ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε

Ανάλυση, Σχεδιασμός και Υλοποίηση Ηλεκτρονικού Καταστήματος Πωλήσεων Προϊόντων Υψηλής Τεχνολογίας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Του

ΓΕΩΡΓΙΛΑ Α. ΧΑΡΑΛΑΜΠΟΥ

Επιβλέπων: Αριστομένης Θανόπουλος

Σπάρτη, Οκτώβριος 2015

Τεχνολογικό εκπαιδευτικό ίδρυμα Πελοποννήσου
Σχολή τεχνολογικών εφαρμογών
Τμήμα μηχανικών πληροφορικής Τ.Ε

Υπεύθυνη Δήλωση

Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας, και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρεις τις όποιες πηγές από τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής Τ.Ε του ΤΕΙ Πελοποννήσου.

(Υπογραφή)

Χαράλαμπος Γεωργιλάς

Περίληψη

Η πτυχιακή εργασία πραγματεύεται την ανάλυση και τον σχεδιασμό ενός πληροφοριακού συστήματος ηλεκτρονικού εμπορίου. Η ανάπτυξη της τεχνολογίας και η διάδοση του διαδικτύου καθώς και η εμπιστοσύνη του αγοραστικού κοινού στις τεχνικές διασφάλισης των προσωπικών δεδομένων, όσο αναφορά τις διαδικτυακές συναλλαγές, δημιούργησαν τις κατάλληλες συνθήκες ώστε το διαδικτυακό εμπόριο να γνωρίσει μεγάλη ανάπτυξη τις τελευταίες δεκαετίες. Αυτό είχε σαν αποτέλεσμα την ανάπτυξη ποικίλων πληροφοριακών συστημάτων που έχουν σχέση με το ηλεκτρονικό εμπόριο τα οποία εξελίχθηκαν ραγδαία τα τελευταία χρόνια προσφέροντας εναλλακτικούς τρόπους συναλλαγών και μιας διαφορετικής αντίληψης στην αγορά του εμπορίου συνολικά.

Στόχος της πτυχιακής εργασία είναι η ανάλυση, ο σχεδιασμός και η ανάπτυξη ενός τέτοιου συστήματος από την αρχή αναλύοντας τα στάδια α) της δημιουργίας της υποδομής για την υποστήριξη μίας τέτοιας πλατφόρμας, β) την υλοποίηση της βάσης και του συστήματος με την χρήση τεχνικών σχεδίασης κώδικα και ενός πλαισίου – σκελετού (framework) γ) την ανάπτυξη διεπαφής χρήστη με την βοήθεια των τεχνολογιών HTML5, CSS 3, Javascript. δ) Την εξασφάλιση της ασφαλούς χρήσης των λειτουργιών του συστήματος. ε) Τον έλεγχο των υπηρεσιών για τυχών σφάλματα και αποσφαλμάτωση τους.

Λέξεις κλειδιά
Πληροφοριακό σύστημα ηλεκτρονικού εμπορίου, E-commerce, Databases, SQL, PHP, Javascript, Laravel, HTML.

Abstract

The thesis deals with the analysis and design of an IT e-commerce system. The development of technology the spread of the Internet and the confidence of buyers in the safeguarding of personal data in online transactions has created the right conditions for the online trade to experience strong growth in recent decades. This resulted in the development of various information systems related to E-commerce which have developed rapidly in recent years offering alternative patterns of trade and a different perception in the trade market as a whole.

The aim of the diploma work is the analysis, design and development of such a system from the start by analyzing the steps of a) creating the infrastructure to support such a platform, b) the implementation of the database and system using code design patterns and a framework c) the user interface development with the aid of technologies like HTML5, CSS 3, Javascript. d) Ensuring the safe use of system functions. e) Errors and debugging.

Keywords
E-commerce platform, E-commerce, Databases, SQL, PHP, Javascript, Laravel, HTML.

Στους γονείς μου.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Αριστομένη Θανόπουλο για την επίβλεψη, την καθοδήγηση και την ευκαιρία που μου έδωσε να εκπονήσω αυτή την πτυχιακή εργασία. Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου που μου παρείχε στηρίξει καθ' όλη την διάρκεια των σπουδών μου.

Περιεχόμενα

1) Κεφάλαιο 1

1.1 Εισαγωγή

Στην εποχή της έκρηξης της τεχνολογίας και καθώς όλο και περισσότερες υπηρεσίες παρέχονται πλέον με κάποιο τρόπο σε ψηφιακή μορφή, παρατηρούμε ότι με τα χρόνια δημιουργήθηκε μία σχέση αλληλεξάρτησης ανάμεσα στην καθημερινότητα μας και την τεχνολογία. Το διαδίκτυο έχει καταφέρει σε πολύ σύντομο χρονικό διάστημα, αναφορικά με άλλες τεχνολογίες (λ.χ. Τηλέφωνο, Τηλεόραση, Ράδιο), να προσπεράσει τα τεχνητά σύνορα των χωρών και της κουλτούρας και έχει καταφέρει να παρέχει ένα διαδραστικό και προσιτό για αρκετές ηλικίες μέσο επικοινωνίας, συγκέντρωσης πληροφοριών και κοινωνικοποίησης.

Ένας από τους τομείς που γνώρισαν μεγάλη άνθηση τις τελευταίες δύο δεκαετίες είναι το ηλεκτρονικό εμπόριο. Το εμπόριο είναι άρρηκτα συνδεδεμένο με την οικονομία και η οικονομία με την κοινωνία, ήταν λοιπόν φυσικό επόμενο με την τεχνολογική ανάπτυξη να δημιουργηθούν δομές για την εκμετάλλευση των ευκαιριών που προσφέρει το ηλεκτρονικό εμπόριο το οποίο, αν και δεν βρίσκεται στο προσκήνιο πολύ καιρό, έχει αφήσει το αποτύπωμα του στις σύγχρονες συναλλαγές κυρίως για την ευκολία την ευχρηστία και τους εναλλακτικούς τρόπους πληρωμής που προσφέρει. Όλα αυτά συνέβαλλαν στην δημιουργία μίας αγοράς η όποια υπολογίζεται ότι μπορεί να φτάσει, μέχρι το 2020, σε κέρδη που θα αγγίζουν το ένα τρις δολάρια, όταν το 2014 τα κέρδη της αγοράς ήταν διακόσια τριάντα δις δολάρια

1.2 Ιστορική αναδρομή

Το ηλεκτρονικό εμπόριο, αν και γνώρισε μεγάλη ανάπτυξη από τα μέσα της δεκαετίας του ενενήντα, στην πραγματικότητα απασχολεί την πληροφορική και τα δίκτυα από την δεκαετία του εβδομήντα. Φυσικά δεν είχε την σημερινή του μορφή και συνήθως έβρισκε εφαρμογή σε εσωτερικά δίκτυα μεγάλων εταιριών για την διευκόλυνση και την κάλυψη των αναγκών ή των υποχρεώσεων τους σε άλλες εταιρίες ή οργανισμούς . Αυτή η εφαρμογή του ηλεκτρονικού εμπορίου ονομάζεται ηλεκτρονική ανταλλαγή δεδομένων (Electronic Data Interchange – EDI) και θεωρείται η απαρχή του B2B commerce , δηλαδή του εμπορίου με την χρήση της τεχνολογίας ανάμεσα σε επιχειρήσεις.

Έτσι στις αρχές της δεκαετίας του ενενήντα και συγκεκριμένα το 1994 ο τότε τριαντάχρονος Jeff Bezos ξεκίνησε να διερευνά την εφαρμογή του ηλεκτρονικού εμπορίου στο ευρύ καταναλωτικό κοινό, καθώς το διαδίκτυο άρχισε να χρησιμοποιείτε ευρέως και στις οικίες. Έτσι δημιούργησε ένα ηλεκτρονικό κατάστημα (Cadabra.com) από το γκαράζ του σπιτιού του. Μέσα σε τρία μόλις χρόνια κατάφερε να έχει κέρδη εκατόν πενήντα εκατομμυρίων δολαρίων στην αναπτυσσόμενη αγορά. Το κατάστημα αυτό μετονομάστηκε και είναι πλέον γνωστό σαν το πασίγνωστο Amazon.com το οποίο είναι το μεγαλύτερο κατάστημα που προσφέρει συναλλαγές ανάμεσα σε έμπορους και χρήστες (B2C commerce).

1.3 Αντικείμενο της πτυχιακής εργασίας

Η πτυχιακή εργασία έχει ως αντικείμενο τον σχεδιασμό, την ανάλυση και την υλοποίηση μίας πλατφόρμας ηλεκτρονικού εμπορίου. Συγκεκριμένα θα αναλυθούν τα στάδια α) της δημιουργίας της υποδομής για την υποστήριξη μίας τέτοιας πλατφόρμας, β) την υλοποίηση της βάσης και του συστήματος με την χρήση τεχνικών σχεδίασης κώδικα και ενός πλαισίου – σκελετού (framework) γ) την ανάπτυξη διεπαφής χρήστη με την βοήθεια των τεχνολογιών HTML5, CSS 3, Javascript. δ) Την εξασφάλιση της ασφαλούς χρήσης των λειτουργιών του συστήματος. ε) Τον έλεγχο των υπηρεσιών για τυχών σφάλματα και αποσφαλμάτωση τους.

2) Κεφάλαιο 2

2.1 Ανάλυση και σχεδιασμός υποδομής

2.1.1 Εξυπηρετητής

Εφόσον η εφαρμογή μας βρίσκεται στο διαδίκτυο θα πρέπει να φιλοξενείτε σε ένα υπολογιστικό σύστημα το οποίο θα είναι προσβάσιμο από τρίτους υπολογιστές που βρίσκονται σε κάποιο εξωτερικό δίκτυο, για να επιτύχουμε κάτι τέτοιο χρειάζεται να παραμετροποιήσουμε το σύστημα μας και να εγκαταστήσουμε κάποιον διακομιστή διαδικτύου (web server), ανάλογα με την γλώσσα που θα χρησιμοποιήσουμε.

Το σύστημα που θα υλοποιήθηκε γράφτηκε σε γλώσσα PHP και γι' αυτόν τον λόγο θα χρησιμοποιήσουμε τον πιο διαδεδομένο web server, συγκεκριμένα τον Apache HTTP server. Η Apache έχει αναπτύξει και άλλους διακομιστές οι οποίοι λειτουργούν με άλλες γλώσσες όπως ο Apache Tomcat ο οποίος «σερβίρει» περιεχόμενο που έχει γραφτεί σε γλώσσα προγραμματισμού Java.

Εγκατάσταση εξυπηρετητή Apache

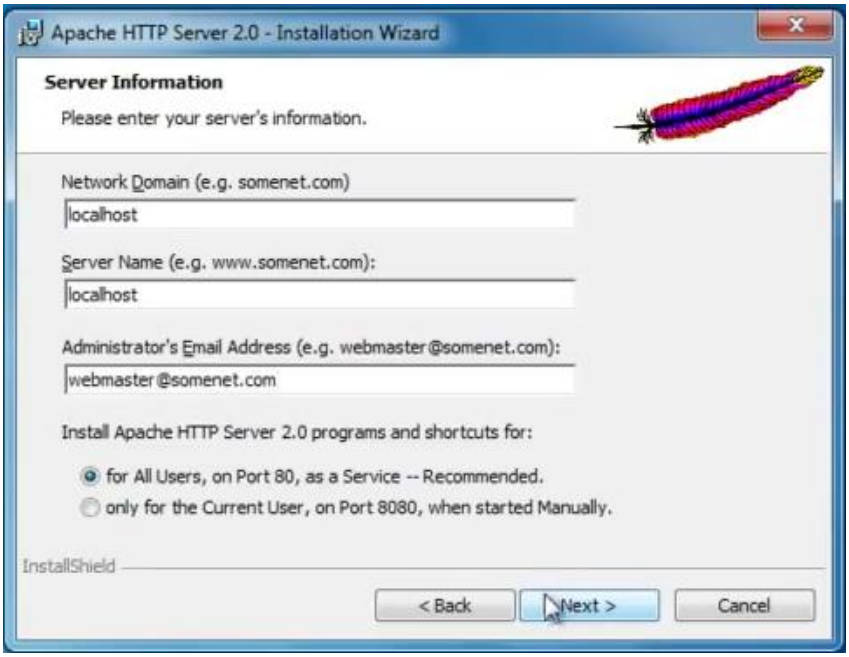
Σε λειτουργικό σύστημα Windows

Για να κάνουμε εγκατάσταση του Apache σε λειτουργικό σύστημα windows πρέπει να κατεβάσουμε και να εγκαταστήσουμε το αντίστοιχο .msi αρχείο από την επίσημη ιστοσελίδα του apache. Δυστυχώς η apache αποφάσισε να μην υποστηρίζει πλέον τους installers για windows και η τελευταία έκδοση η οποία υπάρχει σε μορφή εκτελέσιμου αρχείου για Windows είναι η 2.2.25 την οποία μπορούμε να προμηθευτούμε από το αρχειακό αποθετήριο δεδομένων του apache στον παρακάτω σύνδεσμο.

```
https://archive.apache.org/dist/httpd/binaries/win32/
```

Αφού επιλέξουμε το αρχείο « httpd-2.2.25-win32-x86-openssl-0.9.8y.msi », το ανοίγουμε ώστε να ξεκινήσει η εγκατάσταση. Αφού αποδεχτούμε τους όρους

χρήσης και διαβάσουμε το ενημερωτικό σημείωμα για το τι είναι ο apache και ποιες αλλαγές έγιναν στην τελευταία έκδοση του επιλέγουμε τον όνομα τομέα του δικτύου μας, το όνομα του τομέα του εξυπηρετητή και τέλος δίνουμε την ηλεκτρονική διεύθυνση του διαχειριστή.



Επιλέγουμε τυπική εγκατάσταση και τον προεπιλεγμένο φάκελο εγκατάστασης.

Σε λειτουργικό σύστημα UNIX

Η εγκατάσταση σε συστήματα Unix και συγκεκριμένα Linux είναι αρκετά πιο εύκολη αν η διανομή διαθέτει έναν διαχειριστή πακέτων. Το περιβάλλον unix θεωρείται πιο σταθερό για να δημιουργηθεί ένας εξυπηρετητής έτσι οι περισσότεροι εξυπηρετητές χρησιμοποιούν σαν λογισμικό κάποια διανομή Linux με πιο διαδεδομένες τις εκδόσεις Debian, Fedora, CentOS.

- Με την χρήση του apt (Διανομές βασισμένες σε debian)

  Χρησιμοποιούμε τις παρακάτω εντολές με δικαιώματα διαχειριστή (super user).

```
sudo apt-get update
sudo apt-get install apache2
```

- Με την χρήση yum (CentOS, Fedora)

Χρησιμοποιούμε την εντολή.

```
sudo yum install httpd
```

Τέλος για να γίνει η εκκίνηση της διεργασίας του εξυπηρετητή τρέχουμε τις εντολές:

```
sudo service apache 2 start
```

Σε λειτουργικά βασισμένα σε Debian.

```
systemctl start httpd
```

Στα λειτουργικά Fedora, CentOS.

2.2 Βάση δεδομένων

Τα πληροφοριακά συστήματα πάντα απαιτούν την αποθήκευση δεδομένων για μετέπειτα επεξεργασία τους. Παλαιότερα πολλές εφαρμογές αποθήκευαν την πληροφορία σε αρχεία, λίστες ή καταλόγους, καθώς όμως η τεχνολογία εξελίσσονταν έπρεπε να βρεθεί ένας πιο ολοκληρωμένος τρόπος αποθήκευσης δεδομένων έτσι δημιουργήθηκαν τα συστήματα βάσεων δεδομένων τα οποία οργανώνουν συλλογές δεδομένων και κάνουν εύκολη την διαχείριση τους.

2.2.1 Τύποι βάσεων

Οι τύποι βάσεων που είναι καθιερωμένοι χωρίζονται σε δύο κατηγορίες στις σχεσιακά δομημένες (SQL) και μη σχεσιακά δομημένες (NoSQL) βάσεις. Παραδοσιακά χρησιμοποιούνται σχεσιακά δομημένες βάσεις για την αποθήκευση και επεξεργασία δεδομένων σε συνδυασμό με την γλώσσα SQL (Structured Query Language), που επιτρέπει να κάνουμε ερωτήσεις και να εξάγουμε δεδομένα. Οι βάσεις είναι πολύτιμο εργαλείο σε όλα τα πληροφοριακά συστήματα. Σε συγκεκριμένες περιπτώσεις που χρειάζεται τα δεδομένα να είναι αποκανονικοποιημένα, είτε γιατί δεν ξέρουμε ακόμη τον τύπο και τον όγκο των δεδομένων που πρέπει να αποθηκευτούν είτε για λόγους απόδοσης, μπορούν αποθηκευτούν σε μη σχεσιακές βάσεις δεδομένων ή σε αποκανονικοποιημένα πεδία σε σχεσιακές βάσεις (XML).

Επικρατέστερες σχεσιακές βάσεις είναι οι MySQL, PostgreSQL, Microsoft SQL Server και Oracle Database. Αντίστοιχα μη σχεσιακές βάσεις είναι η MongoDB, Redis , Hadoop. Αξιοσημείωτες είναι η Elastic Search και το Apache Solr, αν και θεωρούνται μηχανές αναζήτησης η εγγενής λειτουργία που διαθέτουν ώστε να αποθηκεύουν δεδομένα σαν έγγραφα τις κάνει άκρως αποδοτικές πλατφόρμες.

2.2.2 MySQL

Για να δημιουργηθεί η πλατφόρμα έγινε χρήση του συστήματος βάσεων MySQL μία ανοιχτής και δημοφιλής βάσης δεδομένων η οποία απαριθμεί μία πολύ μεγάλη ανοιχτή κοινότητα πού έχει αναπτύξει διάφορα χρήσιμα εργαλεία, τα οποία προσθέτουν επιπλέον λειτουργίες στο ήδη πλούσιο σύστημα που προσφέρεται. Διαθέτει εργαλεία που παρέχουν ένα εύχρηστο περιβάλλον χρήστη και εξοικονομούν πολύτιμο χρόνο στους κατασκευαστές λογισμικών. Στο επόμενο κεφάλαιο θα αναλυθεί εκτενώς ο σχεδιασμός της βάσης και τα πεδία κάθε πίνακα.

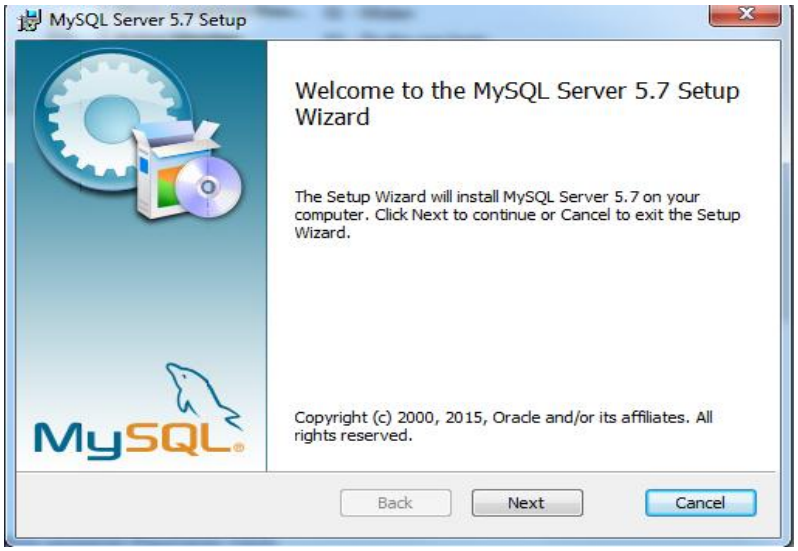
Εγκατάσταση MySQL & MySQL Workbench

Σε λειτουργικό Windows

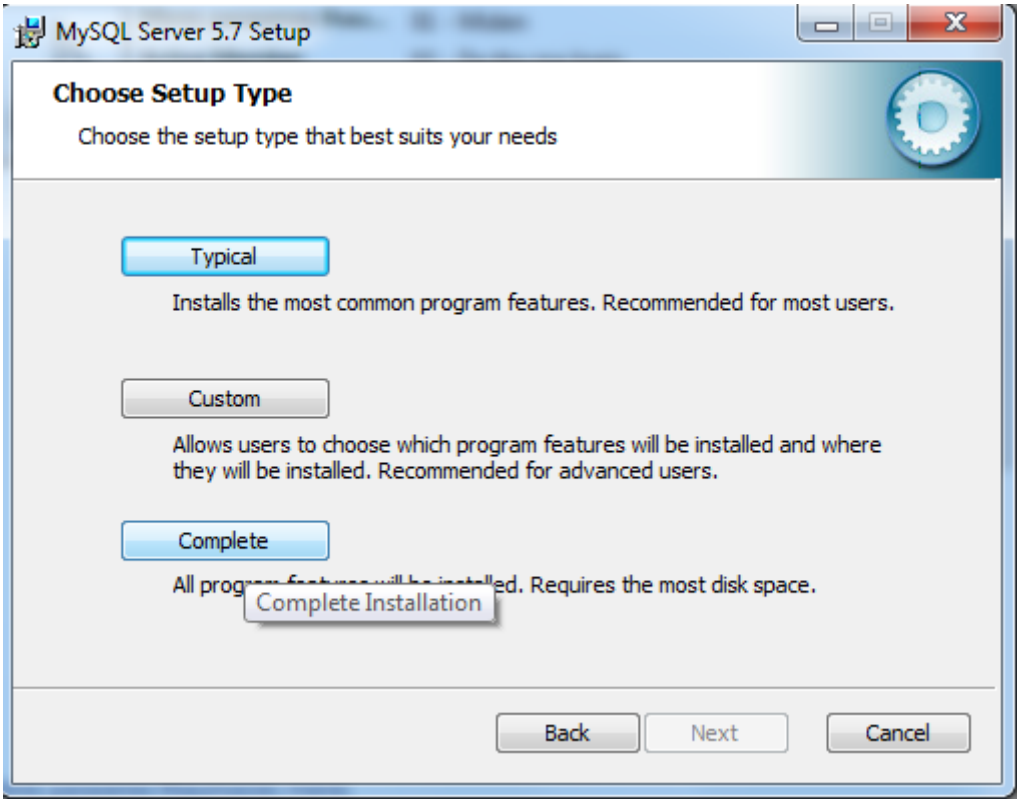
Για να κάνουμε εγκατάσταση της MySQL σε λειτουργικό Windows πρέπει να κατεβάσουμε το αντίστοιχο .msi αρχείο από τον παρακάτω σύνδεσμο. Για να κατεβάσουμε τον installer χρειαζόμαστε πρώτα έναν λογαριασμό oracle.

```
http://dev.mysql.com/downloads/installer/
```

Αφού κατεβάσουμε το αρχείο το ανοίγουμε για να ξεκινήσει η εγκατάσταση.



Επιλέγουμε «επόμενο» και στο επόμενο παράθυρο «Πλήρη εγκατάσταση».

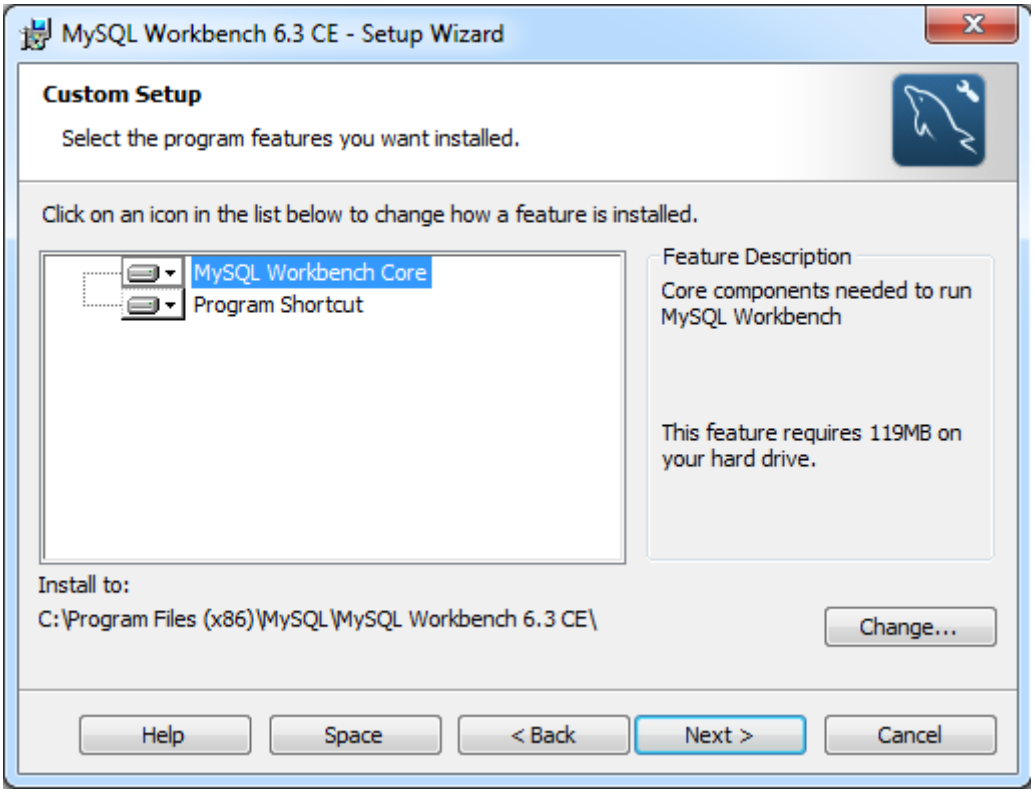


Τέλος δίνουμε ένα όνομα διαχειριστή και έναν κωδικό για την βάση.

Ένα επιπλέον βήμα που γίνεται για λόγους ευχρηστίας της βάσης και είναι προαιρετικό είναι να εγκαταστήσουμε το σύστημα διαχείρισης βάσεων MySQL που ονομάζεται MySQL Workbench. Το σύστημα αυτό είναι ένα βοηθητικό πρόγραμμα που μας επιτρέπει να εκτελούμε διάφορες διεργασίες στην βάση με μεγάλη ευκολία και με τον ελάχιστο δυνατό κόπο. Αφού το κατεβάσουμε από την παρακάτω διεύθυνση.

```
http://downloads.mysql.com/archives/workbench/
```

Εκτελούμε το .msi για να ξεκινήσει η εγκατάσταση.



Επιλέγουμε την πλήρη εγκατάσταση, το λογισμικό θα εντοπίσει αυτόματα την εγκατάσταση του MySQL Server.

Σε λειτουργικό UNIX

Η εγκατάσταση σε Linux έχει παρόμοια διαδικασία. Αφού ανοίξουμε ένα τερματικό στην διανομή μας.

- Με την χρήση του apt (Διανομές βασισμένες σε debian)

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
```

Μετά πρέπει να ενεργοποιήσουμε την MySQL με την παρακάτω εντολή.

```
sudo mysql_install_db
```

Επιπλέον πρέπει να δώσουμε τον κωδικό και το όνομα χρήστη του διαχειριστή καθώς και να παραμετροποιήσουμε την βάση σε όποια σημεία θεωρούμε απαραίτητο. Για να μας δοθούν αυτές οι επιλογές τρέχουμε την παρακάτω εντολή.

```
sudo /usr/bin/mysql_secure_installation
```

- Με την χρήση yum (CentOS, Fedora)

Παρόμοια με λειτουργικά βασισμένα σε Debian με την διαφορά ότι χρησιμοποιούμε τον διαχειριστή πακέτων yum.

```
sudo yum install mysql-server
sudo /sbin/service mysqld start
sudo /usr/bin/mysql_secure_installation
```

2.3 Γλώσσα υλοποίησης

Η γλώσσα που χρησιμοποιήθηκε για την υλοποίηση της πλατφόρμας είναι η PHP. Η PHP δημιουργήθηκε από έναν φοιτητή, τον Rasmus Lerdorf, το 1994 και είχε εντελώς διαφορετική μορφή με αυτή που έχει σήμερα. Αρχικά ο Rasmus δημιούργησε ένα script ώστε να κρατάει στατιστικά για την προσωπική του ιστοσελίδα, ο κώδικας του όμως έγινε αρκετά γρήγορα δημοφιλής και έτσι ξεκίνησε να μετασχηματίζει μία νέα κανονική γλώσσα βασισμένη στην C. Η PHP είναι πολύ δημοφιλής γλώσσα για υλοποίηση δυναμικών ιστοσελίδων και την χρησιμοποιούν μερικές από τις πιο δημοφιλείς πλατφόρμες όπως, Wordpress, Joomla, OpenCart, και Facebook.

Αν και θεωρείται ότι η PHP είναι αργή γλώσσα, σε σχέση με άλλες γλώσσες που υποστηρίζουν διεργασίες διαδικτύου (C#, Java), λόγο της μεταγλώττισης κατά την εκτέλεση του πηγαίου κώδικα, του dynamic typing και την έλλειψη ενός JIT (Just In Time) μεταγλωττιστή. Αυτό τείνει να αλλάξει καθώς, έχει δημιουργηθεί το HHVM το οποίο είναι λογισμικό το οποίο μεταγλωττίζει την PHP σε ένα εικονικό μηχάνημα με την χρήση ενός JIT compiler, αφού μετατρέψει τον κώδικα σε byte code το μεταγλωττίζει σε γλώσσα μηχανής. Είναι ένα σημαντικό βήμα ως προς την απόδοση και ίσως ανατρέψει τα δεδομένα στην αγορά εργασίας.

Εγκατάσταση PHP

Σε λειτουργικό Windows

Για να πραγματοποιήσουμε την εγκατάσταση της PHP σε λειτουργικό Windows δεν πρέπει να κατεβάσουμε το αρχείο με ονομασία «VC11 x86 Thread Safe» από τον παρακάτω σύνδεσμο.

```
http://windows.php.net/download/
```

Αφού κατεβάσουμε το αρχείο το αποσυμπιέζουμε κάπου στο σκληρό μας δίσκο, συνήθως στο C:\php\ . Μετά πρέπει να παραμετροποιήσουμε το αρχείο php.ini-recommended και να το μετονομάσουμε σε php.ini.

Τέλος πρέπει να παραμετροποιήσουμε τον αρχείο «httpd.conf» του Apache HTTP server και να προσθέσουμε στην αρχή του αρχείου τις παρακάτω ρυθμίσεις:

```
LoadModule php5_module "C:/php/php5apache2_2.dll"
AddType application/x-httpd-php .php
PHPIniDir "C:/php"
```

Οι ρυθμίσεις LoadModule και PHPIniDir παραμετροποιούνται ανάλογα με την τοποθεσία που έχουμε αποσυμπίεση τα αρχεία της PHP.

Σε λειτουργικό Linux

- Με την χρήση του apt

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

- Με την χρήση του yum

```
sudo yum install php php-mysql
```

3) Κεφάλαιο 3

3.1 Υλοποίηση της Βάσης δεδομένων

Όπως αναφέρθηκε και προηγούμενος η βάση είναι ένα από τα βασικότερα στοιχεία σχεδόν όλων τον εφαρμογών. Στο παρακάτω κεφάλαιο θα αναλύσουμε τον σχεδιασμό της βάσης, τις ρυθμίσεις που χρησιμοποίησα, τα πεδία και τις συσχετίσεις μεταξύ των πινάκων. Η βάση που θα χρησιμοποιήσουμε είναι η MySQL, οι βάση διαθέτη σύνθεση **utf8_unicode_ci** και η μηχανή – τύπος της βάσης είναι **InnoDB.**

3.1.1 Πίνακες και πεδία της βάσης

Χρησιμοποιώντας το MySQL workbench και την εντολή,

```
SHOW TABLES;
```

η οποία επιστρέφει όλους του πίνακες στην επιλεγμένη βάση παίρνουμε το παρακάτω αποτέλεσμα.

```
+---------------------+
|       Πίνακες       |
+---------------------+
| address             |
| address_user        |
| assigned_roles      |
| categories          |
| currency            |
| migrations          |
| orders              |
| permission_role     |
| permissions         |
| picture_product     |
| pictures            |
| product_wishlist    |
| products            |
| products_categories |
| review_product      |
| review_user         |
| reviews             |
| roles               |
| settings            |
| tagging_tagged      |
| tagging_tags        |
```

```
| user_order          |
| user_wishlist       |
| users               |
| wishlist            |
```

Αντίστοιχα για να πάρουμε τον κάθε πίνακα ξεχωριστά καθώς και τα ευρετήρια που μπορεί να έχει χρησιμοποιούμε τις παρακάτω εντολές στο MySQL cli περιβάλλον.

Για να πάρουμε την περιγραφή του πίνακα και να αποθηκεύσουμε το αποτέλεσμα του ερωτήματος σε ένα αρχείο κειμένου χρησιμοποιούμε την παρακάτω εντολή:

```
mysql --default-character-set=utf8 --user=<χρήστης_βάσης>
--password=<κωδικός_χρήστη> --column-names=TRUE laravel_eshop_db -e
"SELECT COLUMN_KEY as PK,COLUMN_NAME as Name,DATA_TYPE as Type from
INFORMATION_SCHEMA.COLUMNS where TABLE_NAME =’<όνομα_πίνακα>’;" >
C:/db_dump/output.txt
```

Οι πίνακες εμφανίζονται με αλφαβητική σειρά

Πίνακας address

PK	Name	Type
PRI	id	int
	address1	varchar
	address2	varchar
	city	varchar
	state	varchar
	zipcode	varchar
	created_at	timestamp
	updated_at	timestamp
	description	varchar
	country_code	varchar

Πίνακας address_user

PK	Name	Type
MUL	address_id	int
MUL	user_id	int

Πίνακας assigned_roles

PK	Name	Type
MUL	user_id	int
MUL	role_id	int

Πίνακας categories

PK	Name	Type
PRI	id	int
	name	varchar
	created_at	timestamp
	updated_at	timestamp
	parent_id	int
	depth	int
	lft	int
	rgt	int
	slug	varchar

Πίνακας currency

PK	Name	Type
PRI	id	int
	title	varchar
	symbol_left	varchar
	symbol_right	varchar
	code	varchar
	decimal_place	int
	value	double
	decimal_point	varchar
	thousand_point	varchar
	status	int
	created_at	timestamp
	updated_at	timestamp

Πίνακας Orders

PK	Name	Type
PRI	id	int
	order_id	varchar
	items	text
	created_at	timestamp
	updated_at	timestamp

Πίνακας permissions

PK	Name	Type
PRI	id	int
UNI	name	varchar
	display_name	varchar
	created_at	timestamp
	updated_at	timestamp

Πίνακας picture_product

PK	Name	Type
MUL	product_id	int
MUL	picture_id	int

Πίνακας product_wishlist

PK	Name	Type
PRI	id	int
MUL	product_id	int
UNI	wishlist_id	int
	created_at	timestamp
	updated_at	timestamp

Πίνακας products

PK	Name	Type
PRI	id	int
	title	varchar
	description	text
	price	decimal
	availability	tinyint
	featured	tinyint
	rating_cache	float
	rating_count	int
	visits	int
	created_at	timestamp
	updated_at	timestamp
	slug	varchar

Πίνακας products_categories

PK	Name	Type
PRI	product_id	int
PRI	category_id	int

Πίνακας review_product

PK	Name	Type
PRI	review_id	int
PRI	product_id	int

Πίνακας review_user

PK	Name	Type
PRI	review_id	int
PRI	user_id	int

Πίνακας reviews

PK	Name	Type
PRI	id	int
	rating	int
	comment	text
	status	tinyint
	created_at	datetime
	updated_at	datetime

Πίνακας roles

PK	Name	Type
PRI	id	int
UNI	name	varchar
	created_at	timestamp
	updated_at	timestamp

Πίνακας settings

PK	Name	Type
	general_settings	text

Πίνακας tagging_tagged

PK	Name	Type
PRI	id	int
MUL	taggable_id	varchar
MUL	taggable_type	varchar
	tag_name	varchar
MUL	tag_slug	varchar

Πίνακας tagging_tags

PK	Name	Type
PRI	id	int
MUL	slug	varchar
	name	varchar
	suggesttiny	int

```
        count       int
```

Πίνακας user_order

```
PK      Name        Type
        user_id     int
        order_id    int
```

Πίνακας user_wishlist

```
PK      Name        Type
PRI     id          int
MUL     user_id             int
MUL     wishlist_id int
        created_at  timestamp
        updated_at  timestamp
```

Πίνακας users

```
PK      Name                Type
PRI     id                  int
        firstname           varchar
        lastname            varchar
        email               varchar
        password            varchar
        phone               varchar
        created_at          timestamp
        updated_at          timestamp
        remember_token              varchar
```

Πίνακας wishlist

PK	Name	Type
PRI	id	int
	created_at	timestamp
	updated_at	timestamp

3.1.2 Ευρετήρια πίνακα

Αντίστοιχα για να επιστρέψουν όλα τα ευρετήρια που υπάρχουν στο σχήμα της βάσης χρησιμοποιούμε το MySQL cli την παρακάτω εντολή:

```
mysql --default-character-set=utf8 --user=<όνομα_χρήστη>
--password=<κωδικός χρήστη> --column-names=TRUE laravel_eshop_db -e
"SELECT DISTINCT TABLE_NAME,INDEX_NAME,INDEX_TYPE FROM
INFORMATION_SCHEMA.STATISTICS WHERE TABLE_SCHEMA = 'laravel_eshop_db';" >
C:/output.txt
```

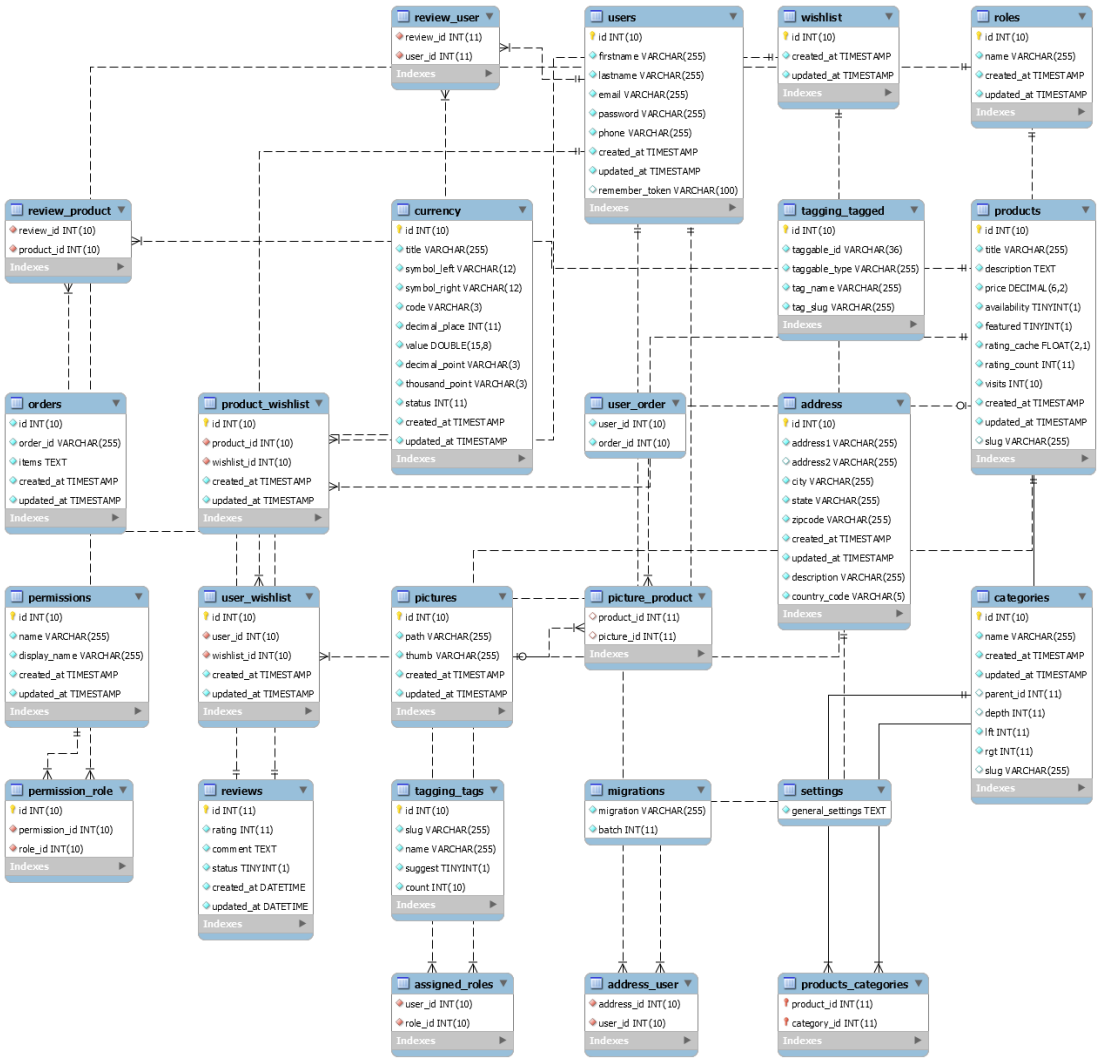
Η οποία επιστρέφει το παρακάτω αποτέλεσμα.

```
TABLE_NAME          INDEX_NAME
address             PRIMARY
address_user        index_address_users
address_user        address_users_usersid_index
assigned_roles      assigned_roles_user_id_foreign
assigned_roles      assigned_roles_role_id_foreign
categories          PRIMARY
currency            PRIMARY
orders              order_id_index
permission_role     PRIMARY
permission_role     permission_role_permission_id_foreign
permission_role     permission_role_role_id_foreign
permissions         PRIMARY
permissions         permissions_name_unique
picture_product     picture_product_id
picture_product     picture_picture_id
pictures            PRIMARY
product_wishlist    PRIMARY
product_wishlist    wishlist_id
product_wishlist    product_wishlist_product_id_foreign
products            PRIMARY
products_categories PRIMARY
products_categories category_id delete cascade
review_product      reviews_product
review_product      product_id
review_user         reviews_user
```

```
review_user         user_id
reviews             PRIMARY
roles               PRIMARY
roles               roles_name_unique
tagging_tagged      PRIMARY
tagging_tagged      tagging_tagged_taggable_id_index
tagging_tagged      tagging_tagged_taggable_type_index
tagging_tagged      tagging_tagged_tag_slug_index
tagging_tags        PRIMARY
tagging_tags        tagging_tags_slug_index
user_wishlist       PRIMARY
user_wishlist       user_wishlist_user_id_foreign
user_wishlist       user_wishlist_wishlist_id_foreign
users               PRIMARY
users               id
users               id_2
wishlist            PRIMARY
wishlist            id
```

3.1.3 Διάγραμμα Σχήματος βάσης

Στο παρακάτω διάγραμμα διακρίνουμε τους πίνακες με τα πεδία τους καθώς και τις συσχετίσεις των πινάκων γραφικά.

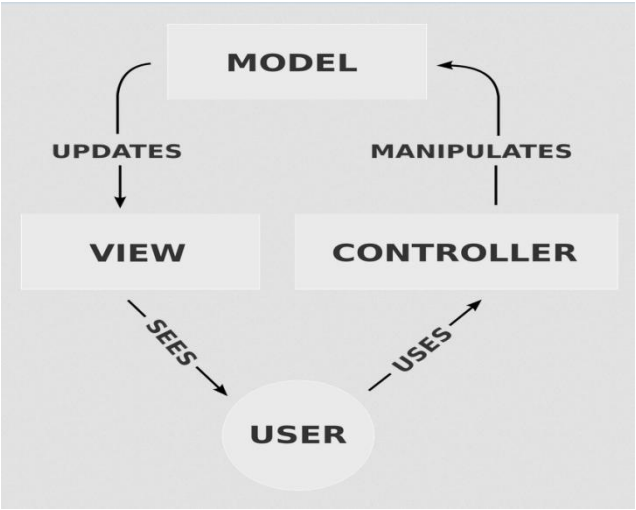


Ο σχεδιασμός αυτός θα βοηθήσει στην απεικόνιση συσχετιζόμενων δεδομένων στην εφαρμογή.

3.2 Υλοποίηση της Εφαρμογής

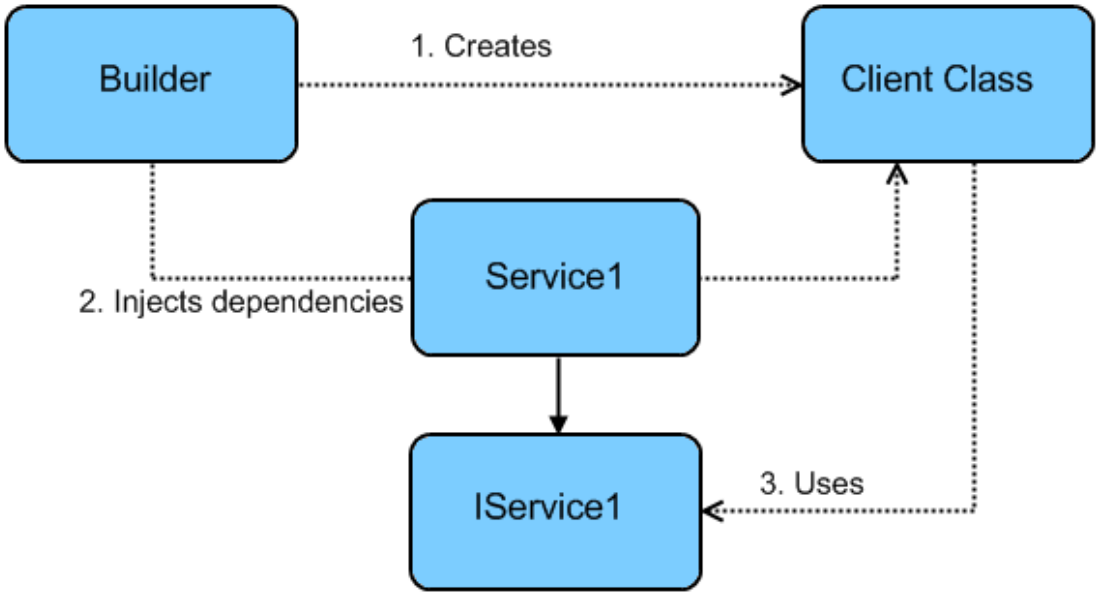
3.2.1 Αρχιτεκτονική MVC

Η αρχιτεκτονική εφαρμογή καθώς και το πλαίσιο εργασίας στηρίζονται στο πρότυπο MVC. Το πρότυπο αυτό διαχωρίζει την δομή της εφαρμογής σε τρείς βασικούς άξονες, τις οντότητες (Models – Entities), τις προβολές (Views) και τους χειριστές (Controllers). Αν και η αρχιτεκτονική αυτή υπάρχει σαν σχεδιαστική τεχνική αρκετά χρόνια έγινε ιδιαίτερα δημοφιλείς με την ανάδειξη των web εφαρμογών. Συγκεκριμένα διαχωρίζει την λογική στις οντότητες που πρέπει να διαχειριστούμε (πχ Χρήστες, Ρόλοι, Προϊόντα), στις προβολές, δηλαδή στο πώς θα εμφανίζεται η πληροφορία του συστήματος στον τελικό χρήστη και τέλος στους ρυθμιστές που είναι υπεύθυνοι για την εκτέλεση των διεργασιών της εφαρμογής και διαθέτουν την λογική του προγράμματος. Αν και σαν μονάδες δουλεύουν ξεχωριστά το τελικό αποτέλεσμα και η εμπειρία που λαμβάνει ο χρήστης είναι μία σύνθεση όλων των παραπάνω. Το MVC βοηθάει στην εγγραφή πολύπλοκων εφαρμογών σε μικρό χρονικό διάστημα και βοηθάει σημαντικά στην αποσφαλμάτωση της εφαρμογής καθώς είναι πιο εύκολο να εντοπισθούν τα σφάλματα.



3.2.2 Αρχιτεκτονική Αποθετηρίου

Η αρχιτεκτονική αποθετηρίου (Repository Pattern) είναι μια τεχνική σχεδίασης που σε συνδυασμό με την αρχιτεκτονική MVC μπορεί να αποδειχθεί ένα πανίσχυρο εργαλείο όσο η εφαρμογή μας μεγαλώνει και γίνεται πιο πολύπλοκη. Λειτουργεί σαν ένα επίπεδο αφαιρετικότητας (abstraction layer) ακριβώς πριν το επίπεδο δεδομένων (data layer) και απογυμνώνει τους ρυθμιστές από την λογική οι οποίοι έχουν πρόσβαση στην διαθέσιμη λογική χρησιμοποιώντας αντιστροφή του ελέγχου (Inversion of control), δηλαδή υλοποιώντας το εξαρτώμενο αντικείμενο στον constructor του ρυθμιστή και μετά περνάμε το νέο αντικείμενο στις αντίστοιχες μεθόδους. Η αρχιτεκτονική αποθετηρίου μπορεί να φανεί χρήσιμη για να γενικεύσουμε τις λειτουργίες του κώδικα καθώς και για να μεταβάλουμε το σύστημα της βάσης που χρησιμοποιούμε, αφού το μόνο που πρέπει να κάνουμε για να μεταφέρουμε την εφαρμογή μας είναι να υλοποιήσουμε το αντίστοιχο αποθετήριο που θα είναι συμβατό με την νέα βάση μας. Στην εφαρμογή θα χρησιμοποιηθεί ένας συνδυασμός MVC και αρχιτεκτονικής αποθετηρίου αφού το framework υποστηρίζει αντιστροφή του ελέγχου μέσω του IOC container που διαθέτει.



3.2.3 Πλαίσιο Εργασίας (Framework)

Για να δημιουργήσω την εφαρμογή χρησιμοποίησα ένα σύνολο από βοηθητικά εργαλεία τα οποία με βοήθησαν να αποφύγω βασικά προγραμματιστικά προβλήματα και να επιταχύνουν την ροή (workflow) της υλοποίησης και της αρχιτεκτονικής της εφαρμογής. Τέτοιου τύπου βοηθήματα ονομάζονται framework και υπάρχουν σχεδόν για όλες τις γνωστές γλώσσες προγραμματισμού για παράδειγμα αν κάποιος αποφασίσει να αναπτύξει μια εφαρμογή σε C# υπάρχει η δυνατότητα να το κάνει μέσω το .NET framework που έχει αναπτυχθεί από την Microsoft, αν επιλέξει σε Java υπάρχει το Spring Framework. Τα frameworks που είναι διαθέσιμα είναι πάρα πολλά για να απαριθμηθούν εδώ, τα παραδείγματα αναφέρονται σε βασικά πλαίσια εργασία που έχουν μεγάλες κοινότητες υποστήριξης.

3.2.4 Το πλαίσιο εργασίας Laravel 4.2

Το πλαίσιο εργασίας που χρησιμοποιήθηκε για την εφαρμογή ονομάζεται Laravel 4.2 . Δημιουργήθηκε από τον Taylor Otwell το 2011 ο σχεδιασμός του έχει βασιστεί στο μοντέλο αρχιτεκτονικής MVC και διαθέτει διαχείριση πακέτων μέσω του composer. Αν και θεωρείται σχετικά νέο framework έχει καταφέρει να συγκεντρώσει τον ενδιαφέρον πολλών προγραμματιστών που αναπτύσσουν εφαρμογές σε PHP και έχει καταφέρει να αναπτύξει μια πολύ αξιόλογη κοινότητα μέσα σε λίγα χρόνια. Η πολύ καλή τεκμηρίωση τον μεθόδων που διαθέτει το κάνει άμεσα ανταγωνιστικό για παλιούς και νέους προγραμματιστές.

3.2.4.1 Εγκατάσταση του Laravel 4.2

Για να πραγματοποιήσουμε την εγκατάσταση του Laravel 4.2 πρέπει πρώτα να εγκαταστήσουμε τον διαχειριστή πακέτων composer. Επίσης πρέπει να πληρούνται κάποιες προϋποθέσεις όπως :

- Να έχουμε εγκατεστημένη την PHP σε έκδοση 5.4 ή νεότερη.
- Να έχουμε εγκατεστημένη και ενεργοποιημένη την επέκταση MCrypt.
- Να έχουμε εγκατεστημένη την επέκταση php-json.

Σε λειτουργικό Windows

```
https://getcomposer.org/Composer-Setup.exe
```

Κατεβάζουμε τον installer και κάνουμε την εγκατάσταση.

Σε λειτουργικό Linux

Κατεβάζουμε το αρχείο «composer.phar» από την σελίδα του composer και υπάρχει η δυνατότητα να το κρατήσουμε τοπικά στον φάκελο της εφαρμογής ή να το μεταφέρουμε σε global περιβάλλον και να το χρησιμοποιούμε κατευθείαν από το τερματικό από όλες τις τοποθεσίες, απλά μεταφέροντας το αρχείο στην παρακάτω τοποθεσία. `usr/local/bin` .

Αφού εγκαταστήσουμε τον composer χρησιμοποιούμε την εντολή `create-project` για να εγκαταστήσουμε το πλαίσιο εργασίας. Συγκεκριμένα πρέπει να δηλώσουμε το αποθετήριο ,το πακέτο και την έκδοση που θέλουμε να κατεβάσουμε.

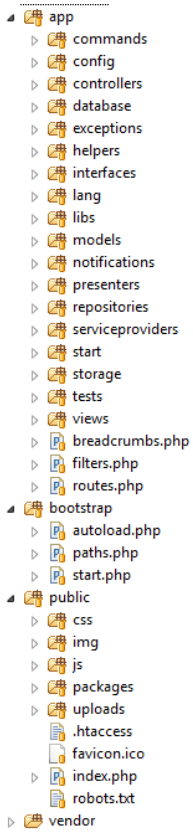
```
composer create-project laravel/laravel {directory} 4.2 --prefer-dist
```

3.2.4.2 Παραμετροποίηση του πλαισίου εργασίας

Αφού εγκαταστήσουμε το Laravel τρέχουμε την εντολή composer update για να κατεβάσουμε τα απαραίτητα πακέτα. Αφού ολοκληρωθεί η διαδικασία το πρώτο πράγμα που πρέπει να κάνουμε είναι να ορίσουμε ένα τυχαίο κλειδί 32 χαρακτήρων στο αρχείο `app/config/app.php` το οποίο θα χρησιμοποιηθεί από την επέκταση Mcrypt και η λειτουργία του θα αναλυθεί στο κεφάλαιο της ασφάλειας της εφαρμογής. Επίσης ορίζουμε την ζώνη ώρας και την προεπιλεγμένη γλώσσα που θα χρησιμοποιεί το framework . Τέλος δίνουμε στον φάκελο app/storage δικαιώματα read – write καθώς εκεί αποθηκεύονται τα logs της εφαρμογής και οι αποθηκευμένες (cached) προβολές (views).

3.2.4.3 Δομή του πλαισίου εργασίας

Εδώ θα αναλύσουμε την δομή των αρχείων του framework για να γίνει κατανοητό πώς ακριβώς δουλεύει η εφαρμογή. Παρακάτω βλέπουμε ένα screenshot της δομής.


app
commands
config
controllers
database
exceptions
helpers
interfaces
lang
libs
models
notifications
presenters
repositories
serviceproviders
start
storage
tests
views
breadcrumbs.php
filters.php
routes.php
bootstrap
autoload.php
paths.php
start.php
public
css
img
js
packages
uploads
.htaccess
favicon.ico
index.php
robots.txt
vendor

Οι βασικοί φάκελοι που διακρίνουμε μόλις ανοίξουμε τον κεντρικό φάκελο είναι οι φάκελοι app, bootstrap, public, vendor.

Φάκελος app

Ο κύριος φάκελος της εφαρμογής, σε αυτόν αποθηκεύονται οι ρυθμίσεις τις εφαρμογής (φάκελος config), οι ρυθμιστές (controllers), οι διεργασίες και οι συναλλαγές που γίνονται με την βάση (migrations – seeds), η τοπικοποίηση (localization – φάκελος lang), οι οντότητες που θα χρησιμοποιήσουμε (models), η αποθήκευση της μνήμης cache, τα logs της εφαρμογής, οι συνεδρίες και οι αποθηκευμένες προβολές. Καθώς και οι προβολές που θα εμφανίσουμε στον τελικό χρήστη. Επίσης πολλή σημαντικά αρχεία είναι τα filters.php και routes.php. Το πρώτο λειτουργεί σαν middleware ανάμεσα στην εφαρμογή και το session, δηλαδή χρησιμοποιείται στον έλεγχο αδειοδότησης, και στο δεύτερο ορίζουμε τα URIs που συνήθως δείχνουν σε κάποια υπηρεσία ή μπορεί να περιέχουν και ένα closure.

Φάκελος bootstrap

Σε αυτόν τον φάκελο βρίσκονται τα scripts που είναι υπεύθυνα για την έναρξη της εφαρμογής, τον προσδιορισμό των απαιτούμενων κλάσεων και βιβλιοθηκών και την φόρτωση τους στον πάροχο υπηρεσιών (service provider) του framework.

Φάκελος public

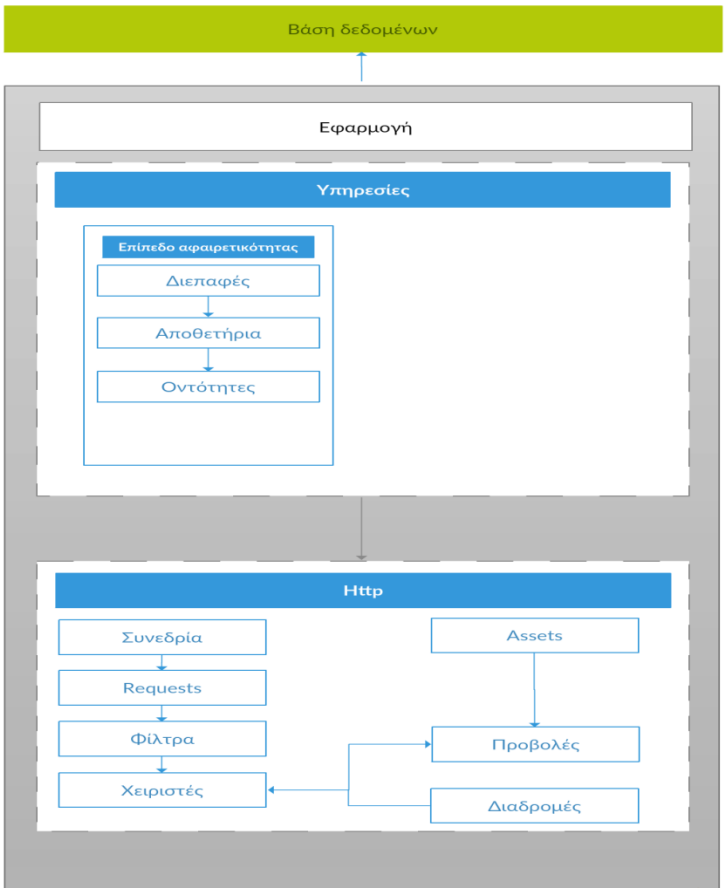
Στον φάκελο public αποθηκεύουμε όλα τα αρχεία που θα είναι προσβάσιμα από τον τελικό χρήστη καθώς και τα βοηθητικά εργαλεία για την οικοδόμηση της οπτικής διεπαφής της εφαρμογής (user interface) και την εμπειρία που θα λαμβάνουν οι χρήστες (UX) .

Φάκελος vendor

Ο φάκελος vendor απαρτίζεται από διάφορα πακέτα/βιβλιοθήκες κώδικα που συνθέτουν την ίδια την εφαρμογή και μέρη αυτής.

3.2.5 Αρχιτεκτονική της εφαρμογής

Η αρχιτεκτονική της εφαρμογής έχει πρωτεύοντα ρόλο όσο αναφορά την ανάπτυξη της γι' αυτό τον λόγο χρειάζεται να κάνουμε προσεκτικό σχεδιασμό, ώστε να ευνοηθούμε σε περίπτωση επέκτασης, αλλαγής ή ενημέρωσης της εφαρμογής. Στο παρακάτω διάγραμμα εμφανίζεται η κεντρική λογική με την οποία έχει δημιουργηθεί η εφαρμογή.



Στο διάγραμμα διακρίνουμε πώς αλληλεπιδρά η εφαρμογή με την βάση δεδομένων και τα υπόλοιπα συστατικά που την συνθέτουν. Οι υπηρεσίες παρέχονται μέσα από το επίπεδο αφαιρετικότητας το οποίο αποτελείτε από τις διεπαφές (interfaces) των κλάσεων, τα αποθετήρια (repositories) που υλοποιούν τις διεπαφές και φυσικά τις οντότητες στις οποίες επιδρούν οι υπηρεσίες.

Όταν φτάνουμε στο επίπεδο του Http επιστρέφουμε τα αποτελέσματα των υπηρεσιών αφού πρώτα επιλέξουμε τις κατάλληλες διαδρομές (routes – URIs), το request που φθάνει στον εξυπηρετητή φιλτράρετε βάση ρόλων (θα αναφερθώ στο μοντέλο RBAC σε επόμενο υποκεφάλαιο). Και τέλος καταλήγει στους χειριστές που

«σερβίρουν» τις κατάλληλες προβολές οι οποίες καλλωπίζονται από παρουσιαστές (presenter pattern).

3.2.6 Eloquent ORM

Όπως τα περισσότερα frameworks που επικοινωνούν με βάσεις δεδομένων έτσι και το Laravel διαθέτει το δικό του ORM που ονομάζεται Eloquent (ευφράδεια). Με την βοήθεια του ORM (object relation mapping), μπορούμε να διαχειριστούμε συσχετιζόμενες εγγραφές στην βάση σαν να ήταν γνωρίσματα του συσχετιζόμενου αντικείμενου. Για παράδειγμα ένας χρήστης μπορεί να έχει πολλούς ρόλους. Μία τέτοια συσχέτιση στην βάση προϋποθέτει έναν συνδετικό πίνακα που θα δείχνει την συσχέτιση ανάμεσα στις οντότητες των χρηστών και των ρόλων . Για να επιστραφούν οι ρόλοι και ο χρήστης από την βάση προϋποθέτει την δημιουργία ενός διπλού join statement. Με την χρήση του Eloquent μπορούμε να επιστρέψουμε την ανάλογη συσχέτιση με πολύ πιο απλό τρόπο αρκεί να έχουμε ορίσει τις αντίστοιχες συσχετίσεις στις οντότητες μας. Το διπλό join λοιπόν αντικαθιστάτε από μία γραμμή κώδικα. Αυτό απλοποιεί ιδιαιτέρως την ανάπτυξη της εφαρμογής με ένα μικρό τίμημα στην απόδοση.

```
User::with('roles')->get();
```

3.2.7 Η υλοποιημένη εφαρμογή

Με βοηθό τα εργαλεία που αναφέρθηκαν και άλλες βοηθητικές βιβλιοθήκες υλοποιήθηκε η τελική εφαρμογή της πτυχιακής εργασίας. Σε αυτό το υποκεφάλαιο θα αναλυθεί η δομή και οι μέθοδοι που υλοποίησα για την δημιουργία της.

3.2.7.1 Οντότητες (Models)

Οι οντότητες είναι βασικό το κύριο κομμάτι της υλοποίησης καθώς δημιουργούν την κατάλληλη σύνδεση με την βάση. Στη εφαρμογή έχουμε με αλφαβητική σειρά τις παρακάτω.

- Address
- Category
- Order
- Permission
- Picture
- Product
- Review

- Role
- User
- Wishlist

Για κάθε οντότητα υπάρχει το αντίστοιχο αρχείο στον φάκελο models που περιέχει την υλοποίηση της αντίστοιχης κλάσης. Οι κλάσεις κληρονομούν από την κλάση Eloquent η οποία είναι κλάση του framework και προσφέρει κάποιες βοηθητικές λειτουργίες στην κλάση, παρακάμπτοντας τις αρχικές τιμές της ή καθορίζοντας της, επίσης μπορούμε να ορίσουμε τις δικές μας μεθόδους και στατικές μεταβλητές. Στα models μπορούμε να αποθηκεύσουμε πληροφορία σχετικά με τον συσχετιζόμενο πίνακα της κλάσης, Ποια πεδία στην βάση είναι εγγράψιμα, και τις συσχετίσεις με τις υπόλοιπες κλάσεις που σαν προέκταση του Eloquent ORM μπορούμε να τις ανακτήσουμε με τον τρόπο που περιγράφηκε πιο πάνω. Για παράδειγμα μπορούμε να δούμε την κλάση address.

```
<?php
use Laracasts\Presenter\PresentableTrait;

class Address extends Eloquent {
	use PresentableTrait;

	/**
	 * Declare the presenter for address Model
	 * @var string
	 */
	protected $presenter = 'app\presenters\AddressPresenter';
	/**
	 * The database table used by the model.
	 *
	 * @var string
	 */
	protected $table = 'address';



	/**
	 * Fillable values for user Address
	 * @var array
	 */
```

```
      Protected    $fillable=
array('address1','address2','city','state','zipcode','description','country
_code');



            /**
             * Address realtion with user
             *
             */
            public function user()
            {
                  return $this->belongsTo('User');
            }


            public static function searchOptions(){
                  $array =
array("created_at"=>"Latest","firstname"=>"Firstname","lastname"=>"Lastname
","email"=>"Email","phone"=>"Phone");
                  return $array;

            }

            public static function searchOrder(){

      $array = array('asc'=>'Ascending','desc'=>'Descending');
                  return $array;
            }

      }
```

3.2.7.2 Ελεγκτές

Οι ελεγκτές παίζουν πρωτεύοντα ρόλο στην εφαρμογή και είναι κομμάτι μεγάλης σημασίας. Κάθε διαδρομή (route) που προσφέρει μία υπηρεσία επικοινωνεί με έναν ελεγκτή ο οποίος επιστρέφει ένα αποτέλεσμα διεργασιών του data layer ή μπορεί να αναφέρεται σε μια υπηρεσία που υλοποιεί ή στην συγκεκριμένη περίπτωση που

υλοποίει το αποθετήριο μας. Μέσα στους ελεγκτές γίνεται και η διαδικασία του server side validation που θα αναλυθεί στο κομμάτι της ασφάλειας.

3.2.7.3 Αποθετήρια

Για να υλοποιήσουμε το επίπεδο αφαιρετικότητας πρέπει να υλοποιήσουμε τις διεπαφές για κάθε κλάση καθώς και την υλοποίηση τους, επίσης θα πρέπει να δημιουργήσουμε έναν πάροχο υπηρεσιών (service provider) το οποίο θα «σερβίρουμε» στο δοχείο υπηρεσιών (service container) για να γίνουν αντιληπτές από τον πυρήνα της εφαρμογής. Για να κάνουμε κάτι τέτοιο αρκεί να δώσουμε το namespace του provider που δημιουργήσαμε στον κλειδί του πίνακα, που επιστρέφει με την ονομασία providers που βρίσκεται στο αρχείο app/config/app.php Αυτό μας παρέχει μεγάλη ευελιξία, καλύτερη δομή στον κώδικα μας και μας επιτρέπει να αποφεύγουμε τα λογικά και συντακτικά λάθη.

Αρχείο app/config/app.php

```
            'providers' => array(
                ‘Provider_full_namespace’, ),
```

Διεπαφή αποθετηρίου

```
      interface AddressRepositoryInterface{

      /**
       * Returns all Addresses
       * @return [Eloquent Collection]
       */
      public function  getAllAddresses();

      /**
       * Returns an address by id
       * @param  [int] $id  [address's id]
       * @return [Eloquent instance]
       */
      public function  getAddressById($id);

      /**
```

```
        * Set address for user
        * @param Address  $address  [Eloquent instance]
        */
       public function setAddress(Address $address);

       /**
        * Set update for address
        * @param  Address $address [Eloquent instance]
        */
       public function updateAddress(Address $address);

       /**
        * Delete an address
        * @param  Address $address [Eloquent instance]
        */
       public function deleteAddress(Address $address);
       }
```

Υλοποίηση αποθετηρίου address

```
class AddressRepository implements AddressInterface{


              public function  getAllAddresses(){
                     return Address::all();
              }


              public function  getAddressById($id){
                     return Address::find($id);
              }


public function setAddress(Address $address){
              $address->description = Input::get('description_new');
              $address->address1 = Input::get('name');
              $address->address2 = Input::get('address2');
              $address->city = Input::get('locality');
              $address->state = Input::get('administrative_area_level_3');
              $address->zipcode = Input::get('postal_code');
              $address->country_code = Input::get('country_short');
```

```
                $address->save();
                }

        public function updateAddress(Address $address){
                        $address->update(array(
                        'address1' => Input::get('name'),
                        'address2' => Input::get('address2'),
                        'city' => Input::get('locality'),
                        'state' => Input::get('administrative_area_level_3'),
                                'zipcode' => Input::get('postal_code'),
                                'description' => Input::get('description'),
                                'country_code' => Input::get('country_short'),
                                ));
                }
                public function deleteAddress(Address $address){
                        $address->delete();
                }
```

Πάροχος υπηρεσιών

```
class AddressServiceProvider extends ServiceProvider{

public function register(){
$this->app->bind('app\interfaces\AddressRepositoryInterface',
'app\repositories\AddressRepository');
        }
}
```

3.2.7.4 Παρουσιαστές (Presenters)

Οι παρουσιαστές (presenters) είναι κλάσεις που χρησιμοποιούνται για να καλλωπίζουν τις προβολές που λαμβάνει ο τελικός χρήστης, η κάθε οντότητα έχει και από έναν αντίστοιχο presenter, λειτουργούν σαν mutators/getters οι οποίες δέχονται ένα αντικείμενο της αντίστοιχης κλάσης. Μπορούν να μεταβάλουν ή να συνδυάσουν δεδομένα και γνωρίσματα του αντικειμένου τα οποία ανήκουν στο αντικείμενο και επιστρέψουν την μεταβλημένη πληροφορία στις προβολές.

Στην εφαρμογή χρησιμοποιηθήκαν οι βιβλιοθήκες του πακέτου `laracasts/Presenter` το οποίο έχει άδεια χρήσης MIT και διευκολύνει την υλοποίηση αυτής της λειτουργίας. Με την χρήση της μεθόδου `present()` που

προσφέρει το πακέτο που χρησιμοποιείται σαν επέκταση του αντικειμένου εφόσον έχουμε προσθέσει την κλάση `Laracasts\Presenter\PresentableTrait` στο αντίστοιχο αρχείο της οντότητας. Τα Traits είναι abstract κλάσεις που δημιουργήθηκαν για να λύσουν το πρόβλημα της πολλαπλής κληρονομικότητας στην PHP.

Για παράδειγμα αν θέλουμε να πάρουμε το πλήρες όνομα ενός χρήστη αρκεί να καλέσουμε την αντίστοιχη μέθοδο από το αντίστοιχο αρχείο presenter.

```
$user->present()->full_name;
```

Παρουσιαστής Χρήστη

```
class UserPresenter extends Presenter{

        /**
         * Returns first name of user
         * @return string
         */
        public function first_name(){
                return $this->firstname;
        }

        /**
         * Returns last name of user
        * @return string
         */
        public function last_name(){
                return $this->lastname;
        }

        /**
         * Returns full name of user
         * @return  string
         */
        public function full_name(){
                return $this->firstname.' '.$this->lastname;
        }


}
```

3.2.7.5 Δρομολογητές (Routes)

Το framework προσφέρει την δυνατότητα ορισμού διαδρομών στην εφαρμογή, αυτό μας επιτρέπει να ορίσουμε τις REST υπηρεσίες μας μέσα από όμορφα δομημένα URIs που συνήθως στέλνουν ή επιστρέφουν δεδομένα επικοινωνώντας με τις μεθόδους των ελεγκτών (controllers), μπορούν όμως να δεχτούν και ένα closure. Το αρχείο routes.php είναι πολύ μεγάλο για να συμπεριληφθεί όλο, πιο κάτω αναφέρονται οι βασικές διαδρομές του πίνακα διαχείρισης για τις κατηγορίες.

```
        Route::get('admin/categories/index', array('as'
=>'admin.categories.index','uses'=>'CategoriesController@getIndex'));
        Route::post('admin/categories/create', array('as'
=>'admin.categories.create','uses'=>'CategoriesController@postCreate'));
        Route::get('admin/categories/destroy/{id}', array('as'
=>'admin.categories.destroy','uses'=>'CategoriesController@postDestroy'));
        Route::post('admin/categories/category-relation/', array('as'
=>'admin.categories.relation','uses'=>'CategoriesController@postCategoryRelation'));
        Route::get('admin/categories/category-detach/{id}', array('as'
=>'admin.categories.detach','uses'=>'CategoriesController@detachCategory'));
        Route::get('admin/categoriescategories/edit/{id}',
array('as'=>'admin.categories.edit', 'uses'=>'CategoriesController@editCategory'));
                Route::patch('admin/categories/update/{id}',
array('as'=>'admin.categories.update', 'uses'=>'CategoriesController@updateCategory'));
```

3.2.7.6 Φίλτρα (Middleware)

Το framework υποστηρίζει επίσης την χρήση φίλτρων ή middleware τα οποία αποκόπτουν τους μη εξουσιοδοτημένους χρήστες από τα να έχουν πρόσβαση σε διαχειριστικά μέρη της εφαρμογής. Δεν επιτρέπεται να αφήσουμε έναν απλό χρήστη να έχει πρόσβαση στο διαχειριστικό περιβάλλον ή έναν χρήστη να έχει πρόσβαση στις πληροφορίες ενός άλλου χρήστη. Στην ουσία λειτουργεί σαν κέλυφος προστασίας που αποκόπτει συγκεκριμένα αιτήματα με βάση συγκεκριμένα κριτήρια.

Στην εφαρμογή χρησιμοποιούμε το μοντέλο RBAC (Role Based Access Control) , δηλαδή την εξουσιοδότηση των χρηστών με βάση τους ρόλους που διαθέτουν. Σε μια πιο πολύπλοκη εφαρμογή ίσως να ήταν περισσότερο χρήσιμο να γίνεται έλεγχος βάση εξουσιοδοτήσεων (permissions). Τα φίλτρα με την χρήση του μοντέλου ελέγχουν αν ο χρήστης διαθέτει τον κατάλληλο ρόλο πρόσβασης και επιτρέπουν ή απορρίπτουν το αίτημα πρόσβασης. Το σύστημα διαθέτει τέσσερις ρόλους, Administrator, Moderator, User, Banned.

3.2.7.7 Προσόψεις (Facades)

Οι προσόψεις παρέχουν μία στατική διεπαφή για τις κλάσεις που είναι διαθέσιμες στο δοχείο των υπηρεσιών του framework (service container), και παρέχουν μία εύκολη συντακτική υλοποίηση των μεθόδων. Το Laravel έχει ενσωματωμένα κάποια facades όπως το `Route::get()` ή το `Input::get()` . Γενικά δεν θεωρείται σωστό για την αρχιτεκτονική της εφαρμογής να χρησιμοποιούνται παντού facades αλλά μπορούν να συμβάλουν στην ταχύτερη ανάπτυξη και το συντακτικό τους θεωρείται πιο κατανοητό. Χωρίς την χρήση των facades, η μέθοδος `Route::get()` θα έπρεπε να γραφτεί με αυτόν τον τρόπο. `$app['router']->get();` θα έπρεπε δηλαδή να πάρουμε το object της κλάσης router από τον service container και να «ζητήσουμε» την μέθοδο `get()` .

3.2.7.8 Τοπικοποίηση (Localization)

Το πλαίσιο εργασίας Laravel 4.2, προσφέρει ένα πολύ απλό façade και έναν helper που μας επιτρέπουν να διαχειριστούμε την τοπικοποίηση της εφαρμογής εύκολα και με μηδαμινό κόπο. Χρησιμοποιόντας τους ISO κωδικούς χωρών (π.χ en – English, el – Ελληνικα κτλπ) και τις μεθόδους `Lang::get()` και `Lang::choice()` μπορούμε ορίζοντας νέο κωδικό γλώσσας στην συνεδρία αν αλλάξουμε την γλώσσα της εφαρμογής. Δημιουργώντας τα κατάλληλα αρχεία στον φάκελο app/lang/ που απλά επιστρέφουν έναν πίνακα με κλειδιά και τιμές για πάραδειγμα.

Αρχείο app/lang/en/messages.php

```
<?php
```

```
Return array(
‘welcome’ => ‘Welcome’,
'user' => 'User|Users', //Πληθυντικός);
```

Αρχείο app/lang/el/messages.php

```
<?php
Return array(
‘welcome’ => ‘Καλώς ήρθατε’
'user' => 'Χρήστης|Χρήστες’, //Πληθυντικό );
);
```

Έτσι μέσα από στις προβολές και τους ελεγκτές που επιστρέφουν μηνύματα χρησιμοποιούμε τις μεθόδους `get()` για μηνύματα τα οποία δεν έχουν πληθυντικό αριθμό και αντίστοιχα την `choice()` ή οποία δέχεται το κλειδί και έναν αριθμό για να προσδιορίσει αν πρέπει να εμφανίσει την φράση στον ενικό ή τον πληθυντικό.

3.3 Γραφικό Περιβάλλον

3.3.1 Προβολές (Views)

Για να δημιουργήσουμε μία γραφική διεπαφή χρειάζεται να υλοποιήσουμε κάποιες προβολές οι οποίες θα οπτικοποιούν τα δεδομένα που επιστρέφουν οι REST υπηρεσίες. Για τις προβολές χρησιμοποιήθηκαν οι τεχνολογίες HTML 5, Twitter bootstrap framework, Jquery και διάφορα Jquery plugins. Επίσης το laravel διαθέτει ενσωματωμένη μία μηχανή η οποία κάνει render τις προβολές και ονομάζεται Blade. Διαθέτει ενδιαφέρον λειτουργίες και σίγουρα βοηθάει αρκετά στην ανάπτυξη του δυναμικού περιεχομένου και στην ενσωμάτωση της σχεδίασης του γραφικού περιβάλοντος.

3.3.1.1 Blade templating engine

Όλες οι προβολές που δημιουργούνται με το blade μεταγλωττίζονται σε κώδικα php και μετά αποθηκεύονται στο `app/storage/views` μέχρι να υποστούν επεξεργασία εκ νέου. Αυτό έχει ως αποτέλεσμα η μηχανή να επιστρέφει γρήγορα τις προβολές με σχεδόν μηδενική επιβάρυνση στην εφαρμογή. Οι προβολές blade χρησιμοποιούν την κατάληξη `.blade.php` και αποθηκεύονται στον φάκελο `resources/views`. Μας δίνεται η δυνατότητα να επεκτείνουμε τις προβολές και να δημιουργήσουμε μια σχέση γονέα - παιδιού, οπού ο γονέας είναι η κεντρική προβολή και το παιδί/παιδιά είναι οι προβολές που ενσωματώνονται στην κεντρική.

Προβολή γονέας:

```
<!-- Stored in app/views/layouts/master.blade.php -->

<html>
    <body>
        @section('sidebar')
            This is the master sidebar.
        @show

        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

Προβολή παιδί

```
@extends('layouts.master')

@section('sidebar')
    <p>This is appended to the master sidebar.</p>
@stop

@section('content')
    <p>This is my body content.</p>
@stop
```

Για να εμφανίζουμε δεδομένα πρέπει να τα περικλείσουμε σε διπλά άγκιστρα `{{ }}` Αυτό προσφέρει την λειτουργικότητα της εντολής `echo` στην PHP. Αν θέλουμε τα δεδομένα να κάνουν escape την html χρησιμοποιούμε τριπλά άγκιστρα αύτη η σύνταξη είναι η αντίστοιχη `htmlspecialchars()` μέθοδο. Στην Πέμπτη έκδοση του laravel αντικαταστάθηκε με μονά άγκιστρα και διπλά θαυμαστικά `{!! !!}` .

Χρησιμοποιώντας την μηχανή blade και πιο συμβατικές τεχνολογίες όπως HTML 5 και JavaScript / JQuery καθώς και τα παρακάτω jquery plugins δημιουργήθηκε η παρακάτω οπτική απεικόνιση των δεδομένων.

HTML Frameworks:

- **Twitter Bootstrap 3**

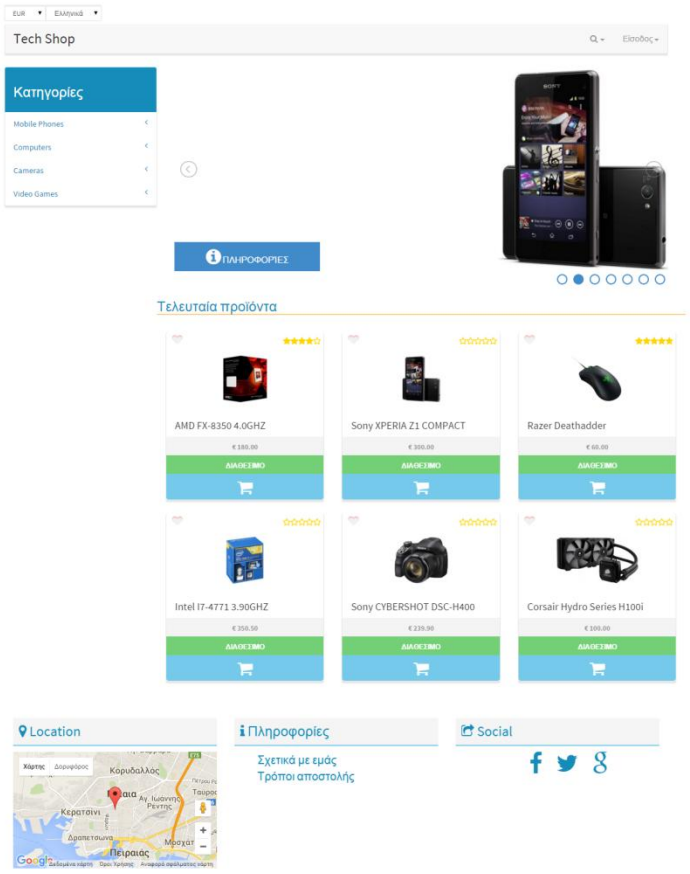
JavaScript frameworks και πρόσθετα

- **Jquery**
    - **bootstrap-number-input.js**
    - **bootstrap-rating-input.min.js**
    - **jquery.geocomplete.min.js**
    - **metisMenu.min.js**
    - **typeahead.js**
    - **bootstrap.min.js**
    - **ekko-lightbox.min.js**
    - **jquery.lightSlider.js**
    - **pace.min.js**
    - **bootstrap-toggle.min.js**
    - **chosen.jquery.min.js**
    - **fileinput.min.js**
    - **jquery.nestable.js**
    - **select2.full.js**

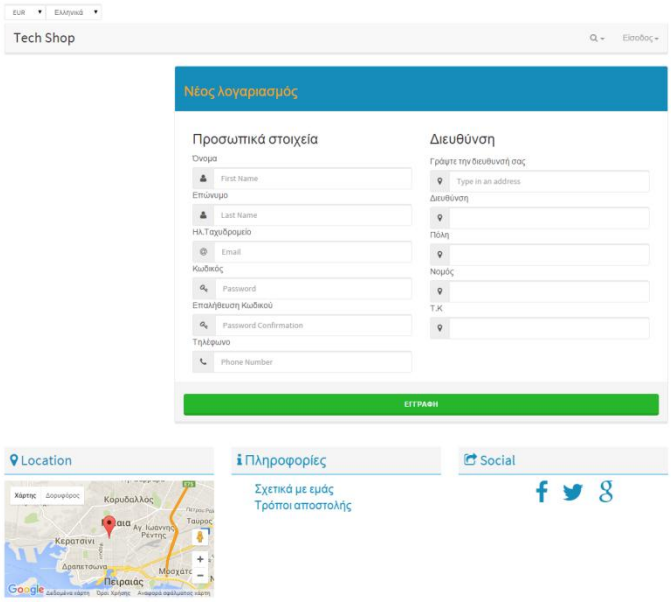
Παραθέτονται κάποιες κύριες προβολές μέσα από την τελική εφαρμογή:

Μπροστινή πρόσοψη (Frontend)

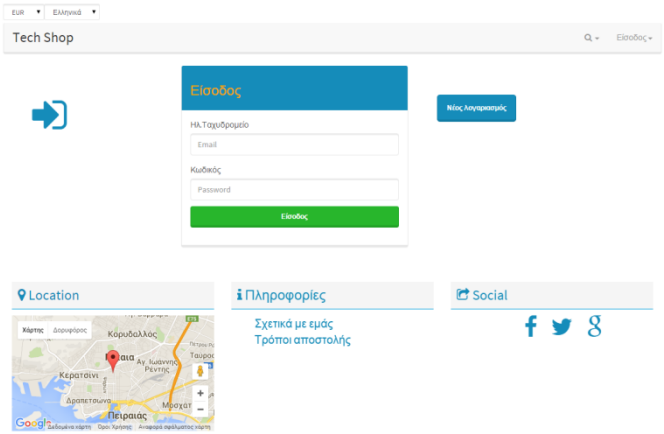
Αρχική σελίδα



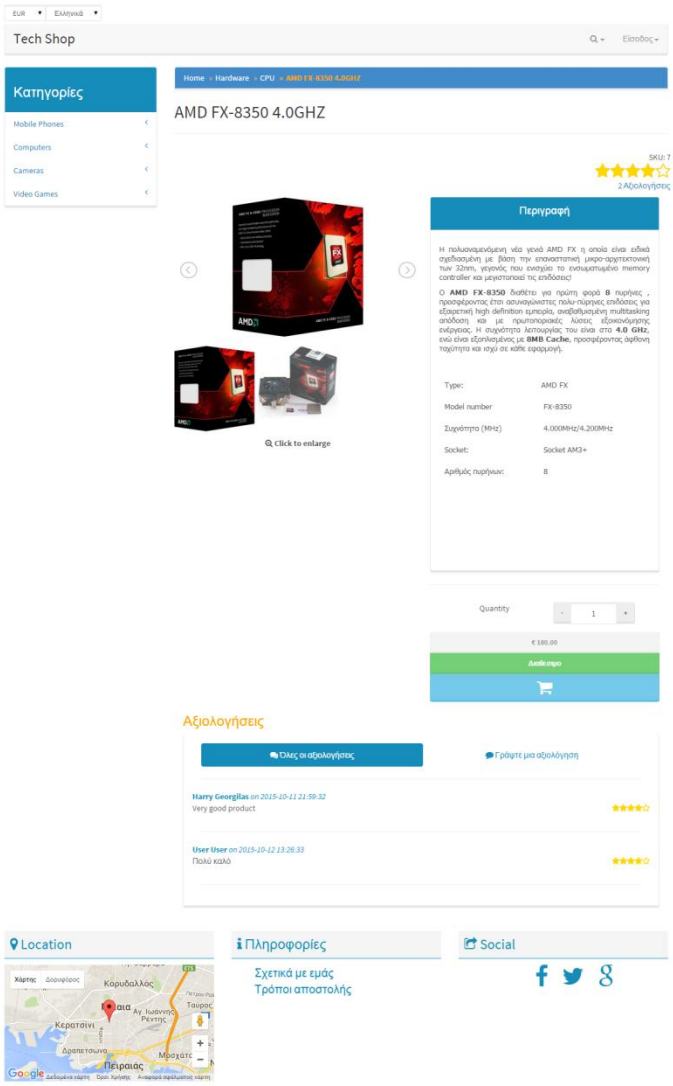
Δημιουργία νέου λογαριασμού



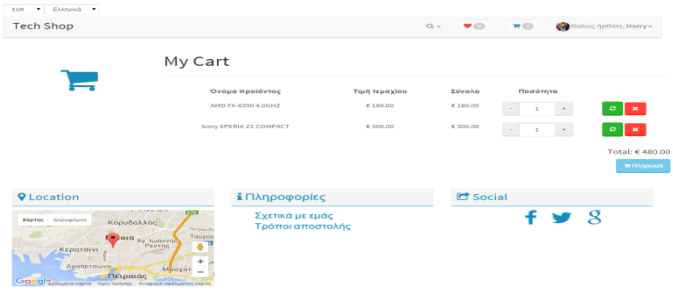
Σύνδεση με υπάρχων λογαριασμό



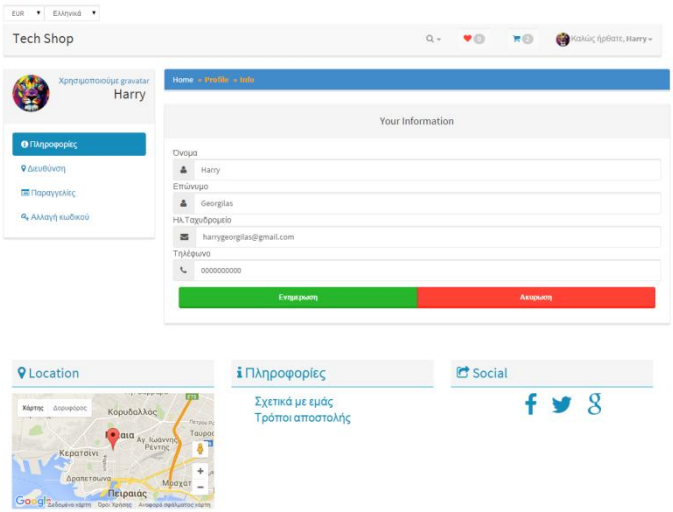
Προβολή προϊόντος



Καλάθι χρήστη

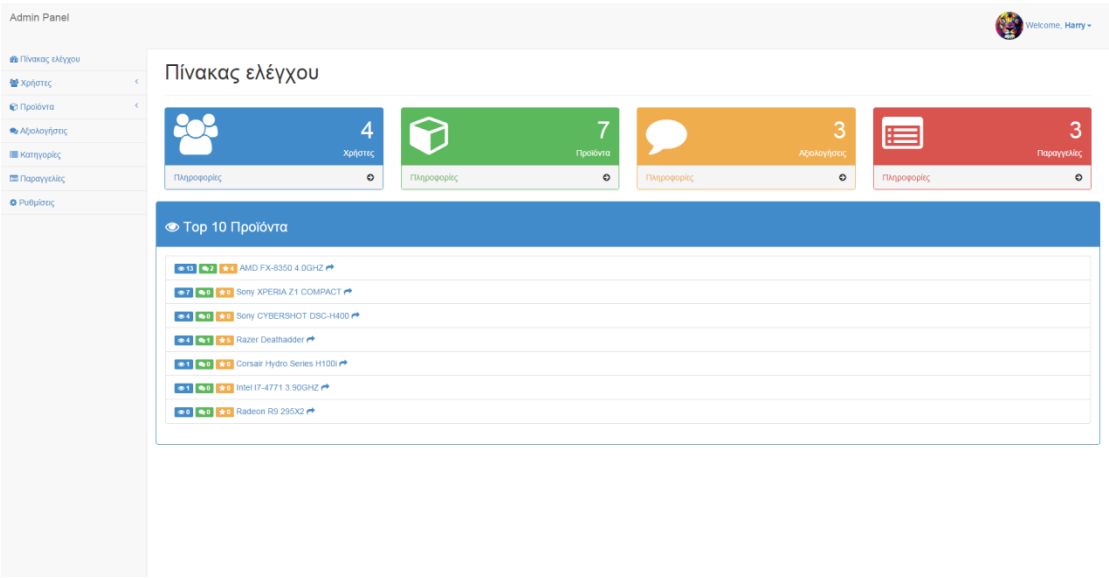


Προβολή λογαριασμού χρήστη

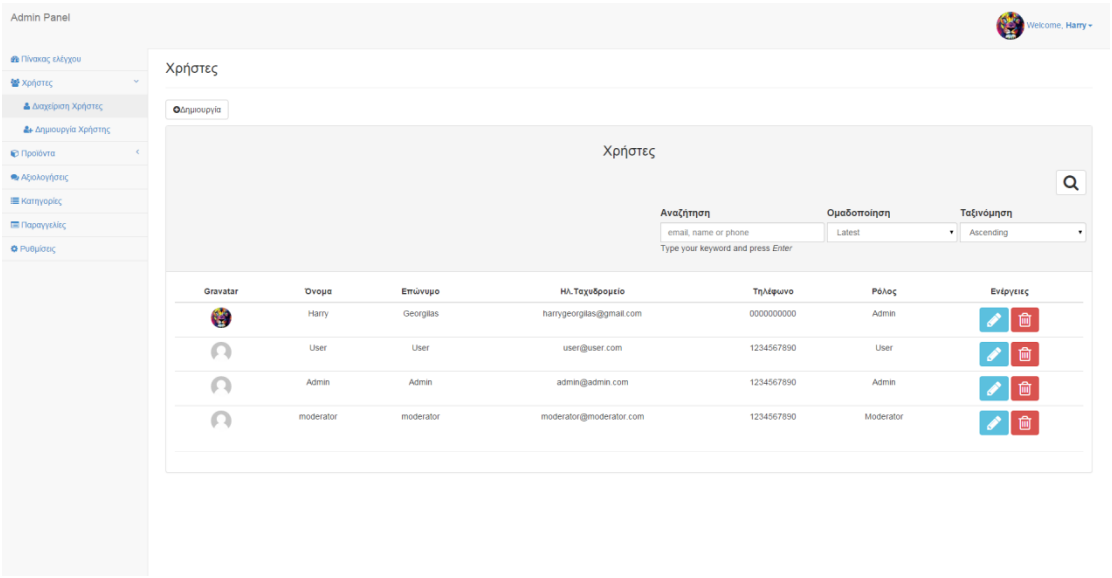


Πίσω πρόσοψη (Backend)

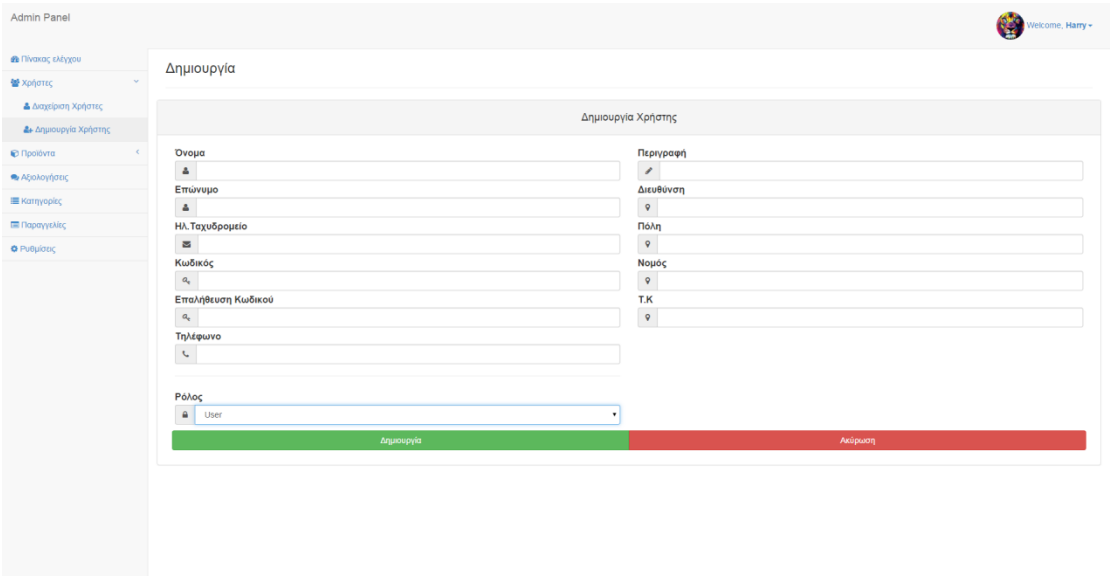
Κεντρικός πίνακας ελέγχου



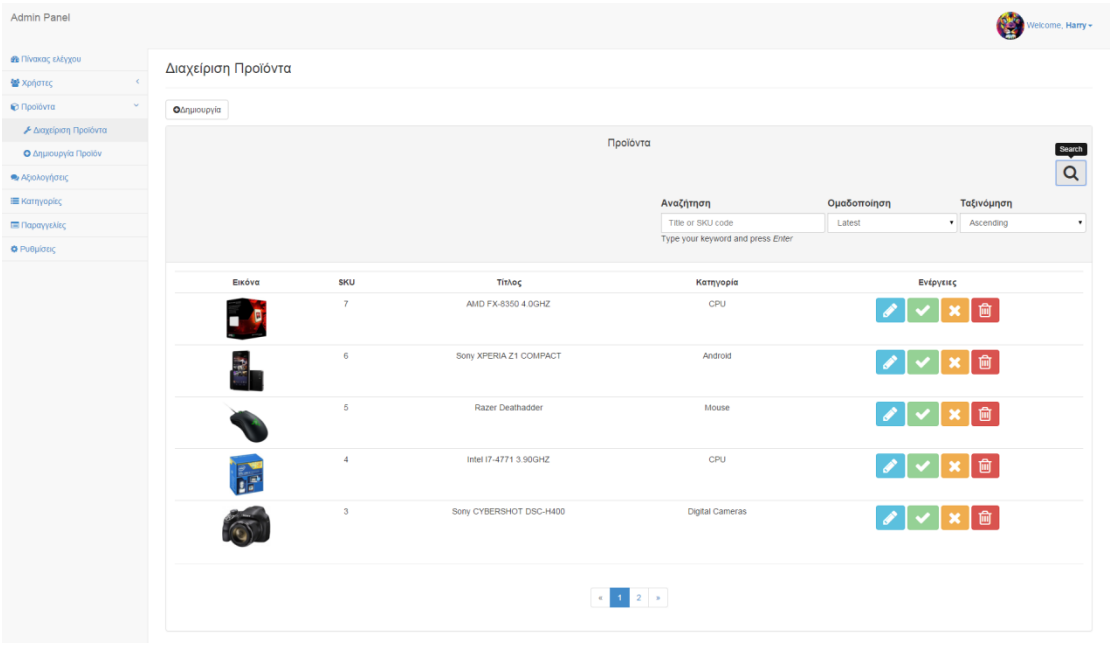
Κεντρική προβολή χρηστών



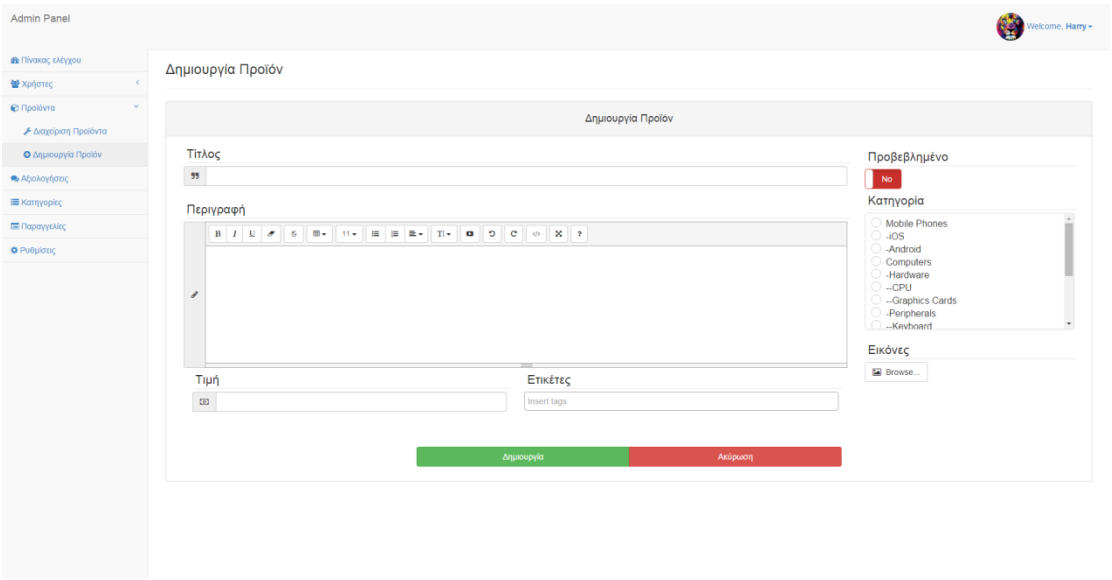
Δημιουργία νέου χρήστη



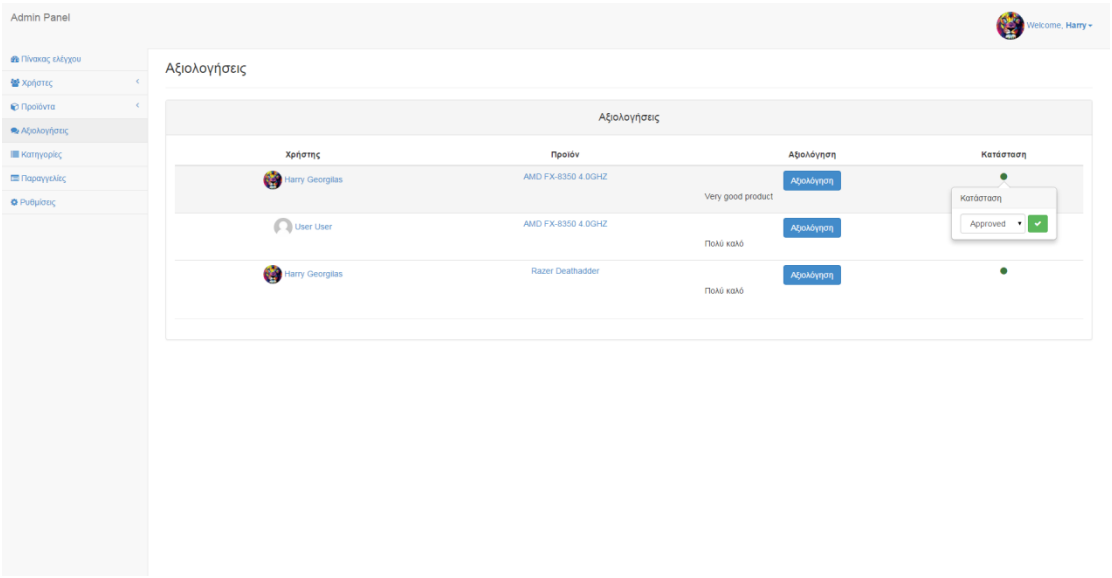
Κεντρική προβολή προϊόντων



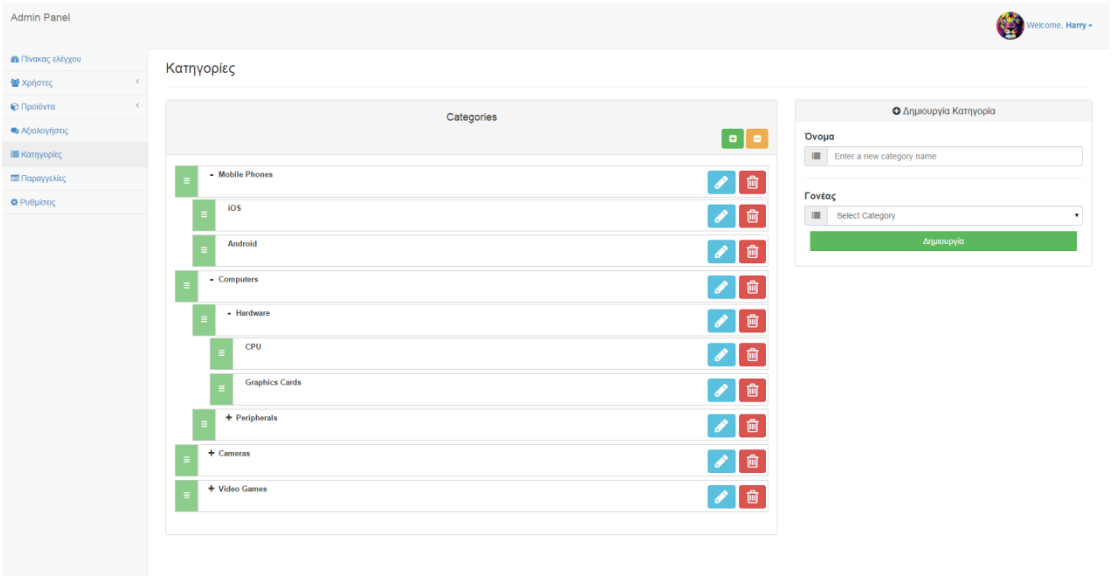
Δημιουργία νέου προϊόντος



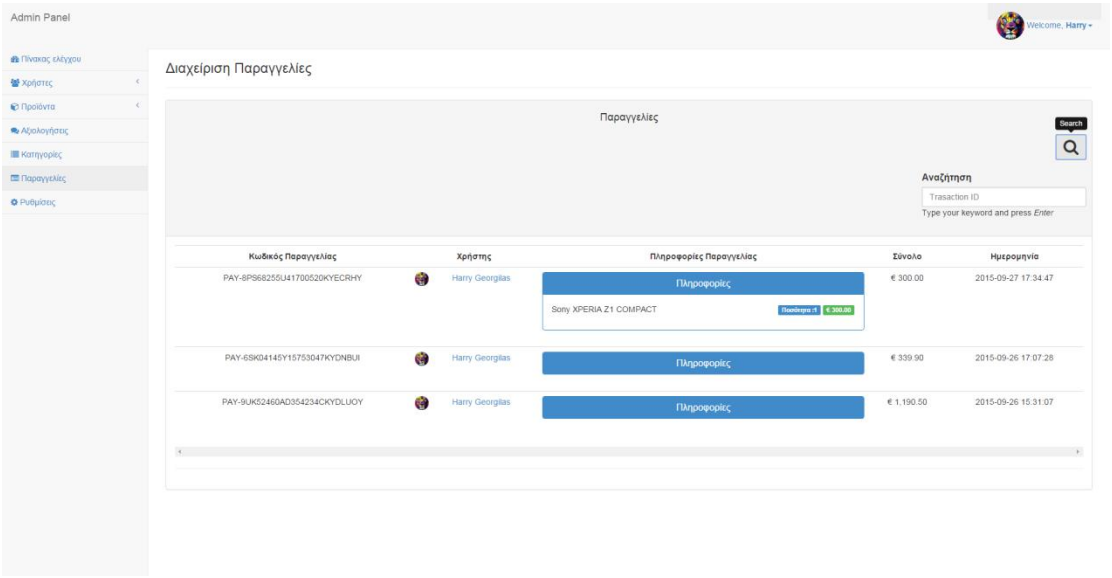
Προβολή αγιολογιών



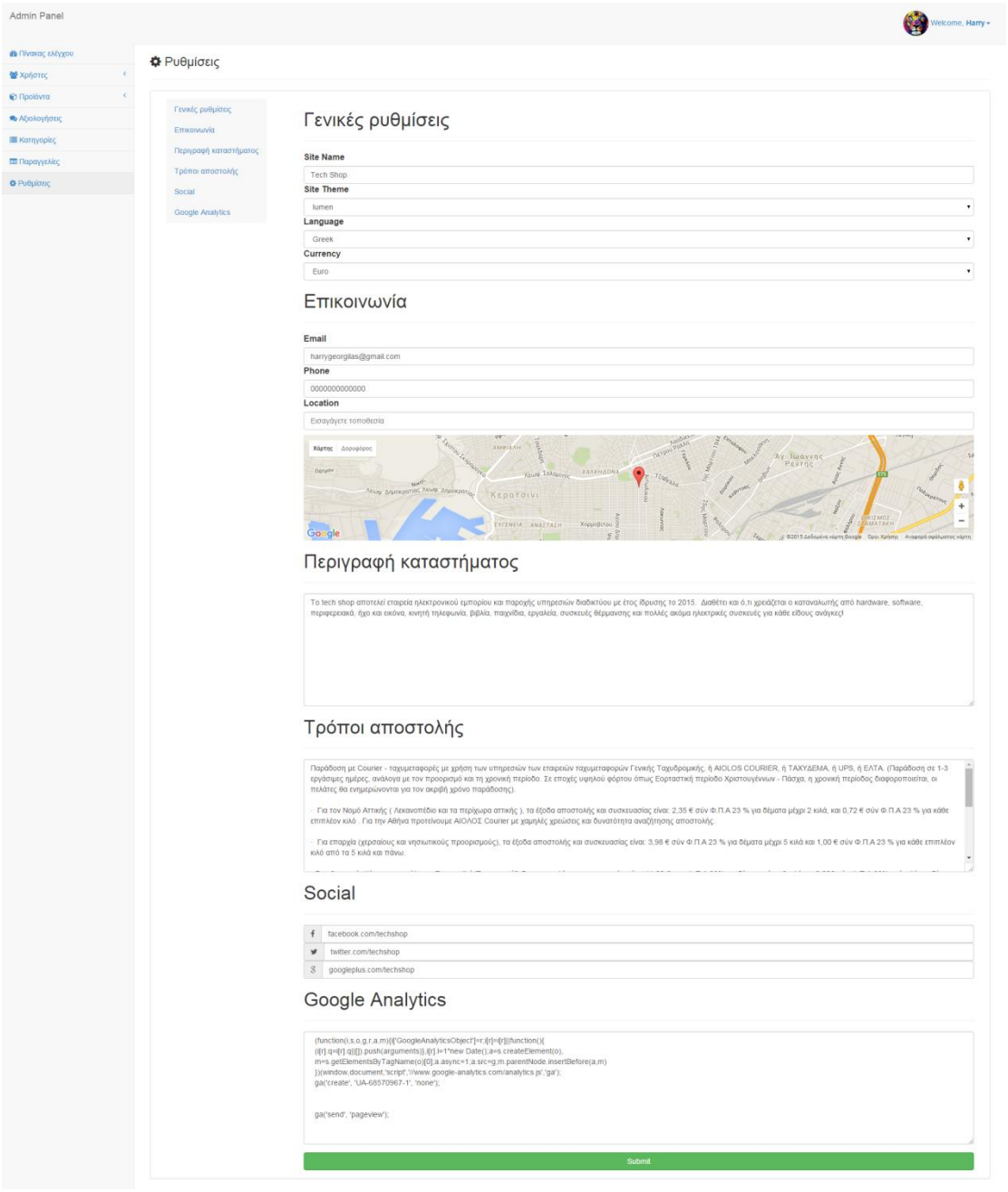
Προβολή Κατηγοριών



Προβολή παραγγελιών



Προβολή γενικών ρυθμίσεων συστήματος



4) Κεφάλαιο 4

4.1 Ασφάλεια εφαρμογής

Η ασφάλεια είναι κύριο κομμάτι κάθε εφαρμογής και πάντα πρέπει να δημιουργούμε δικλίδες ασφαλείας για να θωρακίζουμε την εφαρμογή μας με μεγάλη προσοχή. Οι κακόβουλες τεχνικές που μπορούν να διαταράξουν την ασφάλεια του συστήματος μας είναι πάρα πολλές και επηρεάζουν διαφορετικά μέρη του συστήματος, όπως τον εξυπηρετητή, την βάση ακόμη και την ίδια την εφαρμογή. Το πλαίσιο εργασίας παρέχει κάποιες δικλίδες ασφαλείας όσο αναφορά την βάση των δεδομένων και την εφαρμογή.

4.1.1 Κρυπτογράφηση

Με την χρήση της επέκτασης MCrypt που διαθέτει η PHP το πλαίσιο προσφέρει κρυπτογράφηση AES με την βοήθεια των facades η διαδικασία γίνεται ακόμη πιο εύκολη. Απλά χρησιμοποιώντας τις μεθόδους του `Crypt` façade, για να κρυπτογραφήσουμε ένα αλφαριθμητικό χρησιμοποιούμε την μέθοδο `Crypt::encrypt('secret');` και για να αποκρυπτογραφήσουμε το αλφαριθμητικό χρησιμοποιούμε την μέθοδο `Crypt::decrypt($encryptedValue);`. Πρέπει να επισημανθεί ότι πρέπει να ορίσουμε ένα τυχαίο αλφαριθμητικό κλειδί στο αρχείο `app/config/app.php` που θα χρησιμοποιηθεί από τις αντίστοιχες μεθόδους για την κρυπτογράφηση.

4.1.2 SQL Injection

Μία από τις πιο διαδεδομένες επιθέσεις σε συστήματα που βασίζονται σε βάσεις δεδομένων είναι η επίθεση SQL injection. Ένας κακόβουλος χρήστης μπορεί να περάσει παραμέτρους ή ακόμη και ερωτήματα σε κάποιο πεδίο που καταλήγει σε ένα ερώτημα στην βάση προκαλώντας, για παράδειγμα, την επιστροφή όλων των χρηστών και των στοιχείων τους ή ακόμη και διαγραφή πινάκων. Ας φανταστούμε την περίπτωση που έχουμε ένα «ωμό» sql ερώτημα το οποίο ψάχνει έναν χρήστη στην βάση με αναγνωριστικό το email του και εμείς σε μία φόρμα περιμένουμε το email του χρήστη στην εφαρμογή το ερώτημα μας θα είναι κάπως έτσι `SELECT * FROM user where user.email = $user_email` . Αν ο κακόβουλος χρήστης δώσει στην φόρμα μία τέτοια απάντηση `user@user.com; drop table user;` Αυτομάτως έχουμε χάσει τον πίνακα που αποθηκεύει τα στοιχεία των χρηστών. Το laravel χρησιμοποιεί την επέκταση PDO για να κάνει την διασύνδεση με την βάση που δεν επιτρέπει τέτοιου είδους κακόβουλα ερωτήματα να εκτελεσθούν.

4.1.3 Cross-Site Request Forgery και Cross-Site Scripting

Το cross site Request Forgery αποτελεί κακόβουλη τεχνική που αφήνει εκτεθειμένους του ήδη αυθεντικοποιημένους χρήστες σε επιθέσεις μέσω ψεύτικα κατασκευασμένων υπερσυνδέσεων. Για παράδειγμα έστω ότι βρισκόμαστε σε έναν ισότοπο που διαθέτουμε κάποιου είδους συναλλαγματικό υπόλοιπο, αν έχουμε ήδη συνδεθεί σαν χρήστης και κάποιος στείλει έναν σύνδεσμο που μεταφέρει συναλλαγματικό υπόλοιπο στον δικό του λογαριασμό χωρίς την συγκατάθεση μας πολύ πιθανόν να το πετύχει. Το laravel παρέχει ένα ξεχωριστό csrf token για κάθε συνεδρία. Το csrf token είναι ένα τυχαίο αλφαριθμητικό που ισχύει μόνο για την συγκεκριμένη συνεδρία και ελέγχεται μαζί με τα δεδομένα της φόρμας. Αυτό έχει σαν αποτέλεσμα οι κακόβουλες υπερσυνδέσεις να καθιστούνε άχρηστοι.

Το cross site scripting επιτρέπει σε χρήστες να περάσουν δικά τους script τα οποία κάνουν escape στο περιεχόμενο του html και μπορούν να δημιουργήσουν σοβαρό λειτουργικό πρόβλημα στον σύστημα. Αν σε ένα πεδίο που αποδέχεται σχόλια βάλουμε το παρακάτω javascript `<script>alert(“spam”)</script>` αν το περιεχόμενο του κάνει escape στην σελίδα τότε θα εμφανίζεται κάθε φορά ένας διάλογος που θα εμφανίζει την λέξη spam. Ευτυχώς το Blade engine έχει προνοήσει και δεν αφήνει τα περιεχόμενα να κάνουν escape εκτός αν το επιλέξουμε εμείς.

4.1.4 PHP dotenv

Η βιβλιοθήκη dotenv (.env) είναι ένα πολύ χρήσιμο εργαλείο που επιτρέπει την αποθήκευση όλων των συνθηματικών για το σύστημα μας σε ένα «ειδικό» αρχείο .env το οποίο δεν φαίνεται με συμβατικούς τρόπους στον εξυπηρετητή μας και επίσης δεν εμφανίζει τα στοιχεία στα αρχεία της εφαρμογής μας. Αυτό συμβάλει σημαντικά στην θωράκιση της εφαρμογής και επιτρέπει ευέλικτη αλλαγή στοιχείων από περιβάλλον σε περιβάλλον.

5) Επίλογος

5.1 Συμπεράσματα

Τα πληροφοριακά συστήματα που έχουν σχέση με το ηλεκτρονικό εμπόριο εξελίχθηκαν ραγδαία τα τελευταία χρόνια προσφέροντας εναλλακτικούς τρόπους συναλλαγών. Δημιουργώντας έτσι μια ξεχωριστή εμπειρία αλληλεπίδρασης με το καταναλωτικό κοινό η οποία διέπεται από αμεσότητα (αξιολογήσεις), εχεμύθεια (ανωνυμία) και την δυνατότητα σύγκρισης διαφόρων προϊόντων και προσφορών σε ελάχιστο χρόνο. Επιπλέον πολλά συστήματα προσφέρουν την δυνατότητα προβολής της εξέλιξης της παραγγελίας σε όλα τα στάδια της, από την τιμολόγηση μέχρι την παράδοση.

Στόχος της πτυχιακής ήταν να συμπεριλάβει όλα τα στάδια δημιουργίας ενός συστήματος μέτριας κλίμακας που αφήνει όμως περιθώρια να εξελιχθεί σε κάτι πιο πολύπλοκο. Με την χρήση του πλαισίου εργασίας Laravel, το οποίο αποτέλεσε σημαντική εργαλειοθήκη, η εφαρμογή γράφτηκε με συνεπή τρόπο αξιοποιώντας μοτίβα και αρχιτεκτονικές που συναντάμε και σε μεγαλύτερα προγραμματιστικά έργα. Συγκεκριμένα προσπάθησε να κρατηθεί δομή στον κώδικα που συμπίπτει με το πρότυπο psr0, η βάση διαθέτη κανονικοποιημένη μορφή και όλες οι συναλλαγές με την βάση γίνονται μέσω ασφαλείς διασύνδεσης με την χρήση του προσθέτου PDO που διαθέτει μηχανισμούς εναντίον επιθέσεων SQL injection. Οι προσόψεις υλοποιήθηκαν με βάση την τεκμηρίωση του Twitter bootstrap και διαθέτουν δομή σύμφωνα με τις ενδεδειγμένες πρακτικές συγγραφής HTML. Οι βιβλιοθήκες που χρησιμοποιήθηκαν έχουν ελεγχθεί από το σύστημα Travis-CI και καλύπτουν τις προδιαγραφές για σωστά δομημένο κώδικα.

Η πτυχιακή λειτουργεί σαν προθάλαμος για το πώς λειτουργεί το πλαίσιο εργασίας και σε καμία περίπτωση δεν αναλύθηκαν όλες οι πτυχές και λειτουργίες που προσφέρει, στόχος ήταν η στοιχειώδεις ανάλυση και μελέτη του συστήματος και πώς τα επιμέρους τμήματα συνθέτουν το τελικό αποτέλεσμα.

5.2 Μελλοντικές επεκτάσεις

Το σύστημα σαφώς έχει την δυνατότητα περαιτέρω ανάπτυξης των υπηρεσιών που προσφέρει ή βελτίωσης των ήδη υλοποιημένων. Ο χρόνος που είχα στην διάθεση μου επέτρεψε να φτάσω μέχρι αυτή την μορφή. Μερικές ιδέες για ανάπτυξη των υπηρεσιών είναι οι παρακάτω.

- Χρήση μηχανής αναζήτησης για τα προϊόντα
    - Apache Solr
    - Elastic Search
    - Sphinx
- Υλοποίηση περισσότερων τρόπος πληρωμής εκτός του Paypal
- Σύστημα ελέγχου παραγγελιών
- Τοποικοποίηση εγγραφών στην βάση
- Εσωτερικά στατιστικά χρήσης της πλατφόρμας
- Αυθεντικοποίηση χρηστών με την χρήση κοινωνικών μέσων

6) Βιβλιογραφία

Τεκμηρίωση του πλαισίου εργασίας Laravel από http://laravel.com/docs/4.2

Τεκμηρίωση jquery από https://jquery.com/

Τεκμηρίωση Twitter Bootstrap 3 από http://getbootstrap.com/components/