Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Πρόγραμμα καταγραφής και στατιστικής απεικόνισης των εξόδων ενός πλοίου εμπορικής ναυτιλίας** **Μια οικονομική - πληροφοριακή προσέγγιση** **Cost data processing and analyzing program for a merchant shipping vessel** **An economic – data processing approach**
Ονοματεπώνυμο Φοιτητή	**Νικόλαος Βολιανάκης**
Πατρώνυμο	**Ιωάννης**
Αριθμός Μητρώου	**ΜΠΠΛ 16005**
Επιβλέπων	**Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος 2020**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Κωνσταντίνος Πατσάκης Επίκουρος Καθηγητής

Πίνακας Περιεχομένων

1. Περίληψη 2
2. Εισαγωγή 3
2.1 Ναυτιλιακή οικονομική και λογιστική 3
2.2 Προβλήματα στην εφαρμογή της λογιστικής στις ναυτιλιακές επιχειρήσεις 3
2.3 Τα εργαλεία του πληροφοριακού κλάδου 4
3. Ανασκόπηση πεδίου 5
3.1 Quickbooks 5
3.2 Freshbooks 7
4. Παρουσίαση και χρήση εφαρμογής (User's manual) 9
4.1 Login 9
4.2 Register 10
4.3 Main Page 10
4.4 Add a new port 11
4.5 Insert expenses 12
4.6 Expenses table 13
4.7 Statistics page 13
4.8 All-in-all Statistics 14
4.9 Filtered Statistics 14
5. Αρχιτεκτονική συστήματος 15
5.1 Η γλώσσα προγραμματισμού C# 15
5.2 MVC Framework 16
5.3 Entity Framework 17
6. Συμπεράσματα και μελλοντικές επεκτάσεις 18
7. Βιβλιογραφία 19
7.1 Συγγράμματα 19
7.2 Ιστοσελίδες 19
8. Κώδικας 20
8.1 Controllers 20
8.1.1 AccountController.cs 20
8.1.2 HomeController.cs 21
8.1.3 StatisticsController.cs 22
8.2 Models 23
8.2.1 UserModel.cs 23
8.3 Views 24
8.3.1 CardGrid.cshtml 24
8.3.2 _Chart.cshtml 25
8.3.3 _Freq.cshtml 28

1. Περίληψη

Στην παρούσα εργασία θα αναφερθούμε εμπεριστατωμένα στον κλάδο της ναυτιλίας. Μέσω του κλάδου αυτού πραγματοποιείται το 90% του παγκόσμιου εμπορίου[1]. Αμέσως καταλαβαίνουμε την δύναμη αλλά και την σημαντικότητα του κλάδου αυτού. Σημαντικός πυλώνας της ναυτιλίας είναι οι πλοιοκτήτριες εταιρίες, που με τον στόλο πλοίων τους ικανοποιούν τις ανάγκες του παγκόσμιου πληθυσμού. Κάνεις δεν θα μπορούσε να μειώσει την σημαντικότητα μια πλοιοκτήτριας εταιρίας, καθώς δεν μπορούμε να ορίσουμε ένα εύρος των αγαθών που μεταφέρονται με πλοία. Στην εποχή μας *όλα* τα αγαθά (σιτηρά, σιδηρομετάλλευμα, άνθρακας, πετρέλαιο κλπ.) υπάγονται στην θαλάσσια μεταφορά είτε σε ολικό αλλά είτε και σε μέρος της μεταφοράς τους. Ας επικεντρωθούμε λοιπόν σε μια ναυτιλιακή εταιρία.

Όπως κάθε υγιής εταιρία, έτσι και οι ναυτιλιακές εταιρίες ξοδεύουν αρκετούς πόρους στη ανάλυση κόστους τους. Αυτό υπάγεται στην απόλυτη λογική αν αναλογιστούμε τα τεράστια μεγέθη κόστους που διέπουν ένα πλοίο. Στο σημείο αυτό θα πρέπει να ορίσουμε τον όρο κόστος για την ναυτιλία. Ο όρος αυτός δεν αναφέρεται μόνο στο χρηματικό παράγοντα, αλλά και στον ανθρώπινο και περιβαλλοντικό παράγοντα, χωρίς ουδείς από αυτούς να επικαλύπτει τον άλλον. Στην παρούσα στιγμή θα αναφερθούμε στο οικονομικό κόστος μιας πλοιοκτήτριας εταιρίας όσον αφορά τα πλοία της.

Όπως είναι απόλυτα λογικό, ένα πλοίο έχει μεγάλο αριθμό διαφοροποιημένων εξόδων κατά την διάρκεια ενός ταξιδιού του. Το πρόβλημα έρχεται στο γεγονός ότι κάθε λιμάνι έχει διαφορετικά έξοδα και κόστη, τα οποία πρέπει να καταγράφουν λεπτομερώς από τον καπετάνιο του πλοίου, με σκοπό την αναφορά αυτών στην πλοιοκτήτρια εταιρία. Η παραδοσιακή χειρόγραφη καταγραφή εξόδων όταν μιλάμε για μεγάλο όγκο και διαφορετικότητα είδους οδήγησε σε ένα πρόβλημα παρακολούθησης κόστους. Χωρίς να είναι απορίας άξιο αυτό το πρόβλημα έρχεται να λυθεί με την βοήθεια του κλάδου της πληροφορικής.

Το πρόγραμμα μας είναι μια δικτυακή πλατφόρμα καταγραφής των διαφόρων αυτών εξόδων του πλοίου, με σκοπό την διευκόλυνση της καθημερινής λειτουργίας του ίδιου του πλοίου αλλά και της στατιστικής απεικόνισης αυτών για τους σκοπούς της εταιρίας.

1. Summary

In the present work we will deal in detail with the shipping industry. This sector accounts for 90% of world trade. Immediately we understand the strength and the importance of this industry. An important pillar of shipping is the ship-owning companies, which with their fleet meet the needs of the world's population. Doing could not reduce the importance of a shipowner, as we cannot define a range of goods carried by ships. In our time, all goods (grain, iron ore, coal, oil, etc.) are part of the maritime transport either in their entirety or even part of their transport. So, let's concentrate on a shipping company.

Like any healthy company, shipping companies are spending enough resources on their cost analysis. This comes under the absolute logic of considering the huge cost of a ship. At this point we should define the shipping cost term. This term refers not only to the monetary factor but also to the human and environmental factors, with none of them overlaps the other. At this time, we will refer to the financial cost of a ship-owning company for its ships

It is perfectly normal to say that a ship has a very large number of different expenses during its voyage. The problem results from the fact that each port has different expenses in terms of type and amount, which have to be properly documented by the master of the vessel in order to file a well detailed report to the ship owning company. The traditional hand-writing technique of documenting such a big amount of different expenses, led to a problem of cost checking and controlling. It goes without saying that this problem will be sold with the help of the computer science field and its tools.

Our program is an online cost-documenting platform, offering details of each of the vessel's expenses as well as a statistical analysis of same for the uses of either the master himself or the ship owning company.

[1] http://www.ics-shipping.org/shipping-facts/shipping-and-world-trade

2. Εισαγωγή

2.1 Ναυτιλιακή οικονομική και λογιστική

Η Ναυτιλία είναι μια βιομηχανία περίπλοκη στην οποία οι συνθήκες που καθορίζουν τις διαδικασίες σε έναν τομέα της δεν εφαρμόζονται και σε άλλο. Κάτω από ορισμένες προϋποθέσεις μπορεί να θεωρηθεί σαν ένα σύνολο από αλληλοσχετιζόμενες βιομηχανίες. Τα θεμελιώδη στοιχεία της, τα πλοία, διαφέρουν σε μέγεθος και τύπο, τα οποία προσφέρουν υπηρεσίες μεταφοράς για μεγάλη ποικιλία φορτίων που μεταφέρονται σε μακρινές ή κοντινές θαλάσσιες αποστάσεις[2].

Στο χώρο της ναυτιλιακής βιομηχανίας αν και μπορούμε να διαχωρίσουμε τομείς και αγορές δεν μπορούμε να παραγνωρίσουμε την αλληλεπίδραση και αλληλεξάρτησή τους. Επιπροσθέτως πρέπει σημειωθεί ότι η ναυτιλιακή βιομηχανία, στο μεγαλύτερο μέρος της, ασχολείται με την εκτέλεση του διεθνούς εμπορίου και επομένως λειτουργεί σε ένα περίπλοκο παγκόσμιο πλαίσιο με οικονομικές, πολιτικές και κοινωνικές συμφωνίες ανάμεσα σε ναυτιλιακές εταιρίες, φορτωτές, κυβερνητικές οργανώσεις και άλλα εμπλεκόμενα μέρη.

Είναι πολύ σημαντικό να δοθεί έμφαση στην αναγνώριση των εμπορικών και οικονομικών διαφορών μεταξύ των διαφορετικών τμημάτων της ναυτιλιακής αγοράς, για παράδειγμα οι υπηρεσίες της ναυτιλίας γραμμών είναι τελείως διαφορετικές από αυτές της ναυτιλίας μεταφοράς φορτίων και υπάρχει διαφορετική οικονομική δομή καθόσον μεταφέρονται διαφορετικοί τύποι φορτίων. [3]

Κυριότερος συνδετικός παράγοντας για όλα τα παραπάνω είναι το πλοίο το οποίο αποτελεί μια επιχείρηση εν κινήσει που δημιουργεί έσοδα και έξοδα ανά τον κόσμο, την οικονομική διαχείριση αυτού επωμίζεται ο πλοίαρχος του πλοίου, ο οποίος εκτελεί χρέη λογιστή κατά τη διάρκεια του ταξιδιού, διαχειρίζεται τα χρηματικά διαθέσιμα και το καθημερινό εξοδολόγιο και αποδίδει λογαριασμό στην ναυτιλιακή επιχείρηση. Μέσω της κατάρτισης της ειδικής λογιστικής κατάστασης με τον τίτλο «Γενικός Λογαριασμός Πλοιάρχου» (Master's General Account), η οποία παρέχεται από τον πλοίαρχο στην ναυτιλιακή επιχείρηση, συνοψίζονται όλα τα πεπραγμένα και λογιστικά δρώμενα της περιόδου για την οποία καταρτίστηκε και αποδίδεται οικονομικός απολογισμός. Προφανώς, στις χερσαίες επιχειρήσεις όλο το λογιστικό κύκλωμα συνδέεται μέσω συστημάτων intranet για όλα τα υποκαταστήματα και καταλήγει στο λογιστήριο και κατ' επέκταση την οικονομική διεύθυνση, χωρίς να παρεμβάλλονται ενδιάμεσες οικονομικές υπο-διευθύνσεις με την έννοια του λογαριασμού του πλοιάρχου.

2.2 Προβλήματα στην εφαρμογή της λογιστικής στις ναυτιλιακές επιχειρήσεις

Η οργάνωση και ομαλή λειτουργία ενός λογιστηρίου στις ναυτιλιακές επιχειρήσεις αντιμετωπίζει μια πληθώρα προβλημάτων λογιστικής φύσης κυρίως λόγω του γεγονότος ότι το πλοίο προβαίνει σε οικονομικές συναλλαγές κατά τη διάρκεια του ταξιδιού του ανά την υδρόγειο και μάλιστα σε διαφορετικά νομίσματα. Αυτό συνεπάγεται:

- Την εξαιρετική χρονική καθυστέρηση στην παραλαβή των παραστατικών από τις συναλλαγές που πραγματοποίησε ο πλοίαρχος ή ο πράκτορας στα διάφορα λιμάνια προσέγγισης.
- Την αναζήτηση των αναλογούντων ισοτιμιών αυτών των δοσοληψιών λαμβανομένου υπόψη ότι χρησιμοποιείται μια πληθώρα διαφορετικών νομισμάτων με αποτέλεσμα να προκύπτουν πολλές συναλλαγματικές διαφορές.

2 Βλάχος Γ.- Αλεξόπουλος Α. (1996), Διεθνείς Οργανισμοί και Ναυτιλιακή Πολιτική, Εκδόσεις Α. Σταμούλη, Πειραιάς

3 Βλάχος Γ.Π. – Γεωργαντόπουλος (2003), Ναυτιλιακή Οικονομική», Εκδόσεις Τζέι&Τζέι Ελλάς, Πειραιάς

- Τη μετατροπή εξόδων σε λογαριασμό απαιτήσεων. Ένα μέρος των εξόδων του πλοίου, κυρίως όσον αφορά την επισκευή και συντήρηση αυτού, καλύπτονται από ασφαλιστήρια συμβόλαια, με αποτέλεσμα να απαιτείται ειδικός λογιστικός χειρισμός κατά τη μετατροπή των εξόδων αυτών σε απαιτήσεις από ασφαλιστικές εταιρίες.[4]

2.3 Τα εργαλεία του πληροφοριακού κλάδου

Το παραπάνω πρόβλημα έρχεται να λύσει η Πληροφορική με την πληθώρα εργαλείων της και την δυνατότητα εφαρμογής της σε κάθε κλάδο και βιομηχανία. Σε αυτήν την εργασία δημιουργήσαμε μια πλατφόρμα καταγραφής κόστους. Με αυτήν την πλατφόρμα ο καπετάνιος μπορεί να περνάει τα καθημερινά έξοδα του πλοίου με σκοπό την λεπτομερή καταγραφή του για χρήση του ίδιου και της πλοιοκτήτριας εταιρίας καθώς και για την στατιστική απεικόνιση αυτών για μελλοντική χρήση (decision making etc).

Η πλατφόρμα μας, δίνει την δυνατότητα κατηγοριοποίησης του κάθε εξόδου καθώς και την εισαγωγή ενός πολύ σημαντικού παράγοντα της ναυτιλίας, ο οποίος είναι το λιμάνι. με την εισαγωγή των εξόδων ανά κατηγορία και ανά λιμάνι, μπορούμε πολύ εύκολα να συμπεράνουμε ποιο λιμάνι είναι πιο φτηνό ανά έξοδο. Αυτό είναι μια πολύτιμη πληροφορία για μια ναυτιλιακή εταιρία. Θα υποθέσουμε για παράδειγμα ότι ένα πλοίο θα περάσει από 5 λιμάνια κατά την διάρκεια του ταξιδιού του. μια πολύ συνηθισμένη ανάγκη ενός πλοίου είναι τα τροφο-εφοδια. Με την χρήση της πλατφόρμας μας μπορούμε να διακρίνουμε σε ποιο λιμάνι τα τροφο-εφοδια είναι πιο επωφελή να παραλάβουμε, από άποψη κόστους. Το παραπάνω παράδειγμα σαφώς ισχύει για όλες τις ανάγκες του πλοίου που καλύπτονται στο λιμάνι όπως ανταλλακτικά, λιπαντικά, τσιγάρα κλπ.

Τα έξοδα μας μπορούν να χωριστούν σε δυο κατηγορίες. Η πρώτη είναι το είδος του εξόδου και η δεύτερη είναι ο κωδικός του εξόδου ο οποίος παρέχεται από την εταιρία. Αυτό συμβάλλει τόσο στην ευκολία χρήσης την πλατφόρμας μας από τον καπετάνιο αλλά και στην εύκολη χρήση δεδομένων από την εταιρία χωρίς περεταίρω επεξεργασία.

Επίσης η πλατφόρμα μας επιτρέπει την ταυτόχρονη σύνδεση πολλών διαφορετικών χρηστών, όπου ο κάθε ένας έχει πρόσβαση μόνο στα δικά του έξοδα για την αποφυγή σύγχυσης. Σε αντίθεση με το παραπάνω, η πλοιοκτήτρια εταιρία σαν χρήστης έχει πρόσβαση σε όλα τα έξοδα όλων των χρηστών κάτι το οποίο εξυπηρετεί τους παρακάτω σκοπούς¨

- Μια γενικότερη εικόνα των εξόδων όλου του στόλου.
- Μια μορφή ελέγχου του καπετάνιου και της διαχείρισης του ταμείου του πλοίου από αυτόν.
- Μια πρόβλεψη μελλοντικού κόστους με ακρίβεια μεγάλου βαθμού.

Εν συνέχεια, η πλατφόρμα μας παρουσιάζει ένα στατιστικό μοντέλο των εξόδων ανά μήνα και ανά λιμάνι, κάτι το οποίο εξυπηρετεί όχι μόνο τις παραπάνω ανάγκες αλλά και τις ανάγκες στατιστικών υπηρεσιών οι οποίες αντλούν τα δεδομένα τους από πλοιοκτήτριες εταιρίες.

4 ΣΩΤΗΡΙΑ Ι. ΕΛΕΥΘΕΡΙΟΥ Λογιστής Φοροτεχνικός Α' Τάξης & Ειδικός Εισηγητής Λογιστικής ΥΜΗΤΤΟΥ 75, 11633 ΠΑΓΚΡΑΤΙ – Τ/Φ: 211-0149320 www.sotele.gr

3. Ανασκόπηση πεδίου

Υπάρχει μια πληθώρα εφαρμογών και προγραμμάτων που ασχολούνται με την καταγραφή και ανάλυση εξόδων κάτι το οποίο είναι απόλυτα λογικό αν αναλογιστούμε ότι τα έξοδα είναι μείζονος σημασίας σε όλα τα μεγέθη οικονομικού μοντέλου, από μια πολυεθνική επιχείρηση μέχρι μια πιο μικρή επιχείρηση λιανικής, ακόμα και σε επίπεδο οικογενειακού προϋπολογισμού.

Ενδεικτικά θα αναφέρουμε μια λίστα με τα δημοφιλέστερα προγράμματα καταγραφής εξόδου και λογιστικού περιεχομένου γενικότερα και θα αναλύσουμε εμπεριστατώμενα μερικά από αυτά[5].

- Quickbooks
- Freshbooks
- Wave
- Pivot accounting Pro
- VT Cash book

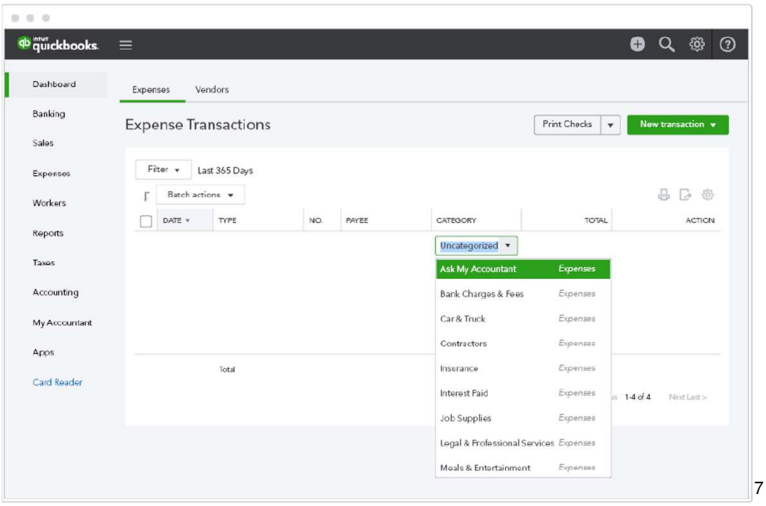
3.1 Quickbooks

Το QuickBooks είναι ένα λογιστικό πακέτο λογισμικού που αναπτύχθηκε και διατέθηκε στην αγορά από την Intuit. Τα προϊόντα QuickBooks απευθύνονται κυρίως σε μικρές και μεσαίες επιχειρήσεις και προσφέρουν λογιστικές εφαρμογές καθώς και εκδόσεις που βασίζονται σε υπηρεσίες cloud που δέχονται πληρωμές από επιχειρήσεις, διαχειρίζονται και πληρώνουν λογαριασμούς και λειτουργίες.

Η Intuit έχει ενσωματώσει πολλές δυνατότητες βασισμένες στο web μέσα στο QuickBooks, όπως δυνατότητες απομακρυσμένης πρόσβασης, απομακρυσμένη βοήθεια και εξωτερική ανάθεση, λειτουργίες ηλεκτρονικών πληρωμών, διαδικτυακές τραπεζικές συναλλαγές και συμφωνίες, δυνατότητες χαρτογράφησης μέσω ενοποίησης με τους Χάρτες Google, δυνατότητες ηλεκτρονικού ταχυδρομείου μέσω Microsoft Outlook και Outlook Express. Για την έκδοση 2008, η εταιρεία έχει προσθέσει επίσης εισαγωγή από υπολογιστικά φύλλα Excel, πρόσθετες επιλογές παρακολούθησης χρόνου υπαλλήλου, προέγκριση ηλεκτρονικών κεφαλαίων και νέες λειτουργίες βοήθειας. Τον Ιούνιο του 2007, η Intuit ανακοίνωσε ότι οι λύσεις QuickBooks Enterprise θα λειτουργούσαν σε διακομιστές Linux, ενώ προηγουμένως απαιτούσε την εκτέλεση ενός διακομιστή Windows.[6]

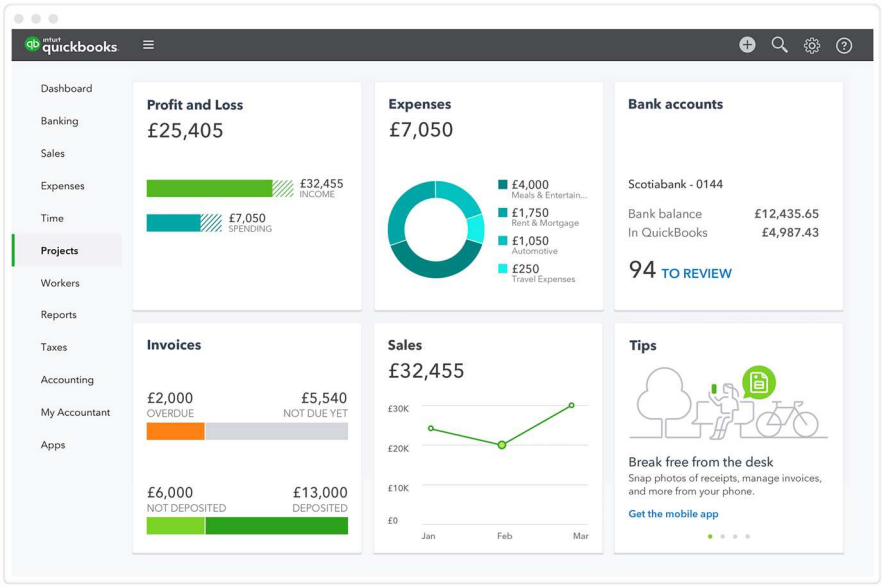
[5] https://www.predictiveanalyticstoday.com/top-free-and-open-source-accounting-software/

[6] https://quickbooks.intuit.com/uk/



[7]

Στην παραπάνω εικόνα παρατηρούμε το παράθυρο εισαγωγής εξόδων στο σύστημα Quickbooks



[8]

Παραπάνω παρατηρούμε την δυνατότητα στατιστικής ανάλυσης του προγράμματος.

Η Intuit προσφέρει επίσης μια υπηρεσία cloud που ονομάζεται QuickBooks Online. Ο χρήστης πληρώνει μηνιαία χρέωση συνδρομής και όχι προκαταβολή και αποκτά πρόσβαση στο λογισμικό αποκλειστικά μέσω ασφαλούς σύνδεσης μέσω προγράμματος περιήγησης στο Web. Η Intuit παρέχει ενημερώσεις κώδικα και αναβαθμίζει τακτικά το λογισμικό αυτόματα, αλλά περιλαμβάνει επίσης αναδυόμενες διαφημίσεις στην εφαρμογή για πρόσθετες υπηρεσίες επί πληρωμή κάτι το οποίο ήταν αντικείμενο κατάκρισης.

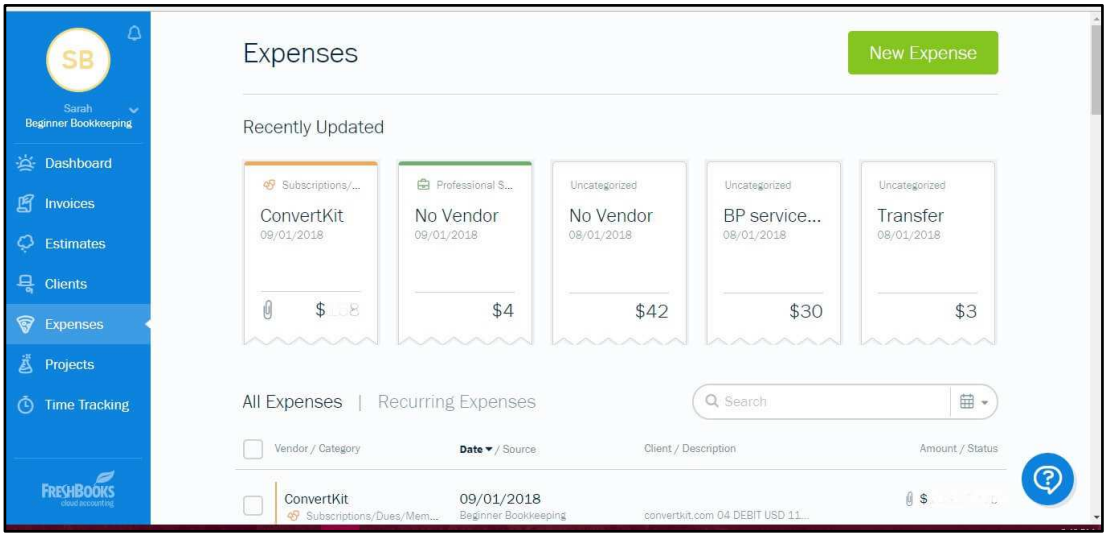
[7] https://quickbooks.intuit.com/global/cloud-accounting-software/

[8] https://quickbooks.intuit.com/uk/accounting-software/

3.2 Freshbooks

Το FreshBooks είναι λογιστικό λογισμικό που διαχειρίζεται η 2ndSite Inc. κυρίως για μικρές και μεσαίες επιχειρήσεις. Πρόκειται για ένα λογισμικό που βασίζεται στον ιστό ως μοντέλο υπηρεσίας (SaaS), στο οποίο μπορούμε να αποκτήσουμε πρόσβαση μέσω επιτραπέζιας ή κινητής συσκευής. Αρχικά το FreshBooks λειτούργησε σαν ένα ηλεκτρονικό πρόγραμμα τιμολόγησης που απευθύνεται σε επαγγελματίες πληροφορικής. Η αρχική έκδοση του FreshBooks αναφέρεται τώρα ως "FreshBooks Classic". Η εφαρμογή front-end του FreshBooks Classic δημιουργήθηκε σε PHP και οι υπηρεσίες backend δημιουργήθηκαν σε Python.

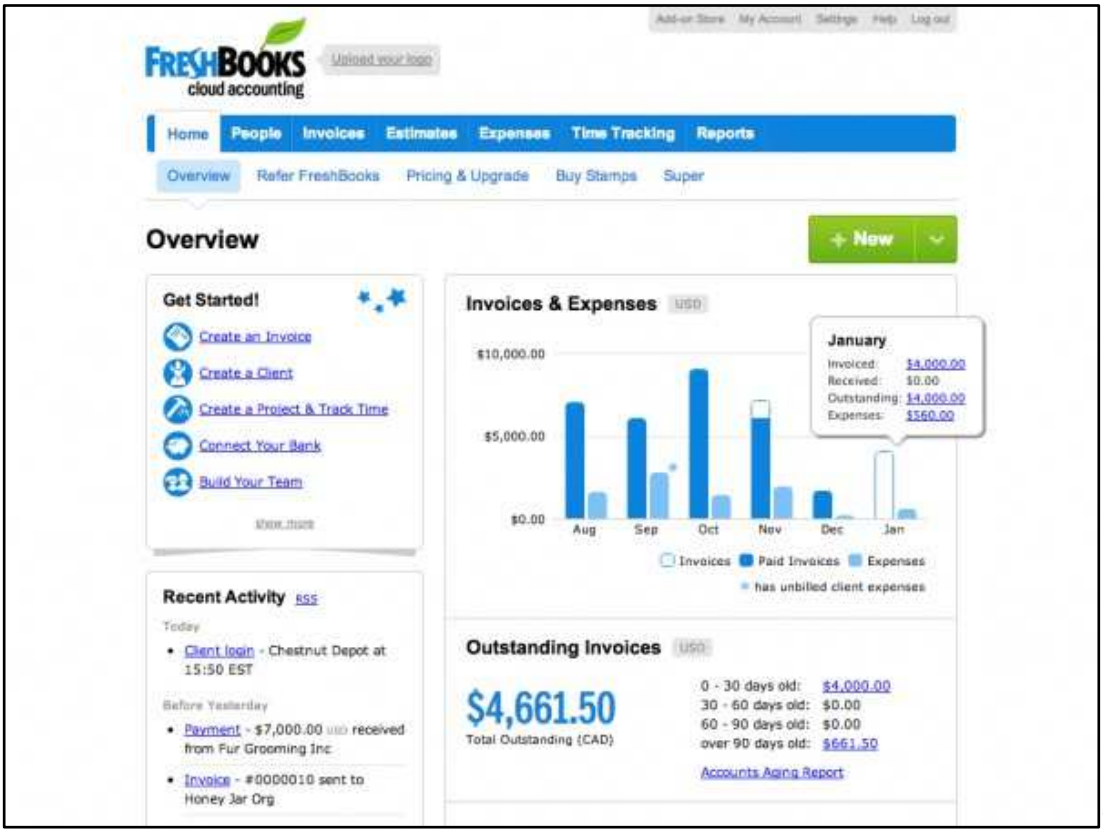
Το FreshBooks προσφέρει ένα προϊόν βάσει συνδρομών που περιλαμβάνει τιμολόγηση, πληρωτέους λογαριασμούς, παρακολούθηση δαπανών, παρακολούθηση χρόνου, συγκράτηση, απόσβεση παγίων στοιχείων, παραγγελίες αγοράς, ενοποιήσεις μισθοδοσίας, λογιστική διπλής καταχώρησης και αναφορές επιχειρήσεων και διαχείρισης βιομηχανικού προτύπου. Όλα τα οικονομικά δεδομένα αποθηκεύονται στο cloud σε ένα ενιαίο καθολικό, επιτρέποντας στους χρήστες να έχουν πρόσβαση στο ίδιο σύνολο βιβλίων ανεξάρτητα από την τοποθεσία σε επιτραπέζιο και κινητό. Προσφέρει ένα δωρεάν API που επιτρέπει στους πελάτες και σε τρίτους προμηθευτές λογισμικού να ενσωματώνουν εξωτερικές εφαρμογές με το FreshBooks. Υποστηρίζει επίσης πολλαπλούς φορολογικούς συντελεστές και νομίσματα. Ενσωματώνει επίσης μια δυνατότητα μισθοδοσίας και μια λειτουργία έργων. Το λογισμικό διατίθεται με επαναλαμβανόμενη μηνιαία αμοιβή ανά χρήση. [9]



[10]

[9] https://en.wikipedia.org/wiki/FreshBooks

[10] https://www.beginner-bookkeeping.com/business-invoicing-software.html



[11]

Αξίζει να αναφέρουμε ότι κανένα από τα παραπάνω προγράμματα δεν ειδικεύεται στον τομέα της ναυτιλίας, αλλά στον τομέα της λογιστικής γενικότερα. Ειδικά και αναλυτικότερα λογιστικά προγράμματα στον τομέα της ναυτιλίας περιλαμβάνουν πλατφόρμες όπως οι παρακάτω:

- Danaos Enterprise
- SAP
- MarineSoft

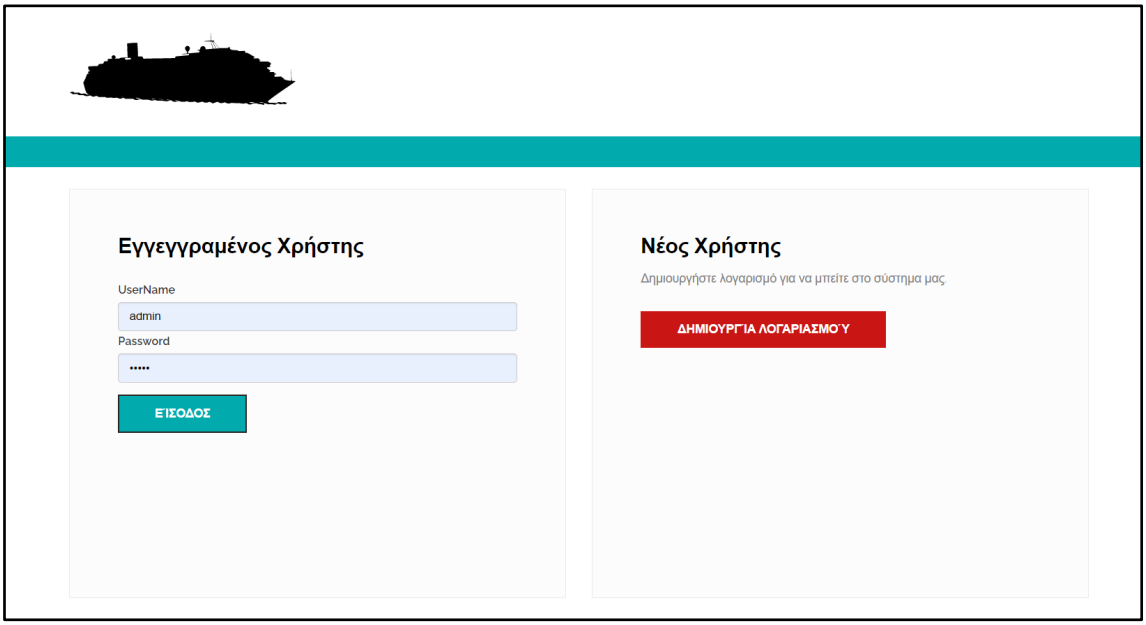
Το πρόβλημα με τις παραπάνω πλατφόρμες είναι ότι, παρότι προσφέρουν ένα ολόκληρο πακέτο λειτουργειών, έχουν μεγάλο κόστος, είναι δύσκολα στην εκπαίδευση και δεν έχουν πάντα φιλικό UI. Τονίζοντας πόσο σημαντική και γεμάτη υποχρεώσεις είναι η δουλειά ενός καπετάνιου, το πρόγραμμα χρειάζεται ένα πλοίο πρέπει να είναι όσο φιλικό στον χρήστη γίνεται και όσο πιο εύκολο στην κατανόηση και χρήση. Λαμβάνοντας τα παραπάνω υπόψιν, η πλατφόρμα μας δημιουργήθηκε με σκοπό να είναι διευκολύνει τον χρήστη στα παραπάνω προβλήματα με τον καλύτερο δυνατό τρόπο.

[11] https://findsoftwarenow.com/viewproduct.php?id=59

4. Παρουσίαση και χρήση εφαρμογής (User's manual)

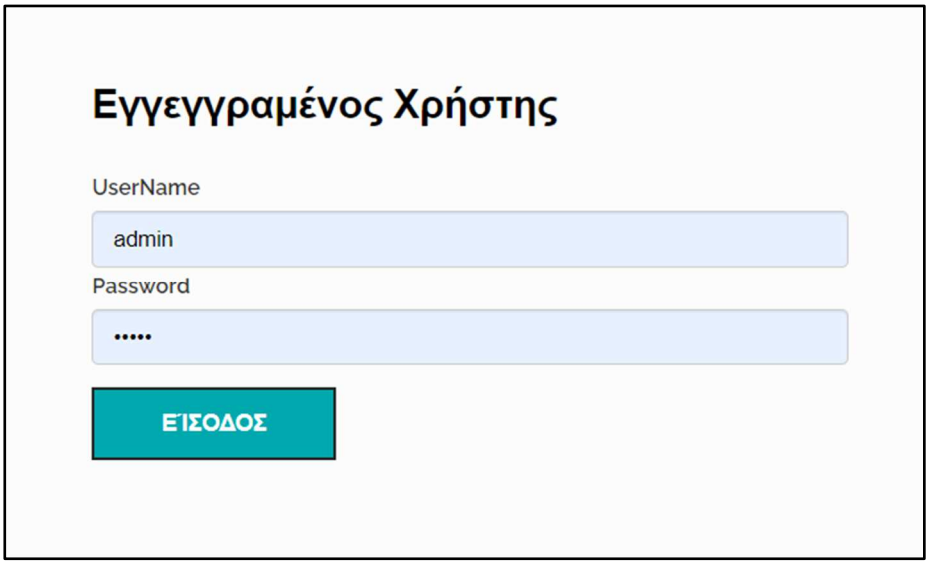
Ανοίγουμε την εφαρμογή μας αρχικά από το Visual studio. Όταν τρέξουμε το πρόγραμμα μας, ανοίγει η πλατφόρμα μας σε παραθυρικό επίπεδο σε έναν local host.

Με το που συνδεθούμε στον local host ανοίγει το αρχικό παράθυρο της πλατφόρμας μας.



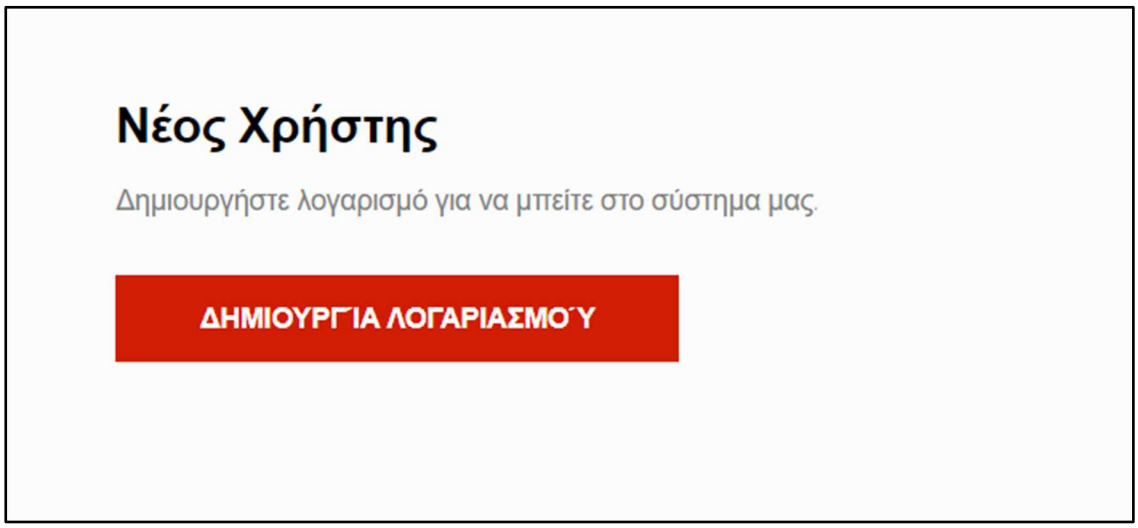
4.1 Login

Συνδεόμαστε με τα στοιχεία μας.



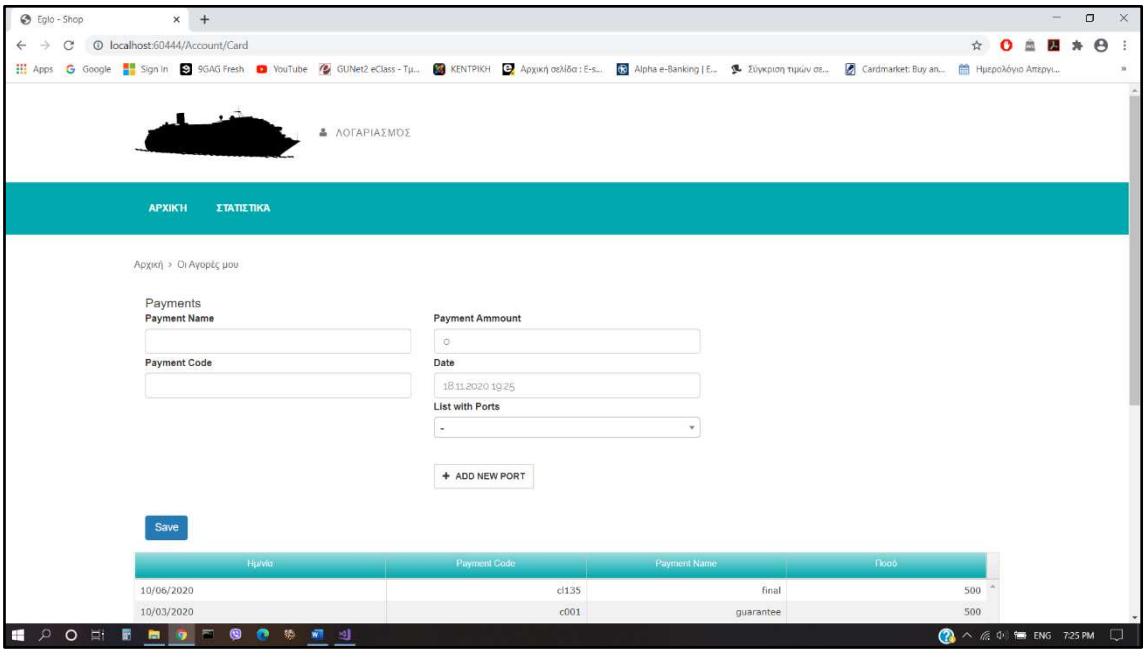
4.2 Register

Εάν δεν έχουμε στοιχεία πρόσβασης μπορούμε να επικοινωνήσουμε με το παρακάτω κουμπί και ο διακομιστής θα ενημερωθεί για το αίτημα μας για στοιχεία πρόσβασης.



4.3 Main Page

Αφού βάλουμε τα στοιχεία πρόσβασης μας και γίνει ο απαραίτητος έλεγχος από το σύστημα εισερχόμαστε στην αρχική μας σελίδα.

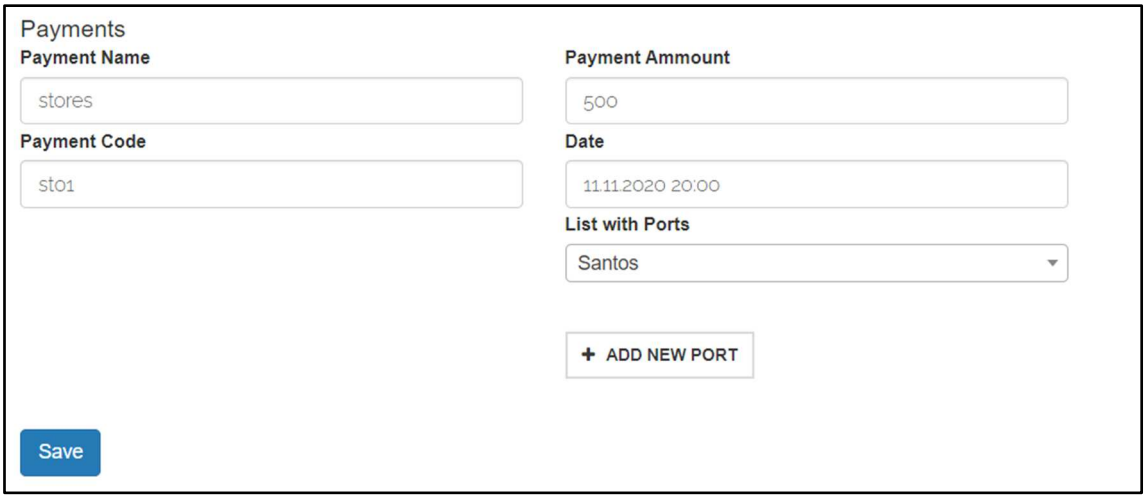


4.4 Add a new port

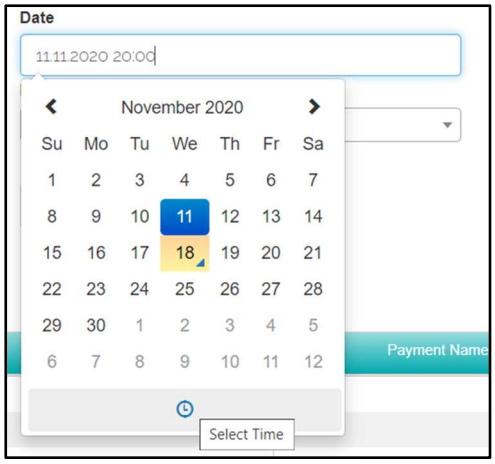
Πολύ εύκολα μπορούμε να προσθέσουμε στο σύστημα ένα καινούργιο λιμάνι με το παρακάτω κουμπί το οποίο θα μας εμφανίσει το παρακάτω αναδυόμενο παράθυρο.

4.5 Insert expenses

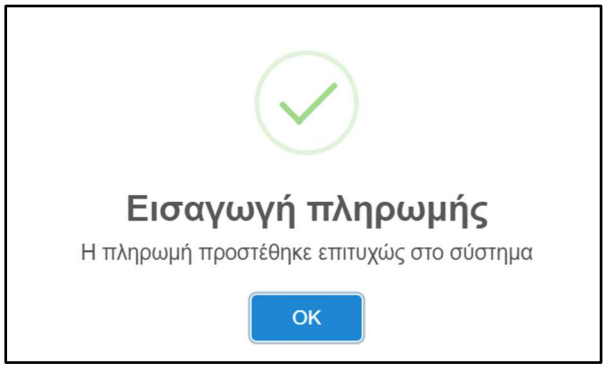
Με την παρακάτω φόρμα συμπληρώνουμε το όνομα του εξόδου μας, τον κωδικό του, το ποσό, την ημερομηνία που έγινε καθώς και σε ποιο λιμάνι.



Το πεδίο της ημερομηνίας δέχεται και συγκεκριμένη ώρα όπως φαίνεται στην παρακάτω εικόνα.

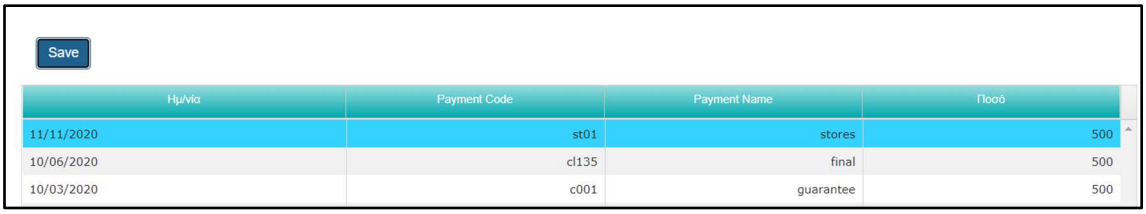


Με την εισαγωγή νου νέου εξόδου μας, το σύστημα μας ενημερώνει ότι η διαδικασία πραγματοποιήθηκε σωστά.



4.6 Expenses table

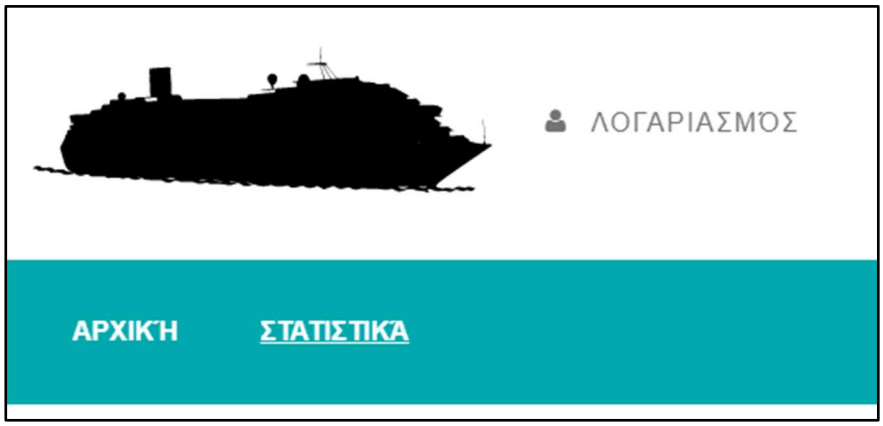
Στον πίνακα με όλα τα έξοδα που έχουμε δημιουργήσει παρατηρούμε ότι αυτόματα το νέο έξοδο μας έχει ενημερωθεί στην λίστα.



Επίσης παρατηρούμε ότι ο πίνακας μας διαθέτει μερικά βοηθητικά στοιχεία όπως εμφάνιση 100,200 ή 500 εξόδων την φορά, ένα ποσό του συνολικού εξόδου, έναν καταμετρητή εξόδων καθώς και ένα κουμπί ανανέωσης.

4.7 Statistics page

Στην αρχική μας σελίδα με το κουμπί “ΣΤΑΤΙΣΤΙΚΑ“ θα συνδεθούμε στην δεύτερη σελίδα μας με τα στατιστικά στοιχεία μας που προκύπτουν από τα έξοδα που έχουμε εισάγει στο σύστημα.



4.8 All-in-all Statistics

Η δεύτερη σελίδα μας, ανοίγει με ένα στατιστικό μοντέλο όλων των εξόδων που έχουμε εισάγει στο σύστημα. Μπορούμε να διακρίνουμε τα έξοδα μας ανά μηνά και το συνολικό ποσό ανά μηνά.

4.9 Filtered Statistics

Με τα ειδικά φίλτρα στην ίδια σελίδα όμως, μπορούμε να αντλήσουμε παραπάνω από μια πληροφορίες. Μπορούμε για παράδειγμα να δούμε ένα στατιστικό μοντέλο, μόνο για ένα κωδικό εξόδου και μόνο. Εναλλακτικά μπορούμε να δούμε όλα τα είδη εξόδων σε ένα μόνο λιμάνι ή μπορούμε να ένα συνδυασμό και των δυο παραπάνω.

Ένα επιπλέον πολύτιμο δεδομένο που μας δίνει το σύστημα μας είναι η συχνότητα των εξόδων μας ανά χρονική περίοδο. Αυτό είναι ένα πολύτιμο στοιχείο για μελλοντική χρήση πάνω σε διαχείριση επίλογων και αποφάσεων.

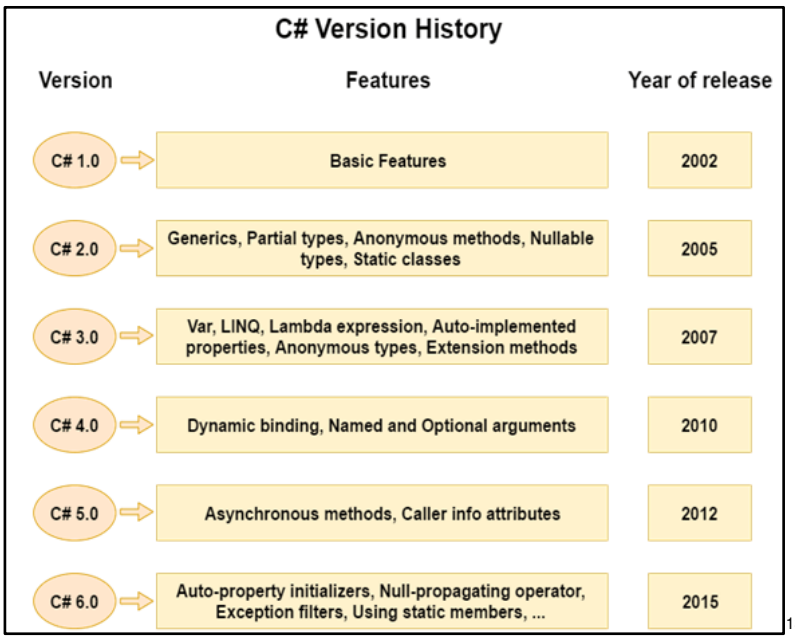
5. Αρχιτεκτονική συστήματος

Η πλατφόρμα μας δημιουργήθηκε με την χρήση του Visual Studio, ενός εργαλείου προγραμματισμού της Microsoft. Έχει χρησιμοποιηθεί η γλώσσα C# με χρήση του Entity Framework το οποίο ακολουθεί το μοντέλο MVC. Ενδεικτικά αναφέρουμε μερικά από τα χαρακτηριστικά των ανωτέρω στοιχείων:

5.1 Η γλώσσα προγραμματισμού C#

Η C# είναι κατά κύριο λόγο μια αντικειμενοστρεφής γλώσσα προγραμματισμού που ενσωματώνει ωστόσο μερικά χαρακτηριστικά διαφόρων προγραμματιστικών προτύπων. Αναπτύχθηκε στη Microsoft, από μια ομάδα κάτω από την ηγεσία του Anders Hejlberg, σαν μέρος του .NET Framework. Η C# από τα θεμέλια της σχεδιάστηκε να είναι αντικειμενοστρεφής. Ο αντικειμενοστρεφής προγραμματισμός επικράτησε σαν προγραμματιστικό πρότυπο την προηγούμενη δεκαετία και παραμένει στις πρώτες προτιμήσεις των προγραμματιστών. Οι ανάγκες για κατανεμημένα συστήματα πελάτη-εξυπηρέτη συμπίπτουν με την ενθυλάκωση και την ανταλλαγή μηνυμάτων που είναι βασικά χαρακτηριστικά του αντικειμενοστρεφούς προγραμματισμού. Κατά πολλούς ειδικούς στις γλώσσες προγραμματισμού, η επιτυχής λειτουργία των προγραμματιστικών συστημάτων σε δικτυακά περιβάλλοντα αυξανόμενης πολυπλοκότητας βασίζεται στην αντικειμενοστρέφεια.

Η C# παρέχει μια ξεκάθαρη και αποδοτική αντικειμενοστρεφή πλατφόρμα παρέχοντας στους προγραμματιστές μια συλλογή βιβλιοθηκών δοκιμασμένων αντικειμένων που παρέχουν λειτουργικότητα που ποικίλει από απλούς τύπους δεδομένων, σε διεπαφές εισόδου/εξόδου ή δικτυακές και εργαλεία για τη δημιουργία παραθυρικών εφαρμογών. Αυτές οι βιβλιοθήκες μπορούν να προσαρμοστούν στις ανάγκες του προγραμματιστή. Επιπρόσθετα η C# υποστηρίζει και τον προγραμματισμό βασισμένο σε components (component-based programming) ο οποίος επιτρέπει τον προσδιορισμό αυτόνομων μονάδων λειτουργικότητας (components) που είναι απομονωμένα και τεκμηριωμένα, παρουσιάζοντας ένα μοντέλο με ιδιότητες, μεθόδους, events και μετα δεδομένα για το component. Η C# υποστηρίζει αυτά τα χαρακτηριστικά άμεσα κάνοντας έτσι τη διαδικασία δημιουργίας και χρήσης των components πολύ εύκολη.[12]



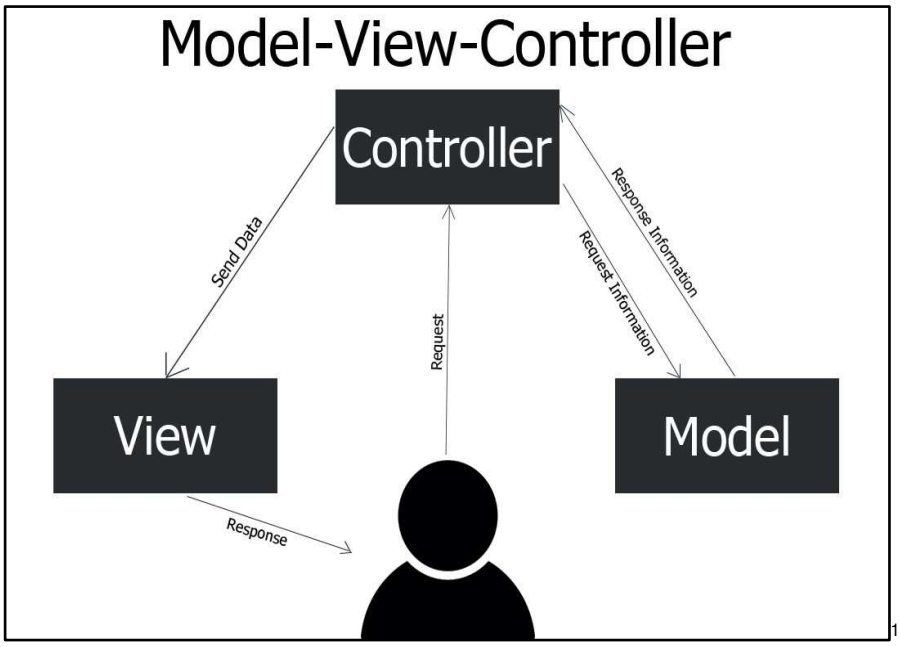
[13]

[12] Γκουβας Σωτηριος 2012 «ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΤΗΝ ΓΡΑΦΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΣΩΜΑΤΟΣ ΜΕ ΥΠΟΣΤΗΡΙΞΗ ΠΟΛΛΑΠΛΗΣ ΑΛΛΗΛΕΠΙΔΡΑΣΗΣ ΧΡΗΣΤΗ.»

[13] https://www.javatpoint.com/csharp-history

5.2 MVC Framework

Βασιζόμενο στο MVC Pattern, το οποίο είναι ένα μοντέλο αρχιτεκτονικής λογισμικού, το ASP.NET δημιουργεί τη λειτουργικότητα της πλατφόρμας. Το pattern αυτό χωρίζει τη βασική λειτουργικότητα σε τρία κομμάτια, αυτό του Model, του View και του Controller. Αυτά τα τρία κομμάτια συνεργάζονται μεταξύ τους για να πάρουν το input από τον χρήστη και να παράγουν κάποιο response σε αυτόν. Η παρακάτω εικόνα δείχνει μια γραφική αναπαράσταση του πως συνεργάζονται τα τρία αυτά στοιχεία.[14]



[15]

Ουσιαστικά ο τρόπος λειτουργίας ενός τέτοιου συστήματος είναι ως εξής:

- Ο χρήστης βλέπει το interface μιας πλατφόρμας μέσω του View. Όλο το interaction, τα κουμπιά, οι φόρμες κ.α. βρίσκονται μέσα σε αυτό.
- Όταν ο χρήστης πατήσει κάποιο από τα links ή τα κουμπιά, πληροφορία για το τι πατήθηκε, που ουσιαστικά σημαίνει το τι θέλει να κάνει ο χρήστης, στέλνεται στους αντίστοιχους Controllers που είναι συνδεδεμένοι με το αντίστοιχο interface.
- Οι Controllers επεξεργάζονται την πληροφορία που έστειλε ο χρήστης, και στη συνέχεια επεξεργάζονται τα Models που υπάρχουν σε αυτή την εφαρμογή. Τα Models αυτά, είναι η βάση δεδομένων και ότι άλλες κλάσεις χρειάζονται για τη λειτουργία της πλατφόρμας.
- Τέλος, μόλις ενημερωθούν και τα Models, ενημερώνεται το νέο View που βλέπει ο χρήστης, πάλι μέσω των αντίστοιχων Controllers που στέλνουν την απαραίτητη πληροφορία.[16]

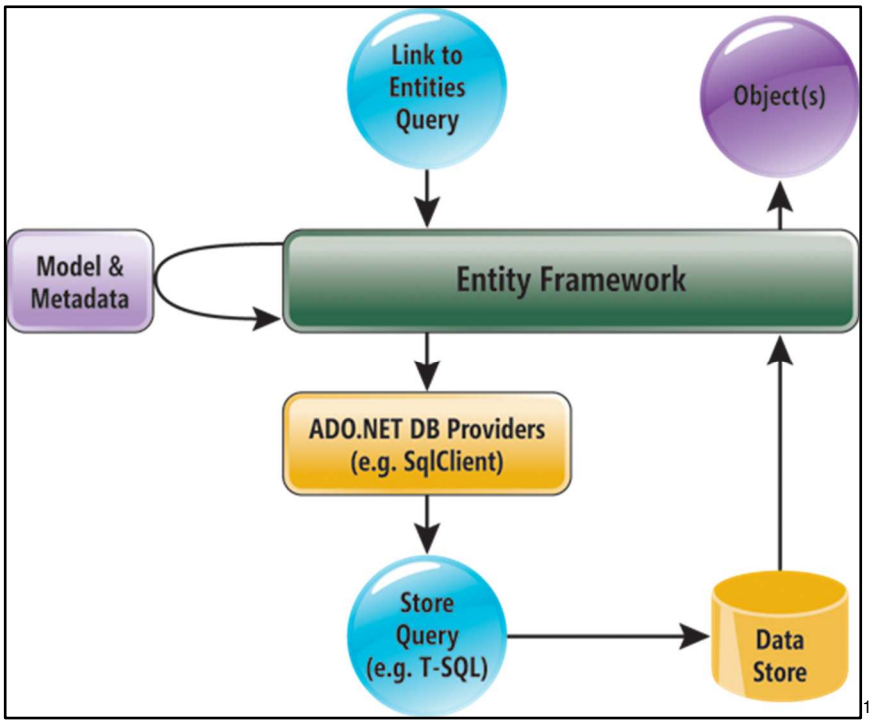
[14] https://apothesis.lib.hmu.gr/bitstream/handle/20.500.12688/9059/LiapisChristos2018.pdf?sequence=1&isAllowed=y

[15] https://www.mvps.net/docs/what-is-model-view-controller/

[16] https://apothesis.lib.hmu.gr/bitstream/handle/20.500.12688/9059/LiapisChristos2018.pdf?sequence=1&isAllowed=y

5.3 Entity Framework

Πριν από την έκδοση .NET 3.5 οι προγραμματιστές αναγκάζονταν να γράψουν επιπλέον κώδικα για τη διαχείριση της βάσης δεδομένων, ανοίγοντας μία σύνδεση με τη βάση, δημιουργώντας DataSet για τη ανάγνωση ή εγγραφή δεδομένων. Η διαδικασία αυτή ήταν επίπονη για τους προγραμματιστές κι επιρρεπής σε σφάλματα. Για το λόγο αυτό η Microsoft δημιούργησε το Entity Framework που αυτοματοποιεί όλες τις σχετικές δραστηριότητες μεταξύ βάσης δεδομένων και της εφαρμογής. Ο επίσημος ορισμός που δίνει η Microsoft είναι πως το Entity Framework είναι ένα object-relational mapping (ORM) framework που επιτρέπει στους προγραμματιστές .NET να εργαστούν με σχεσιακά δεδομένα χρησιμοποιώντας domain-specific αντικείμενα (Microsoft, 2018). Το Entity Framework επιτρέπει τη δημιουργία ενός μοντέλου γράφοντας κώδικα ή χρησιμοποιώντας κουτιά και γραμμές στον EF Designer. Και οι δύο προσεγγίσεις μπορούν να χρησιμοποιηθούν σε μία υπάρχουσα βάση δεδομένων ή για τη δημιουργίας μίας νέας.[17]



[18]

[17] https://apothesis.lib.hmu.gr/bitstream/handle/20.500.12688/9059/LiapisChristos2018.pdf?seq uence=1&isAllowed=y

[18] https://docs.microsoft.com/en-us/archive/msdn-magazine/2010/august/msdn-magazine-data-points-deny-table-access-to-the-entity-framework-without-causing-a-mutiny

6. Συμπεράσματα και μελλοντικές επεκτάσεις

Στην παρούσα εργασία δημιουργήσαμε μια πλατφόρμα η οποία συνδυάζει δυο από τους μεγαλύτερους κλάδους. Τον κλάδο της οικονομίας (συγκεκριμένα την ναυτιλιακή εξειδίκευση) και τον κλάδο της πληροφορικής. Οι σύγχρονες οικονομικές συνθήκες, οι οικονομικές, κοινωνικές, αναπτυξιακές και περιβαλλοντικές προκλήσεις απαιτούν να βρισκόμαστε συνεχώς σε εγρήγορση, να είμαστε και να μπορούμε να ανταποκριθούμε στις συνεχώς αυξανόμενες απαιτήσεις. Με την πάροδο των χρόνων, είναι αναπόφευκτο οι ανάγκες και οι διαθέσιμες επιλογές να διαφοροποιούνται συνεπώς η αγορά πρέπει να είναι έτοιμη να ανταπεξέλθει και να ικανοποιήσει τις ανάγκες των καταναλωτών προσανατολισμένη πάντα στις σύγχρονες εξελίξεις.

Σκοπός μας, λοιπόν, ήταν να χρησιμοποιήσουμε την τεράστια γκάμα εργαλείων της πληροφορικής με σκοπό να λύσουμε προβλήματα, όπως τα παραπάνω, που αναδύονται στην ναυτιλία και στην οικονομία γενικότερα. Πέραν την χρηστικότητας της πλατφόρμας μας, θέλαμε να δείξουμε πως κανένας κλάδος δεν πρέπει να είναι μονοδιάστατος. Μόνο η ταυτόχρονη χρήση επιστημών και εργαλείων μπορούν να λύσουν τα προβλήματα που αντιμετωπίζει καθημερινά μια επιχείρηση ή ένας άνθρωπος. Μια πληθώρα εργαλείων δεν είναι τίποτα αν δεν υπάρχει ένα πρόβλημα να χρησιμοποιηθούν και αντιστοίχως μια θεωρία δεν μπορεί να είναι πυλώνας στην κοινωνία μας αν δεν υπάρχουν τρόποι και μέθοδοι για να υλοποιηθεί στην πράξη.

Η πλατφόρμα μας, στηριζόμενη πάνω στις βασικές αρχές και εργαλεία της πληροφορικής, έρχεται να αναλύσει, στατιστικά και παρουσιαστικά, οικονομικά και λογιστικά στοιχεία με πολλά αποτελέσματα και απόρροιες. Βασίζεται στον ανθρώπινο παράγοντα και προσπαθεί ναι είναι όσο περισσότερο φιλική στον χρήστη γίνεται. Και όπως έχουμε αναφέρει παραπάνω, όπως τα προβλήματα σε έναν κλάδο δεν είναι σταθερά αλλά μεταβλητά, έτσι και η πλατφόρμα μας μπορεί να έχει πολλές μελλοντικές επεκτάσεις με σκοπό την πρόβλεψη και επίλυση μελλοντικών προβλημάτων. Για παράδειγμα μπορούμε να δημιουργήσουμε παραπάνω από ένα στατιστικό μοντέλο για περεταίρω οικονομικές πληροφορίες όπως ένα διάγραμμα με ποσοστιαίες μεταβολές εξόδων ανά χώρα αντί για μια μόνο ποσοτική παρουσίαση. Επίσης μπορούμε να συνδέσουμε στην πλατφόρμα μας το στοιχείο του ανταγωνισμού. Συγκρίνοντας τα έξοδα που ήδη έχουμε καταγράψει σε ένα λιμάνι, μπορούμε να διαπραγματευτούμε καλύτερες τιμές με άλλο προμηθευτή σε επόμενο ή και στο ίδιο ταξίδι του πλοίου.

Από πληροφοριακής άποψης, θα μπορούσαμε να εξελίσσουμε το σύστημα μας ώστε συγκρίνοντας τα έξοδα ώστε να μας προτείνει αυτό μια πιθανή λύση ή μια πιθανή εξοικονόμηση χρημάτων. Το σύστημα θα μπορεί να αναλύει ήδη υπάρχοντα δεδομένα και να προτείνει την βέλτιστη λύση κόστους.

Βλέποντας το θέμα από την αντίθετη μεριά, η πλατφόρμα μας, και όλες οι πλατφόρμες τέτοιου είδους, μπορούν να βοηθήσουν να κατανοήσουμε τα κοινά προβλήματα που υπάρχουν σε ένα μοντέλο με σκοπό να χρησιμοποιήσουμε την πληροφορική για να τα λύσουμε με τον καλύτερο δυνατό τρόπο. πολλές φορές αλληλεπίδραση με τον χρήστη και η ανταλλαγή πληροφοριών με αυτόν, μας δίνουν τις πιο σημαντικές βάσεις για να αναπτύξουμε ένα πληροφοριακό σύστημα που θα μπορεί να είναι ενημερωμένο με τις ανάγκες της σύγχρονης κοινωνίας.

7. Βιβλιογραφία

7.1 Συγγράμματα

- Βλάχος Γ.- Αλεξόπουλος Α. (1996), Διεθνείς Οργανισμοί και Ναυτιλιακή Πολιτική, Εκδόσεις Α. Σταμούλη, Πειραιάς
- 1Βλάχος Γ.Π. – Γεωργαντόπουλος (2003), Ναυτιλιακή Οικονομική», Εκδόσεις Τζέι&Τζέι Ελλάς, Πειραιάς
- ΣΩΤΗΡΙΑ Ι. ΕΛΕΥΘΕΡΙΟΥ Λογιστής Φοροτεχνικός Α’ Τάξης & Ειδικός Εισηγητής Λογιστικής ΥΜΗΤΤΟΥ 75, 11633 ΠΑΓΚΡΑΤΙ – Τ/Φ: 211-0149320 www.sotele.gr
- 1Γκουβας Σωτηριος 2012 «ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΤΗΝ ΓΡΑΦΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΣΩΜΑΤΟΣ ΜΕ ΥΠΟΣΤΗΡΙΞΗ ΠΟΛΛΑΠΛΗΣ ΑΛΛΗΛΕΠΙΔΡΑΣΗΣ ΧΡΗΣΤΗ.»

7.2 Ιστοσελίδες

- http://www.ics-shipping.org/shipping-facts/shipping-and-world-trade
- https://www.predictiveanalyticstoday.com/top-free-and-open-source-accounting-software/
- https://quickbooks.intuit.com/uk/
- https://quickbooks.intuit.com/global/cloud-accounting-software/
- https://quickbooks.intuit.com/uk/accounting-software/
- https://en.wikipedia.org/wiki/FreshBooks
- https://www.beginner-bookkeeping.com/business-invoicing-software.html
- https://findsoftwarenow.com/viewproduct.php?id=59
- https://www.javatpoint.com/csharp-history
- https://apothesis.lib.hmu.gr/bitstream/handle/20.500.12688/9059/LiapisChristos2018.pdf?seq uence=1&isAllowed=y
- https://docs.microsoft.com/en-us/archive/msdn-magazine/2010/august/msdn-magazine-data-points-deny-table-access-to-the-entity-framework-without-causing-a-mutiny
- https://apothesis.lib.hmu.gr/bitstream/handle/20.500.12688/9059/LiapisChristos2018.pdf?sequence=1&isAllowed=y
- https://www.mvps.net/docs/what-is-model-view-controller/
- https://apothesis.lib.hmu.gr/bitstream/handle/20.500.12688/9059/LiapisChristos2018.pdf?sequence=1&isAllowed=y

8. Κώδικας

8.1 Controllers

8.1.1 AccountController.cs

Σε αυτό το αρχείο για να εμφανίσουμε τα στοιχεία που είναι αποθηκευμένα στην βάση ανάλογα με τον χρήστη.

```
  public ActionResult SubmitPayment (UserModel payment)
        {

            int UserID = ClsSessions.UserID;
            try
            {
                using (Database1Entities2 db = new Database1Entities2())
                {

                    try
                    {
                        tablePayments tmppayment = new tablePayments();
                        tmppayment.PaymentCode = payment.paymentcode;
                        tmppayment.PaymentName = payment.paymentname;
                        tmppayment.Ammount = payment.paymentammount;
                        tmppayment.Date =  DateTime.Parse(payment.paymentdate);
                        tmppayment.UserID = ClsSessions.UserID;
                        tmppayment.PortID = payment.portID;
                        db.tablePayments.Add(tmppayment);

                        db.SaveChanges();
                    }
                    catch (Exception ex) { }

                    return new JsonResult() { Data = "Ok", JsonRequestBehavior =
JsonRequestBehavior.AllowGet };
                }
            }
            catch { }
            return new JsonResult() { Data = "Error", JsonRequestBehavior =
JsonRequestBehavior.AllowGet };
             }
```

8.1.2 HomeController.cs

Με τον παρακάτω κώδικα μέσα στο αρχείο του τίτλου κάνουμε τον έλεγχο του login

```
public ActionResult Login()
        {
            LoginModel user = new LoginModel();

            if (string.IsNullOrEmpty(ClsSessions.UserName))
            {
                return View(user);
            }
            else
            {
                return RedirectToAction("Dashboard", new { code =
ClsSessions.CustCode });
             }
```

Στο παρακάτω κομμάτι του ιδίου αρχείου τραβάμε δεδομένα από την βάση και φτιάχνουμε ένα session για μελλοντική χρήση.

```
public ActionResult Login(LoginModel credentials)
        {

            try
            {
                  TableLogin user = new TableLogin();
                using (Database1Entities2 db = new Database1Entities2())
                {
                    user = db.TableLogin.Where(x => x.UserName ==
credentials.username.ToLower().Trim() && x.Password ==
credentials.password).FirstOrDefault();
                    if (user != null)
                    {

                        ClsSessions.UserName = user.UserName;
                        ClsSessions.Password = user.Password;

                        ClsSessions.UserID = user.UserId;
                        ClsSessions.Role = user.Role;


                        return new JsonResult() { Data = "OK",
JsonRequestBehavior = JsonRequestBehavior.AllowGet };
                    }
                    else
                    {
                        return new JsonResult() { Data = "NOTFOUND",
JsonRequestBehavior = JsonRequestBehavior.AllowGet };
                    }
                }
            }
            catch (Exception e)
            {
                return new JsonResult() { Data = "NOTFOUND", JsonRequestBehavior
= JsonRequestBehavior.AllowGet };
            }
        }
```

8.1.3 StatisticsController.cs

Στο συγκεκριμένο αρχείο δημιουργούμε τα στατιστικά μας, με δεδομένα ανάλογα με τον χρήστη που είναι συνδεδεμένος και έχουμε και ένα groupby στις κατηγορίες εξόδων και στα λιμάνια.

```
public class StatisticsController : Controller
    {
        public ActionResult Index()
        {
            if (ClsSessions.UserName == null || ClsSessions.UserName == "")
            {
                return RedirectToAction("Login");
            }
            else
            {
                using (Database1Entities2 db = new Database1Entities2())
                {
                    UserModel model = new UserModel();
                    try
                    {
                        var paymentcode = db.tablePayments.Where(x => x.UserID ==
ClsSessions.UserID).GroupBy(x => x.PaymentCode).Select(x =>
x.FirstOrDefault()).ToList();
                        if (paymentcode != null)
                        {
                            var paymentfirst =
paymentcode.FirstOrDefault().PaymentCode;
                        }
                        var port = db.Ports.FirstOrDefault();
                        ClsSessions.PortID = -1;
                        ClsSessions.PayementCodeID = -1;
                    }
                    catch (Exception ex) { }
                    return View(model);
                }
        }
       public ActionResult RefreshChart()
        {
            UserModel model = new UserModel();
            return View("_Chart",model);
        }
        public ActionResult RefreshFreq()
        {
            return View("_Freq");
        }

    }
```

8.2 Models

8.2.1 UserModel.cs

Στο παρακάτω κομμάτι κώδικα από το αρχείου του τίτλου τραβάμε την λίστα των εξόδων για να τα χρησιμοποιήσουμε στο στατιστικό.

```
using (Database1Entities2 db = new Database1Entities2())
{
   List= db.tablePayments.Where(x => x.UserID ==
  ClsSessions.UserID).GroupBy(x => x.PaymentCode).Select(x =>
  x.FirstOrDefault()).ToList();
}


foreach (var item in List)
{
    KeyVal option = new KeyVal();
    option.Key = item.Id;
    option.Val = item.PaymentCode;
    ClassificationsTypes.Add(option);
}

ClassificationsTypes.Insert(0, new KeyVal
{
    Val = "-",
    Key = -1
});
```

Στο παρακάτω κομμάτι κώδικα τραβάμε από την βάση δεδομένων μας τα λιμάνια για το drop-down list

```
public static SelectList PortDropdown(int selected)
        {
            List<KeyVal> ClassificationsTypes = new List<KeyVal>();
            List<Ports> List = new List<Ports>();


            using (Database1Entities2 db = new Database1Entities2())
            {
                List = db.Ports.ToList();
            }


            foreach (var item in List)
            {
                KeyVal option = new KeyVal();
                option.Key = item.Id;
                option.Val = item.PortName;
                ClassificationsTypes.Add(option);
            }

            ClassificationsTypes.Insert(0, new KeyVal
            {
                Val = "-",
                Key = -1
            });
        }
```

8.3 Views

8.3.1 CardGrid.cshtml

Στο παρακάτω κομμάτι κώδικα από το αρχείου του τίτλου δημιουργούμε τον πίνακα των εξόδων που βλέπουμε στην αρχική σελίδα.

```
<script>
    var expanded = {};
    var grid, id;
    var WebRootPath = ''

    $(document).ready(function ()

        var WebRootPath = 'h'

        @*var postData = {
            "YearID": '@Model.YearID'
        };*@

        var WebRootPath = ''
        var element = $("#grid").kendoGrid({
            dataSource: {
                type: "aspnetmvc-ajax",
                transport: {
                    read: {
                        url: WebRootPath + "/Account/CustomersCardJson",
                        data: { curCode: '@code' }
                    }
                },
                EnableCustomBinding: false,
                serverPaging: false,
                serverSorting: false,
                serverFiltering: false,
                serverGrouping: false,

                pageSize: 100,

                schema: {
                    data: "Data",
                    total: "Total",
                },
                requestEnd: onRequestEnd,

                aggregate: [

                    { field: "totalAllString", aggregate: "sum" }
                ]
            },
            sortable: true,
            groupable: false,
            filterable: {
                mode: "row"
            },

            pageable: {
                refresh: true,
                pageSizes: [100, 200, 500],
                buttonCount: 5
            },
```

```
            columns:
                    [
                    { field: "datestr", encoded: false, title: "Ημ/νία", width:
"100px", sortable: true, filterable: false },

                    { field: "paymentcode", type: "string", title: "Payment
Code", width: "80px", sortable: true, filterable: false, attributes: { style:
"text-align: right;" } },

                    { field: "paymentname", type: "string", title: "Payment
Name", width: "80px", sortable: true, filterable: false, attributes: { style:
"text-align: right;" } },

                    { field: "paymentammount", type: "decimal", title: "Ποσό",
width: "80px", sortable: true, filterable: false, decimals: "2", footerTemplate:
"<div style='text-align:right'>#: data.totalAllString ? data.totalAllString.sum:
0 #</div>", attributes: { style: "text-align: right;" } },
                    ]
        });

    });
```

8.3.2 _Chart.cshtml

Στο παρακάτω κομμάτι κώδικα από το αρχείου του τίτλου δημιουργούμε τον το γράφημα μας το οποίο διαβάζει και παρουσιάζει τα έξοδα μας ανά μήνα ξεκινώντας το γράφημα με μηδενικά κόστη.

```
Layout = null;
    using (Database1Entities2 db = new Database1Entities2())
    {
        string PaymentCode = "";
        string PaymentName = "";
        int Count = 0;
        var datenow = DateTime.Now;
        var res = db.tablePayments.Where(x => x.UserID ==
ClsSessions.UserID).OrderByDescending(x => x.Date).ToList();
        res.Where(x => x.Date.Year.Equals(datenow.Year)).ToList(); //pairnw gia
to trexw etos gia na vrw syxnothta agoras
        var hidden = true;
        List<decimal> Cdata = new List<decimal>();
        decimal AmmountJanuary = 0;
        decimal AmmountFeb = 0;
        decimal AmmountMarc = 0;
        decimal AmmountApril = 0;
        decimal AmmountMay = 0;
        decimal AmmountJune = 0;
        decimal AmmountJuly = 0;
        decimal AmmountAugust = 0;
        decimal AmmountSep = 0;
        decimal AmmountOct = 0;
        decimal AmmountNov = 0;
        decimal AmmountDec = 0;
```

```
        if (ClsSessions.PayementCodeID != null && ClsSessions.PayementCodeID != -
1)
        {
            var payment = res.Where(x => x.Id ==
ClsSessions.PayementCodeID).FirstOrDefault();
            PaymentName = payment.PaymentName;
            PaymentCode = payment.PaymentCode;
            res = res.Where(x => x.PaymentCode == PaymentCode).ToList();
            hidden = false;
        }

        if (ClsSessions.PortID != null && ClsSessions.PortID != -1)
        {
            res = res.Where(x => x.PortID == ClsSessions.PortID).ToList();
            hidden = false;
        }
        foreach (var item in res)
        {
            if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.January)
            {

                AmmountJanuary = AmmountJanuary + item.Ammount;

            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.February)
            {

                AmmountFeb = AmmountFeb + item.Ammount;

            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.March)
            {

                AmmountMarc = AmmountMarc + item.Ammount;

            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.April)
            {

                AmmountApril = AmmountApril + item.Ammount;

            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.May)
            {

                AmmountMay = AmmountMay + item.Ammount;

            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.June)
            {

                AmmountJune = AmmountJune + item.Ammount;
```

```
            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.July)
            {

                AmmountJuly = AmmountJuly + item.Ammount;


            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.August)
            {

                AmmountAugust = AmmountAugust + item.Ammount;


            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.September)
            {

                AmmountSep = AmmountSep + item.Ammount;

            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.October)
            {

                AmmountOct = AmmountOct + item.Ammount;


            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.November)
            {

                AmmountNov = AmmountNov + item.Ammount;


            }
            else if (item.Date.Month ==
(int)MyProject.Models.UserModel.Months.December)
            {

                AmmountDec = AmmountDec + item.Ammount;


            }

        }
```

8.3.3 _Freq.cshtml

Το παρακάτω κομμάτι του αρχείου μας δίνει την συχνότητα των εξόδων, κάνοντας ένα count των εξόδων και διαιρώντας το με τον αριθμό των ημέρων του έτους και στο τέλος μας βγάζει το αποτέλεσμα στην περιγραφή.

```
       Layout = null;
    using (Database1Entities2 db = new Database1Entities2())
    {
        string PaymentCode = "";
        string PaymentName = "";
        int Count = 0;
        var datenow = DateTime.Now;
        var res = db.tablePayments.Where(x => x.UserID ==
ClsSessions.UserID).OrderByDescending(x => x.Date).ToList();
        res.Where(x => x.Date.Year.Equals(datenow.Year)).ToList(); //pairnw gia
to trexw etos gia na vrw syxnothta agoras
        var hidden = true;
        if (ClsSessions.PayementCodeID != null && ClsSessions.PayementCodeID != -
1)
        {
            var payment = res.Where(x => x.Id ==
ClsSessions.PayementCodeID).FirstOrDefault();
            PaymentName = payment.PaymentName;
            PaymentCode = payment.PaymentCode;
            res = res.Where(x => x.PaymentCode == PaymentCode).ToList();
            hidden = false;
        }

        if (ClsSessions.PortID != null && ClsSessions.PortID != -1)
        {
            res = res.Where(x => x.PortID == ClsSessions.PortID).ToList();
            hidden = false;
        }
        Count = res.Count();
        if(Count == 0)
        {
            hidden = true;
        }
        else
        {
            Count = 365 / Count;
        }


<div>

    <br>
    @if (hidden == false)
    {
        <label>Συχνότητα αγοράς</label>
        <p style="font-size:1em">Η συναλλαγή με κωδικό: <b>@PaymentCode </b> και
περιγραφή: <b>@PaymentName</b>  γίνεται ανά <b>@Count</b>   μέρες </p>
    }

</div>

    }
```