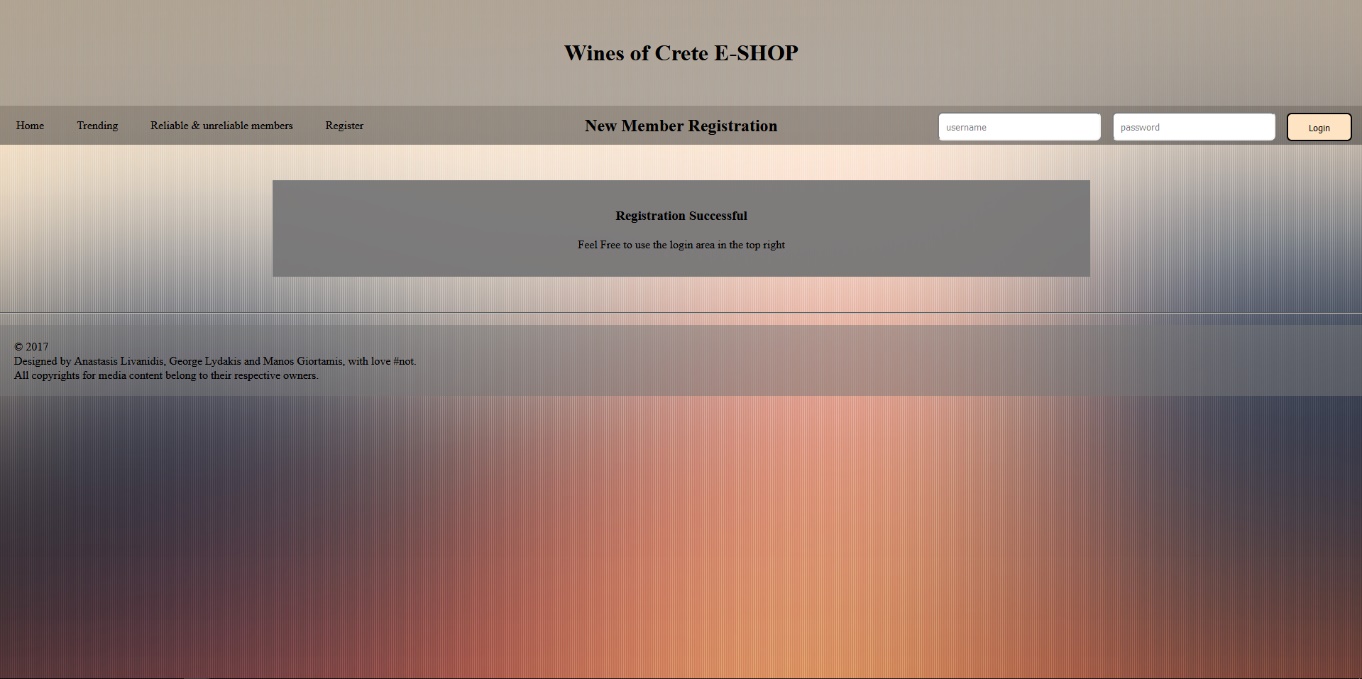


Η σελίδα των Δημοφιλέστερων κρασιών με φίλτρο αναζήτησης:

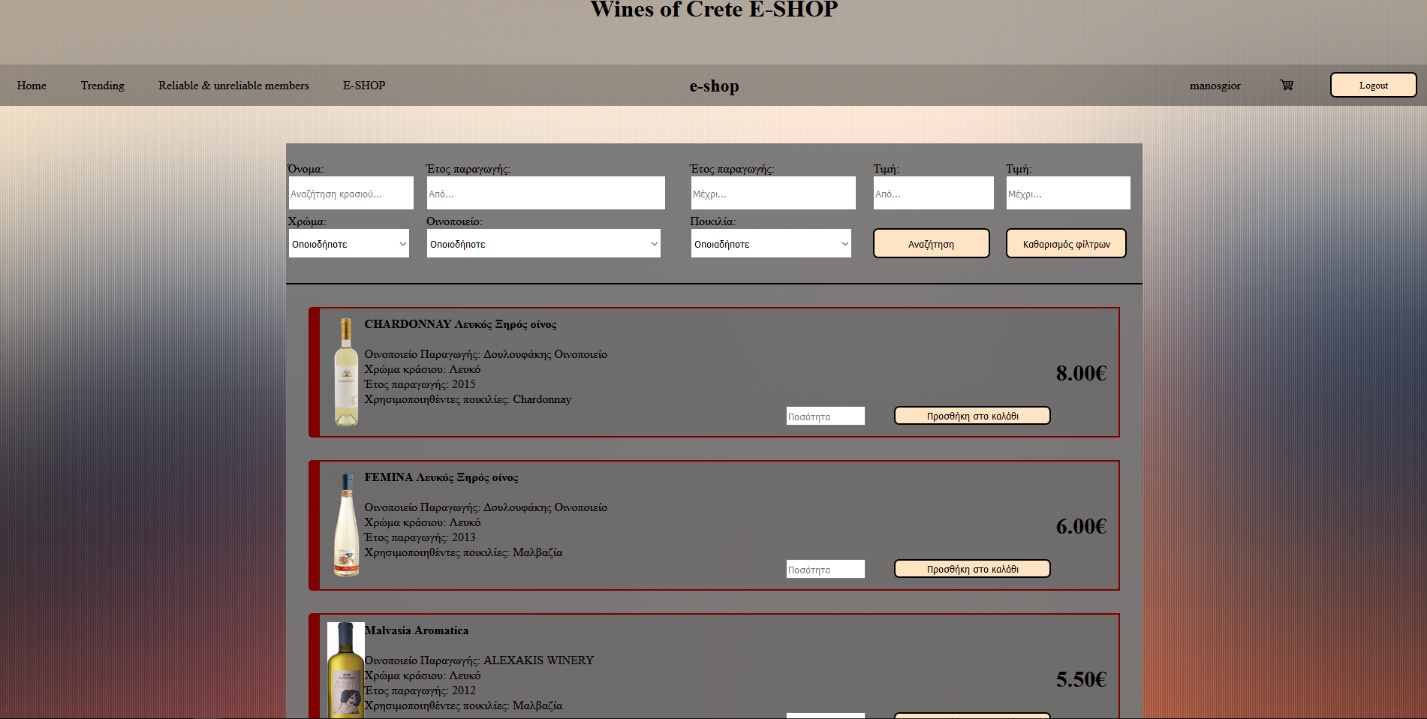


Η εγγραφή χρήστη:

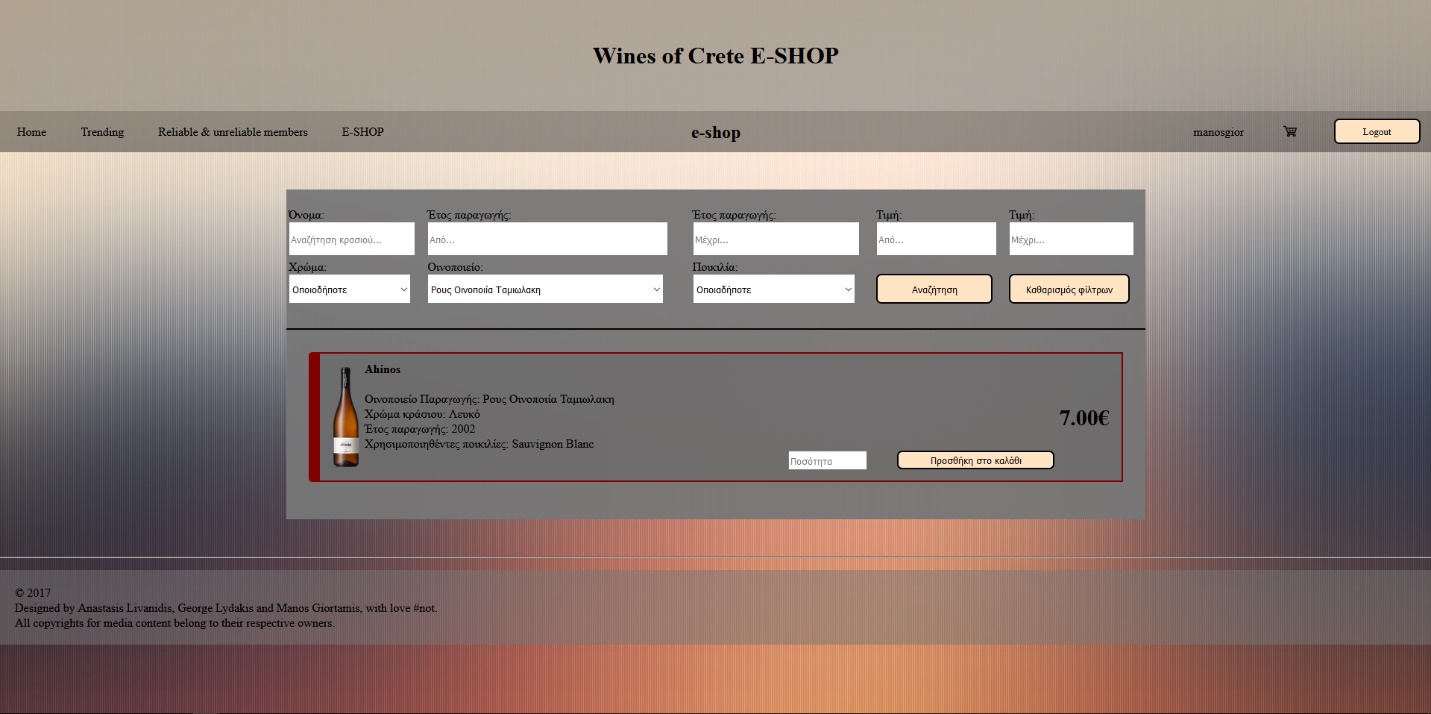


Επιτυχημένη εγγραφή:

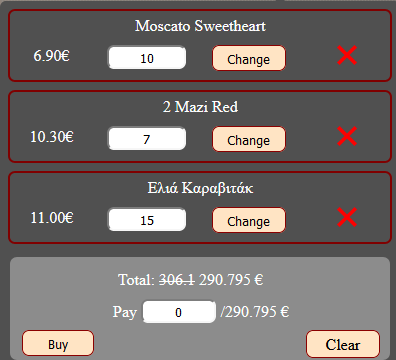
To E-SHOP:

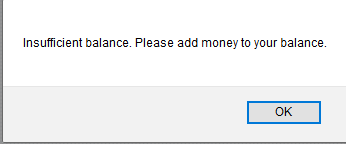


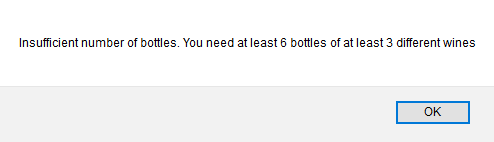
To E-SHOP με φίλτρα αναζήτησης:



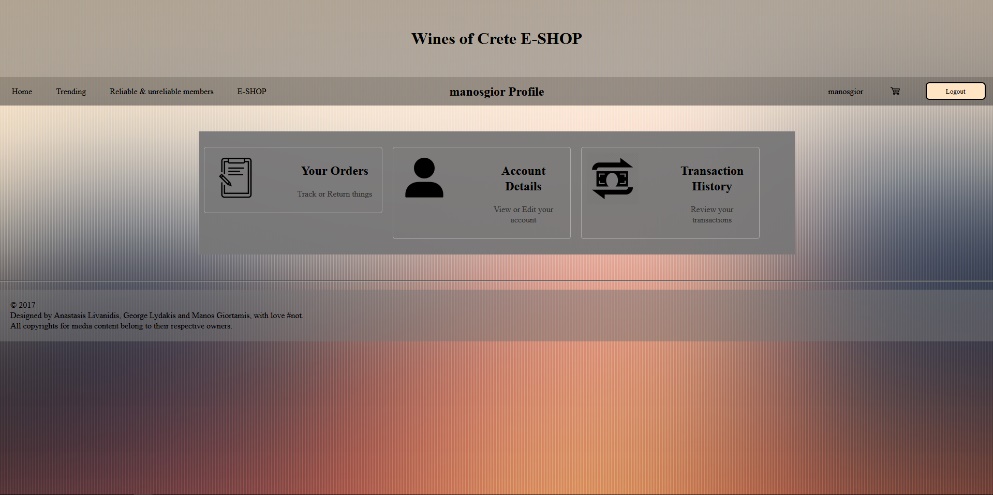
To καλάθι (δίπλα με έκπτωση):

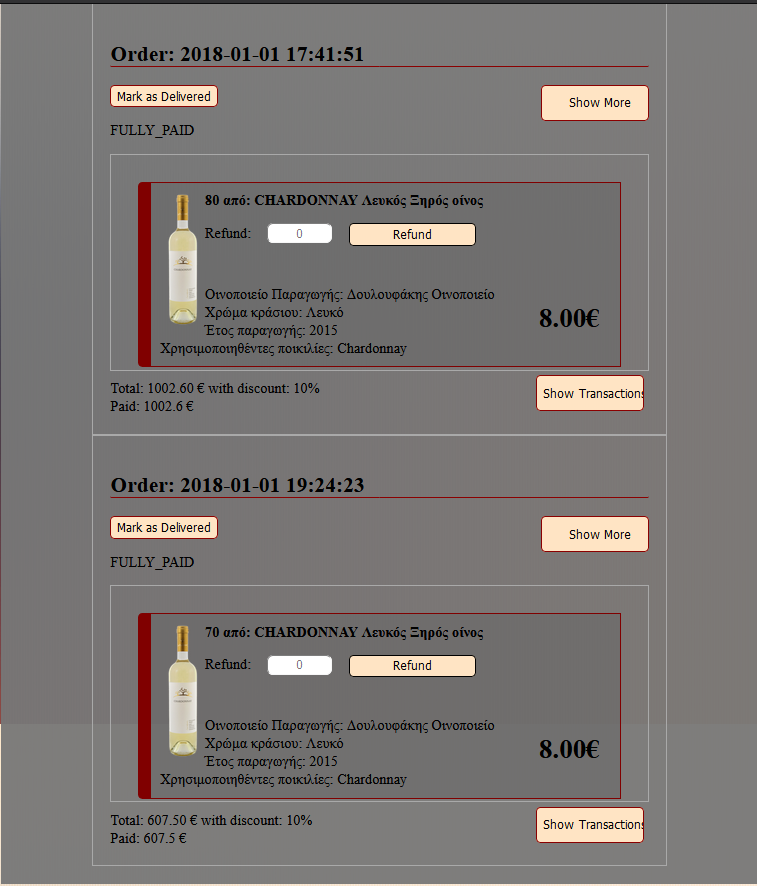


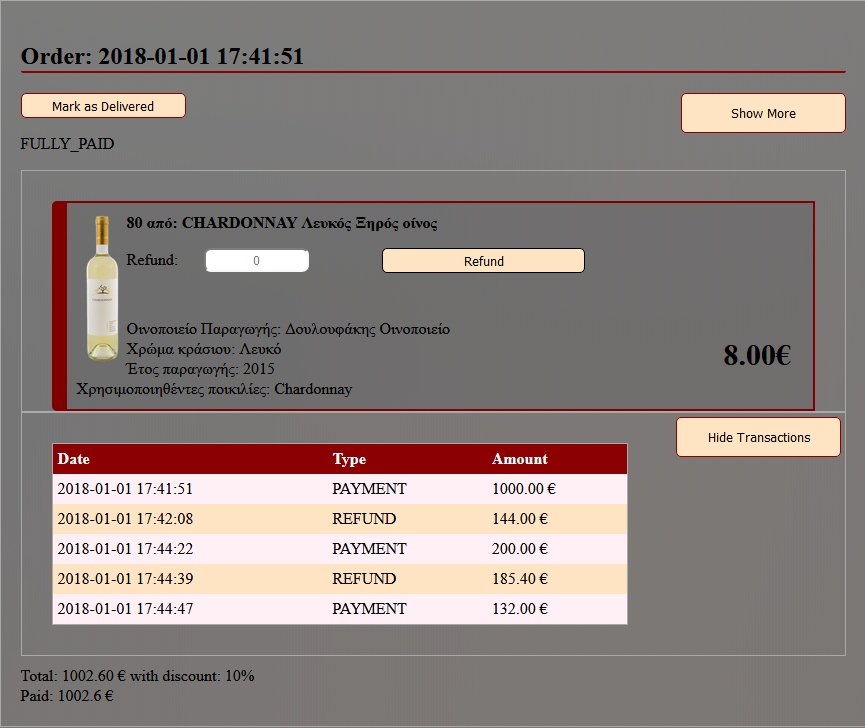
Error message όταν δεν υπάρχει αρκετό balance:

Error message όταν δεν υπάρχουν αρκετά κρασιά στην παραγγελία (έμπορος):

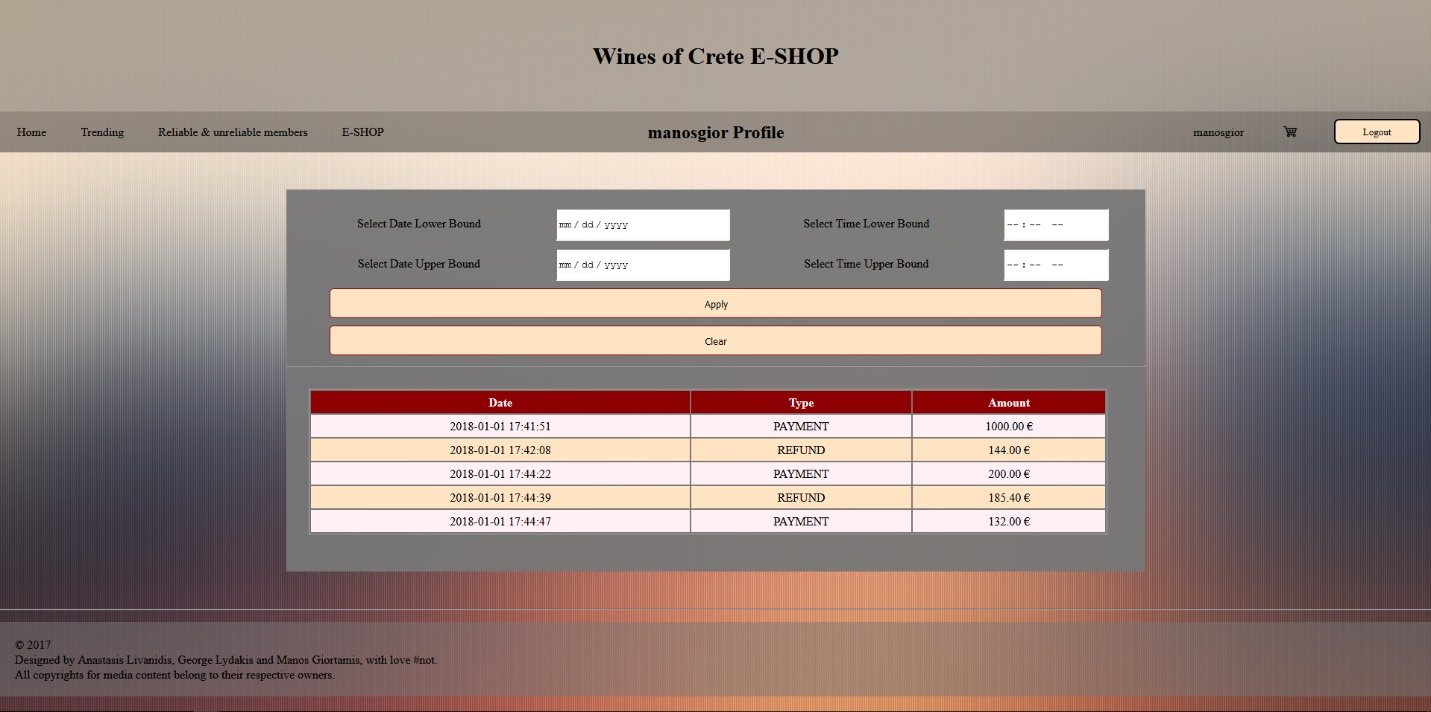
Σελίδα προφίλ του χρήστη:



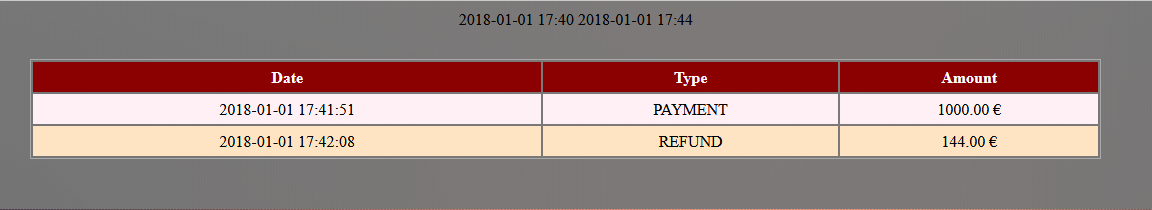
Οι παραγγελίες του χρήστη: 

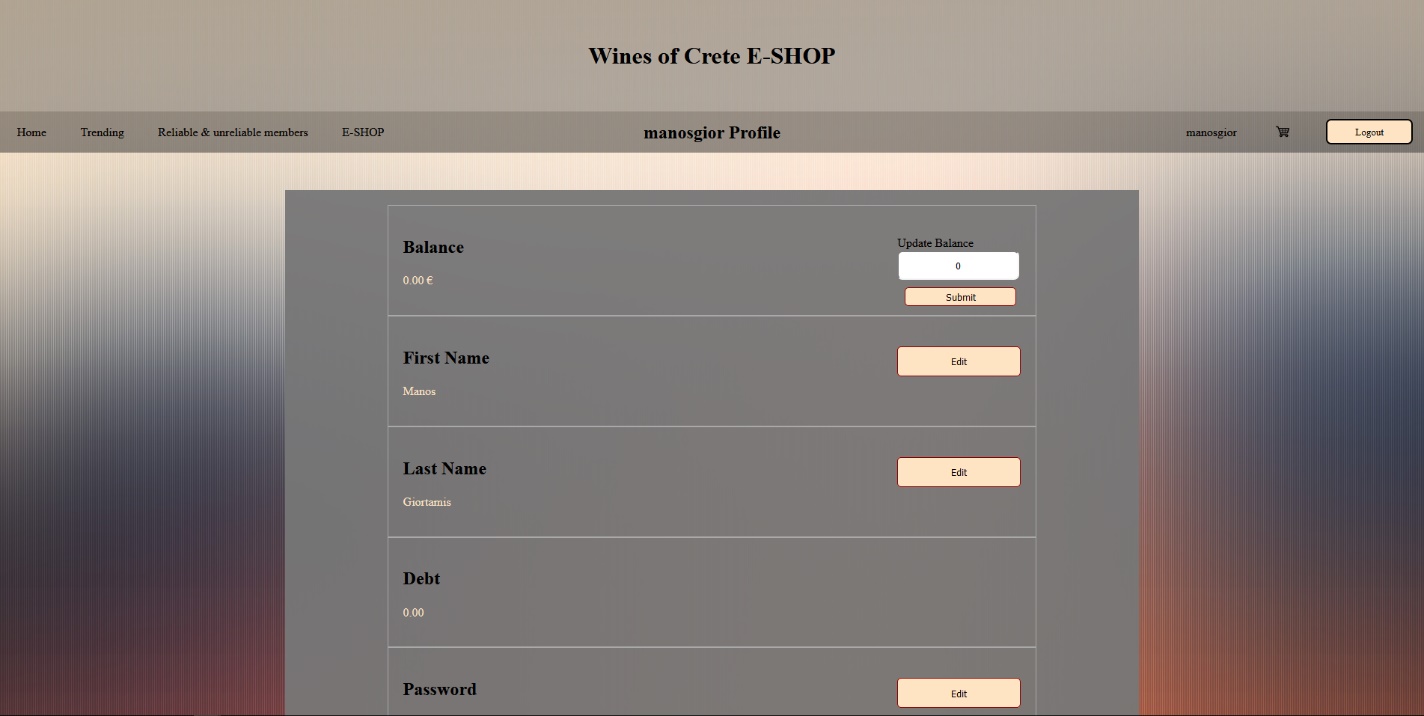
Δοσοληψίες της παραγγελίας:

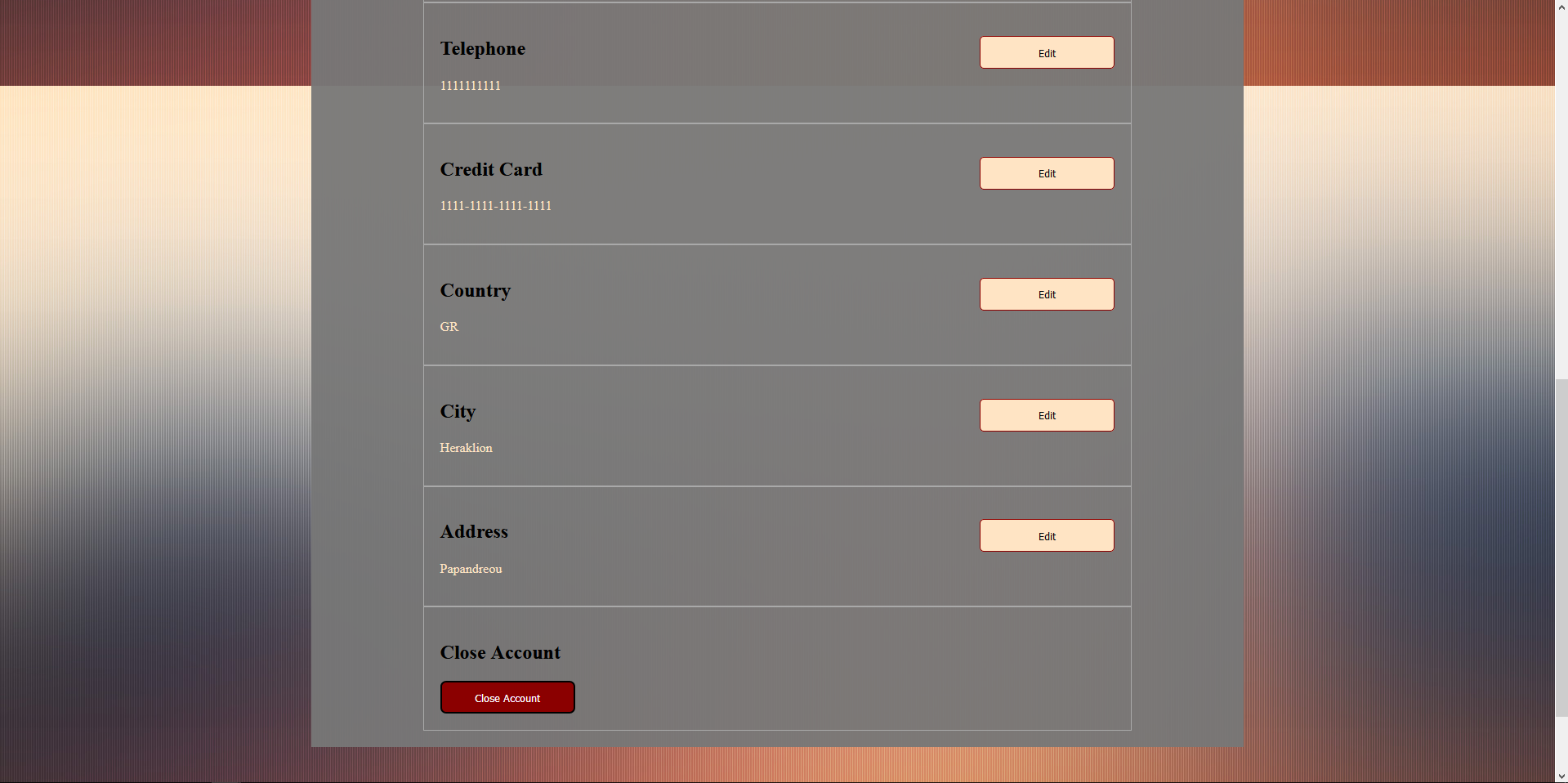
Ιστορικό παραγγελιών του χρήστη:



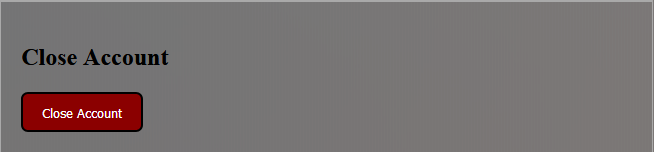
Ιστορικό παραγγελιών με φίλτρα:



Γενικές πληροφορίες:



Κλείσιμο λογαριασμού:



Ενότητα 8: Περιορισμοί Υλοποίησης και Δυνατότητες Βελτίωσης

Ως προς την εκφώνηση, δεν υπάρχει κάτι στο οποίο να περιοριστήκαμε (με βάση και την διόρθωση του ότι και οι έμποροι και οι πελάτες παραγγέλνουν με δυνατότητα πληρωμής εντός 10 ημερών).

Ως προς τους δικούς μας στόχους, τα triggers θα μπορούσαν να είναι περισσότερα και να ελέγχονται πιο πολλά πράγματα και στην βάση εκτός του server, η σελίδα θα μπορούσε να είναι SPA (Single Page Application), να λειτουργεί δηλαδή με AJAX requests που θα αλλάζουν τη σελίδα δυναμικά χωρίς refresh, η σελίδα να έχει μεγαλύτερο έλεγχο ως προς το ποιος browser χρησιμοποιείται ώστε να κάνει τις κατάλληλες αλλαγές για μη-συμβατά elements, η δομή των αρχείων να ήταν πιο οργανωμένη, δηλαδή τα αρχεία που δεν αποτελούν ολόκληρες σελίδες αλλά είναι μέρος σελίδων να είναι σε ξεχωριστούς φακέλους, η σελίδα να υλοποιούσε REST μιας και με την PHP κάτι τέτοιο είναι δυνατό, καθώς και άλλες μικροαλλαγές στο GUI για την διευκόλυνση του χρήστη.