# Κατασκευή e-shop με PHP με τη χρήση του Laravel framework

Περιεχόμενα

[Κεφάλαιο 1 – Εισαγωγή 5](#_Toc110877850)

[1.1 Αντικείμενο της Πτυχιακής Εργασίας (ΠΕ) 5](#_Toc110877851)

[1.2 Σκοπός της Εργασίας 5](#_Toc110877852)

[1.3 Οργάνωση της Εργασίας 6](#_Toc110877853)

[ΚΕΦΑΛΑΙΟ 2 – Βιβλιογραφική ανασκόπηση τεχνολογιών 7](#_Toc110877854)

[2.1 HTML (HyperText Markup Language) 7](#_Toc110877855)

[2.2 CSS (Cascading Style Sheet) 8](#_Toc110877856)

[2.3 Βιβλιοθήκη Bootstrap 10](#_Toc110877857)

[2.4 Javascript 10](#_Toc110877858)

[2.5 Ajax (Asynchronous Javascript And XML) 12](#_Toc110877859)

[2.6 Apache Web Server 12](#_Toc110877860)

[2.7 Γλώσσα Προγραμματισμού PHP 13](#_Toc110877861)

[2.8 Βάση δεδομένων 14](#_Toc110877862)

[2.9 Σύστημα διαχείρισης βάσης δεδομένων (DBMS) 14](#_Toc110877863)

[2.10 MySQL / MariaDB 15](#_Toc110877864)

[2.11 Το Laravel Framework 15](#_Toc110877865)

[2.12 Δυνατότητες του Laravel 17](#_Toc110877866)

[2.13 Composer 18](#_Toc110877867)

[2.14 Model, View, Controller 20](#_Toc110877868)

[2.15 Model 21](#_Toc110877869)

[2.16 Views 21](#_Toc110877870)

[2.17 Controller 21](#_Toc110877871)

[2.18 Τεχνικές υλοποίησης και συνδυασμός όλων των προαναφερθέντων τεχνολογιών 22](#_Toc110877872)

[2.19 Visual Studio Code 25](#_Toc110877873)

[ΚΕΦΑΛΑΙΟ 3 – Τεχνικά δομικά στοιχεία του έργου 26](#_Toc110877874)

[3.1 Διαδικασία ανάπτυξης 26](#_Toc110877875)

[3.2 Περιγραφή του σχήματος της βάσης δεδομένων 28](#_Toc110877876)

[ΚΕΦΑΛΑΙΟ 4 – Τεχνική ανάλυση εφαρμογής 34](#_Toc110877877)

[4.1 Routes (δρομολόγια αιτημάτων) και middlewares 34](#_Toc110877878)

[4.2 Models 37](#_Toc110877879)

[4.3 Εγγραφή χρήστη 37](#_Toc110877880)

[4.4 Είσοδος χρήστη 40](#_Toc110877881)

[4.5 Αρχική σελίδα (όλα τα προιόντα) 40](#_Toc110877882)

[4.5.1 Αναζήτηση προϊόντος 41](#_Toc110877883)

[4.6 Σελίδα προϊόντος 42](#_Toc110877884)

[4.7 Σελίδα κατηγορίας 42](#_Toc110877885)

[4.8 Προσθήκη στο καλάθι 42](#_Toc110877886)

[4.8.1 Όψη Javascript 43](#_Toc110877887)

[4.8.2 Όψη PHP 44](#_Toc110877888)

[4.8.3 Αυξομείωση ποσότητας στο καλάθι 44](#_Toc110877889)

[4.8.3.1 Όψη Javascript 44](#_Toc110877890)

[4.8.3.2 Όψη PHP 45](#_Toc110877891)

[4.8.4 Διαγραφή προιόντος από το καλάθι 45](#_Toc110877892)

[4.8.5 Προβολή καλαθιού 46](#_Toc110877893)

[4.8.6 Παραγγελία 46](#_Toc110877894)

[4.9 Διαχείριση 47](#_Toc110877895)

[4.9.1 Διαχείριση κατηγοριών 47](#_Toc110877896)

[4.9.2 Διαχείριση προιόντων 48](#_Toc110877897)

[4.10 Σελίδα: Σχετικά με εμάς 49](#_Toc110877898)

[4.11 Παραγγελίες πελάτη 49](#_Toc110877899)

[ΚΕΦΑΛΑΙΟ 5 – Στιγμιότυπα χρήσης 51](#_Toc110877900)

[5.1 Αρχική οθόνη 51](#_Toc110877901)

[5.2 Αναζήτηση 51](#_Toc110877902)

[5.3 Εγγραφή 52](#_Toc110877903)

[5.4 Είσοδος 53](#_Toc110877904)

[5.5 Σελίδα προιόντος 53](#_Toc110877905)

[5.6 Σελίδα κατηγορίας 54](#_Toc110877906)

[5.7 Σελίδα Σχετικά με εμάς 54](#_Toc110877907)

[5.8 Καλάθι 55](#_Toc110877908)

[5.9 Παραγγελία 55](#_Toc110877909)

[5.10 Επιτυχής παραγγελία 55](#_Toc110877910)

[5.11 Παραγγελίες πελάτη 56](#_Toc110877911)

[5.12 Κεντρική οθόνη διαχείρισης 56](#_Toc110877912)

[5.13 Διαχείριση κατηγοριών 57](#_Toc110877913)

[5.13 Επεξεργασία κατηγορίας 57](#_Toc110877914)

[5.14 Εισαγωγή κατηγορίας 58](#_Toc110877915)

[5.15 Διαχείριση προϊόντων 58](#_Toc110877916)

[5.16 Επεξεργασία προϊόντος 59](#_Toc110877917)

[5.17 Εισαγωγή προϊόντος 59](#_Toc110877918)

# Κεφάλαιο 1 – Εισαγωγή

## 1.1 Αντικείμενο της Πτυχιακής Εργασίας (ΠΕ)

Το αντικείμενο της παρούσας πτυχιακής εργασίας είναι η μελέτη και έπειτα η χρήση των τεχνολογιών Ιστού, ή αλλιώς οι web τεχνολογίες, με τις οποίες θα γίνει η υλοποίηση ενός eshop, δηλαδή ενός ηλεκτρονικού καταστήματος που θα εμπορεύεται είδη για ηλεκτρονική αγορά από τους καταναλωτές. Μολονότι ο τομέας του ηλεκτρονικού εμπορίου είναι ανεπτυγμένος, είναι σίγουρο ότι υπάρχει ολοένα και περισσότερος χώρος για δημιουργία και ωρίμανση. Οι περισσότερο γνωστές πλατφόρμες δε χρησιμοποιούν PHP frameworks, παρά κάνουν χρήση αρκετά εξειδικευμένων τεχνολογιών αποκλειστικά προσανατολισμένες στο ηλεκτρονικό εμπόριο. Στην παρούσα ΠΕ γίνεται προσπάθεια να αναπτυχθεί ένα eshop χρησιμοποιώντας το διασημότερο και ίσως το καλύτερο σήμερα framework για τη γλώσσα προγραμματισμού PHP.

Οι τεχνολογίες που επελέγησαν για την κωδικοποίησης της εφαρμογής είναι σύγχρονες και ανοικτού κώδικα (open-source), ενώ χρησιμοποιούνται επί πολλά έτη κυριαρχώντας σε πολλά σημεία έναντι εναλλακτικών τεχνολογιών. Ο server που φιλοξενεί την εφαρμογή-ιστοσελίδα τοπικά είναι ο Apache, η γλώσσα προγραμματισμού που επικοινωνεί με το server είναι η PHP και η βάση δεδομένων είναι η MySQL. Στο επίπεδο πελάτη (client layer) χρησιμοποιούνται HTML, Javascript και CSS. Επιπλέον open-source βιβλιοθήκες, τεχνολογίες και εργαλεία ολοκληρώνουν την εμπειρία του πελάτη-χρήστη της εφαρμογής. Το περιβάλλον γραφής-κειμενογράφος (IDE) του λογισμικού είναι το Visual Studio Code.

## 1.2 Σκοπός της Εργασίας

Σκοπός της πτυχιακής εργασίας είναι να δώσει μια πρόταση για τη λογισμική υποστήριξη εταιρειών που έχουν κατάστημα και επιθυμούν να δέχονται ηλεκτρονικές πληρωμές από πελάτες μέσω του διαδικτύου. Η μεγαλύτερη έμφαση έχει δοθεί στο κομμάτι της παρουσίασης του ηλεκτρονικού καταστήματος προς εξυπηρέτηση των πελατών και την αγορά προϊόντων. Έχει γίνει επίσης πρόνοια για την ανάπτυξη του διαχειριστικού συστήματος που προορίζεται για χρήση από το προσωπικό, το οποίο δεν απαιτείται να έχει ιδιαίτερες γνώσεις ή να εκπαιδευτεί στο λογισμικό.

Ένα επιπρόσθετο όφελος που προκύπτει κατά τη διαδικασία υλοποίησης της εν λόγω εφαρμογής είναι η εκμάθηση και η εξοικείωση τις με γλώσσες προγραμματισμού που απαιτούνται για τη δημιουργία ολοκληρωμένων συστημάτων που προορίζονται για χρήση μέσα από έναν ιστότοπο. Παράλληλα, η χρήση εργαλείων ανοιχτού κώδικα, τα οποία επιλέχθηκαν για τη δημιουργία του προγράμματος, συνιστά μικρό κόστος χρησιμοποίησής τους.

## 1.3 Οργάνωση της Εργασίας

Όσον αφορά την δομή της εν λόγω Πτυχιακής Εργασία (ΠΕ) είναι οργανωμένη ως εξής:

Στο επόμενο κεφάλαιο θα γίνει η βιβλιογραφική επισκόπηση και θεωρητική περιγραφή των τεχνολογιών και των γλωσσών προγραμματισμού που χρησιμοποιήθηκαν για την επιτυχή περάτωση του συστήματος.

Στο τρίτο κεφάλαιο θα γίνει εισαγωγή στα τεχνικά δομικά στοιχεία με τα οποία «χτίστηκε» η εφαρμογή όπως και των βοηθητικών εργαλείων και βιβλιοθηκών που είναι βοηθητικά και συχνά απαραίτητα στη δημιουργία σχετικού λογισμικού. Πιο συγκεκριμένα θα γίνει ανάλυση της βάσης δεδομένων και της αρχιτεκτονικής του έργου σε επίπεδο φακέλων και αρχείων

Στο τέταρτο κεφάλαιο θα γίνει εμβάθυνση στην τεχνική ανάλυση των βασικότερων και κρισιμότερων αρχείων του έργου που περιλαμβάνει την επεξήγηση κώδικα.

Στο πέμπτο κεφάλαιο θα γίνει παράθεση από στιγμιότυπα οθόνης από τη χρήση της εφαρμογής.

# ΚΕΦΑΛΑΙΟ 2 – Βιβλιογραφική ανασκόπηση τεχνολογιών

## 2.1 HTML (HyperText Markup Language)

Η HTML είναι μια γλώσσα σημάνσεων (markup language) και αποτελεί παιδί του πατέρα του, της γλώσσας XML. Eκτελείται σε όλα τα λειτουργικά συστήματα και είναι αγνωστική αναφορικά με τη συσκευή στην οποία εκτελείται και αποδίδεται. Χρησιμοποιείται για τον ορισμό δομημένων εγγράφων και έχει ευρύτατη χρήση στον Ιστό, καθώς όλες οι ιστοσελίδες βασίζονται σε αυτή τη γλώσσα. Οι περιηγητές Ιστού χρησιμοποιούν την HTML για να αποδώσουν κείμενο, εικόνες, βίντεο και άλλου είδους περιεχόμενο σε ιστοσελίδες. Τα HTML στοιχεία είναι τα πλέον θεμελιώδη δομικά συστατικά του Διαδικτύου και χρησιμοποιούνται ως ετικέτες κωδικοποιημένες με γωνιακές αγκύλες. Υπάρχουν 3 κύρια κομμάτια που αποτελούν ένα στοιχείο: η ετικέτα έναρξης, το περιεχόμενο και η ετικέτα κλεισίματος. Η ετικέτα έναρξης έχει την ίδια ονομασία με την ετικέτα κλεισίματος και εντός των ετικετών εσωκλείεται το περιεχόμενο, το οποίο ονομάζεται και ως κόμβος του εγγράφου HTML. Ένας HTML κόμβος εκτός από περιεχόμενο δύνανται να έχει και ιδιότητες και λέμε ότι αυτές οι ιδιότητες χαρακτηρίζουν έναν κόμβο. Η πιο συχνή ιδιότητα είναι η κλάση (class). Για παράδειγμα το: <p class=“comment”> είναι μια ετικέτα έναρξης. Ακολουθεί το περιεχόμενο της ετικέτας: “my comment” και ο κόμβος τελειώνει ως </p>. Η ετικέτα κλεισίματος είναι παρόμοια με την ετικέτα έναρξης, αλλά έχει μια διαγώνια κάθετο (slash) πριν το όνομα της ετικέτας. Συνεπώς το: <p class=”comment”>my comment</p> είναι ένας μοναδικός και αυτόνομος κόμβος HTML.

Η HTML5 είναι η νεότερη έκδοση της γλώσσας και δημοσιεύτηκε το 2008 με στόχο τη βελτίωση της υποστήριξης της γλώσσας με τα τελευταία χαρακτηριστικά διαχείρισης πολυμέσων. Η νέα έκδοση είχε επίσης ως στόχο να καταστήσει τη γλώσσα πιο ευανάγνωστη για τον άνθρωπο αφαιρώντας την αυστηρότητα του προτύπου XHTML διατηρώντας όμως τη συμβατότητα με παλαιότερο λογισμικό. Η HTML5 συμπεριλαμβάνει αναλυτικά επεξεργαστικά μοντέλα για την ενθάρρυνση περισσότερο δια-πλατφορμικών υλοποιήσεων. Επεκτείνει, βελτιώνει και εξορθολογεί τη σήμανση των εγγράφων και παρουσιάζει μία λογισμική γέφυρα (APIs, application programming interfaces) για την επικοινωνία μεταξύ συστημάτων και τη δημιουργία περίπλοκων διαδικτυακών εφαρμογών.

Η HTML5 ορίζει μια πληθώρα νέων στοιχείων και ιδιοτήτων έχοντας ως σκοπό να κάνει τη γλώσσα πιο απλή και σημασιολογική για τους προγραμματιστές αλλά και για τους επεξεργαστές περιεχομένου όπως οι μηχανές αναζήτησης και αναγνώστες οθονών. Κάθε στοιχείο της HTML5 ανήκει σε 0 ή περισσότερες κατηγορίες που ομαδοποιούν στοιχεία με παρόμοια χαρακτηριστικά:

**Περιεχόμενο μετα-δεδομένων**: Το περιεχόμενο που ορίζει τη συμπεριφορά ενός HTML εγγράφου και τις συσχετίσεις μεταξύ άλλων εγγράφων ή επικοινωνεί άλλη πληροφορία από αυτή του εγγράφου.

**Περιεχόμενο ροής:** Η πλειοψηφία των στοιχείων του περιεχομένου που χρησιμοποιείται εντός του κόμβου body στο έγγραφο. Παραδείγματα: a, article, audio, div, header, form, nav, p.

**Περιεχόμενο ενοτήτων**: Εδώ ανήκει το περιεχόμενο που ορίζει το πεδίο εφαρμογής της επικεφαλίδας και του υποσέλιδου μιας ενότητας. Παραδείγματα: article, aside, nav, section

**Περιεχόμενο επικεφαλίδας**: Το περιεχόμενο που ορίζει την επικεφαλίδα μιας ενότητας. Πχ. h1, h2, hgroup.

**Περιεχόμενο φράσεων**: Το περιεχόμενο που αποθηκεύει και δηλώνει τη σήμανση του κειμένου στο έγγραφο. Παραδείγματα: abbr, audio, canvas, img, em.

**Ενσωματωμένο περιεχόμενο**: Το περιεχόμενο το οποίο εισάγει άλλο πόρο ή προσθέτει περιεχόμενο από άλλο λεξιλόγιο εντός του εγγράφου.

**Διαδραστικό περιεχόμενο**: Το περιεχόμενο το οποίο προορίζεται για διάδραση με το χρήστη. Παραδείγματα: a, button, menu, select.

## 2.2 CSS (Cascading Style Sheet)

Το CSS χρησιμοποιείται για τον ορισμό της οπτικής εμφάνισης των HTML εγγράφων, δηλαδή το πώς τα στοιχεία αποδίδονται στην οθόνη. Επιτρέπει τον σχεδιαστή να ορίσει με ακρίβεια τη γραμματοσειρά, τα χρώματα, τα περιθώρια, τα περιγράμματα κλπ. όπως επίσης και να ορίσει τη σχεδίαση της διάταξης, τη διαίρεση του κειμένου σε γραμμές και στήλες αλλά και την κίνηση στατικών στοιχείων. Καλώς διαχωρισμένα επίπεδα διατάξεων επιτρέπουν τη δομή του εγγράφου να είναι καθαρή και πλούσια σε σχεδίαση και εξαλείφουν την ανάγκη για τον προγραμματισμό μεμονωμένων στοιχείων και παρέχουν ικανοποιητική συμβατότητα λειτουργιών για παλαιότερους περιηγητές Ιστού. Επιτρέποντας στον browser το χειρισμό πλούσιας γραφικής διεπαφής και των εφέ κίνησης, οι προγραμματιστές μπορούν να βελτιστοποιήσουν την απόδοση και το responsiveness των εφαρμογών τους αφού ο browser μπορεί να κάνει χρήση των πιο αποδοτικών τεχνικών της πλατφόρμας για το χειρισμό αυτών των λειτουργιών. Το CSS χρησιμοποιείται επίσης και από Javascript προγραμματιστές μιας που τα στυλ CSS μπορούν να προγραμματιστούν με σκοπό την κίνηση του HTML. Επί παραδείγματι, το περιεχόμενο ενός εγγράφου μπορεί να κυλίσει στο DOM της σελίδας από μία ορισμένη γωνία. Παρατίθενται τα παρακάτω τα κυριότερα συστατικά και οι προδιαγραφές του W3C CSS:

Επιλογείς (selector): Οι CSS επιλογείς είναι μοτίβα που χρησιμοποιούνται για την αντιστοίχιση των στοχείων στο δέντρο DOM. Τα μοτίβα χρησιμοποιούνται για την εφαρμογή κανόνων στα αντιστοιχισμένα στοιχεία. Οι επιλογείς δύνανται επίσης να χρησιμοποιηθούν στον κώδικα JavaScript για την επιλογή στοιχείων για κωδικοποίηση.

Μεταμορώσεις (Transforms): Οι δυσδιάστατες και οι τρισδιάστατες μεταμορφώσεις επιτρέπουν τα στοιχεία να μεταφορτωθούν στο χώρο των 2 και 3 διαστάσεων. Τα στοιχεία δύνανται να μεταφραστούν (translated), να περιστραφούν και να αυξομειωθούν στον υφιστάμενο άξονά τους αποδίδοντας έτσι πλούσιο και διαδραστικό περιεχόμενο στο HTML έγγραφο.

Μεταβάσεις (Transitions): Οι CSS μεταβάσεις επιτρέπουν στα στοιχεία να έχουν ομαλή κίνηση για ορισθέν χρονικό διάστημα με σκοπό την παραγωγή πλούσιων εφέ κινήσεων.

Media Queries: Χρησιμοποιούνται για την εφαρμογή CSS κανόνων σε συγκεκριμένες συσκευές και διαστάσεις, μεγέθη, προσανατολισμό, αναλογίας, βάθος χρωμάτων κλπ. Χρησιμοποιώντας τα media queries αποφεύγουμε την πιθανή αυξανόμενη ποσότητα του HTML για τη μορφοποίηση του εγγράφου για διαφορετικές συσκευές που άλλοτε θα απαιτείτο για την υποστήριξή τους. Για παράδειγμα, η διάταξη μιας ιστοσελίδας μπορεί να έχει περισσότερο οριζόντιο χώρο σε έναν desktop browser, αλλά μέσω των media queries μπορούμε να οριοθετήσουμε τα μεμονωμένα κομμάτια της διάταξης σε κάθετη μορφή για την αποφυγή του οριζόντιου scrolling σε κινητές συσκευές.

## 2.3 Βιβλιοθήκη Bootstrap

Το Bootstrap είναι ένα front end Framework για HTML και CSS που χρησιμοποιείται για γρήγορη και ευκολότερη ανάπτυξη ιστοσελίδων. Περιλαμβάνει πληθώρα από υλοποιημένα στοιχεία χρησιμοποιώντας HTML,CSS και JavaScript όπως μενού πλοήγησης, κουμπιά κ.α. Και υποστηρίζεται από όλους τους φυλλομετρητές (browsers). Το Bootstrap αναπτύχθηκε από τους Mark Otto και Jacob Thornton στα μέσα του 2010 για λογαριασμό του Twitter ως ένα Framework για την εξασφάλισή μιας ενιαίας αισθητικής στις διάφορες λειτουργίες του. Το Αύγουστο του 2011 εκδόθηκε σαν έργο ανοικτού κώδικα (open source project) και το Φεβρουάριο του 2011 έγινε το πιο δημοφιλές έργο στο GitHub. Το Bootstrap διαθέτει διάφορους τρόπους χρήσης ανάλογα με επίπεδο του χρήστη. Αυτός μπορεί να χρησιμοποιήσει τα ήδη έτοιμα στοιχεία που παρέχει το bootstrap είτε να χρησιμοποιήσει τα κατάλληλα tags στο κώδικα HTML για την μορφοποίηση της ιστοσελίδας.

## 2.4 Javascript

Η Javascript (JS εν συντομία) είναι μια γλώσσα που προστίθεται σε ιστοσελίδες από προγραμματιστές για τον προγραμματισμό εφαρμογών οποιασδήποτε κλίμακας και μεγέθους και πρόκειται για κώδικα που εκτελείται στην πλευρά του χρήστη (client side), δηλαδή στο μηχάνημά του μέσω του browser που ο χρήστης χρησιμοποιεί και δεν απαιτεί καμία επικοινωνία με το διακομιστή, ο οποίος παραδίδει (ή «σερβίρει») την ιστοσελίδα. Το μεγαλύτερο ατού της είναι ότι είναι γλώσσα ανοικτού κώδικα (open-source) και είναι ίσως το πιο σημαντικό χαρακτηριστικό σε όλες τις γλώσσες προγραμματισμού. Αυτό σημαίνει ότι υπάρχει ενεργή κοινότητα που συνεχώς προχωρά την ανάπτυξή της και βοηθά στη βελτίωση των υφιστάμενων λειτουργιών και στη διόρθωση των λαθών που παρατηρείται στην ενσωμάτωσή τους από τους φυλλομετρητές Ιστού.

Πρόκειται για μία από τις 3 πιο θεμελιώδεις τεχνολογίες του Ιστού (HTML, CSS και JavaScript) και έχει ως κύριο στόχο να κάνει τις ιστοσελίδες πιο διαδραστικές, ελκυστικές και πλούσιες σε περιεχόμενο με αποτέλεσμα να καταστεί απαραίτητη σε κάθε εφαρμογή Ιστού. Έχει υιοθετηθεί από όλους του φυλλομετρητές Ιστού και αρχικά είχε δημιουργηθεί για την εκτέλεση κώδικα στην πλευρά του χρήστη. Σήμερα όμως, λόγω της δημοφιλίας που έλαβε, χρησιμοποιείται και στον προγραμματισμό διαδικτυακών εφαρμογών στην πλευρά του server (server side) αλλά και για τον προγραμματισμό desktop συστημάτων και εφαρμογές σε κινητά τηλέφωνα με τρανταχτά παραδείγματα το Node.js και το React.js. Τα τελευταία χρόνια ορισμένα συστήματα βάσεων δεδομένων όπως η MongoDB και η CouchDB χρησιμοποιούν Javascript για μεμονωμένες λειτουργίες αλλά και για λειτουργίες ερωτημάτων στη βάση δεδομένων.

Σε αντίθεση με τις παραδοσιακές γλώσσες προγραμματισμού, όπου ο κώδικας γίνεται compile και τον μεταφράζει σε bytecode για την κατανόησή του από τη μηχανή, η Javascript είναι διερμηνευόμενη γλώσσα κατά το χρόνο εκτέλεσης στο μηχάνημα του χρήστη και κάθε εντολή εκτελείται σε πραγματικό χρόνο από τον browser. Στην περίπτωση λάθους στον κώδικα (πχ. συντακτικό λάθος), ο κώδικας πριν το λάθος δεν επηρεάζεται σε αντίθεση με τον κώδικα που ακολουθεί του λάθους και αυτό συντελεί στον τερματισμό της εφαρμογής. Λόγω της περιπλοκότητας και της διαφορετικότητας της εν λόγω γλώσσας σε σχέση με άλλες, έχει αναπτυχθεί ένα μεγάλο πλήθος από Javascript βιβλιοθήκες με σκοπό την απλούστευση, την επαναχρησιμοποίηση κώδικα κατά μήκος της εφαρμογής, την καλύτερη απόδοση και χρήση πόρων εκ μέρους των λειτουργιών και τη χρήση λιγότερου του κώδικα από την πλευρά του προγραμματιστή και αυτό συμβάλλει στην ταχύτερη και παραγωγή πιο ποιοτικού κώδικα για τον προγραμματισμό σύνθετων διαδικτυακών εφαρμογών. Συνεπώς ο σκελετός της εφαρμογής είναι πιο εύκολα διαχειρίσιμος από μεγάλες ομάδες προγραμματιστών και επιτρέπει σε αυτούς να ακολουθούν συγκεκριμένους κανόνες και code standards και δομή αρχείων με απώτερο στόχο τόσο της μείωσης της περιπλοκότητας της εφαρμογής όσο και της μελλοντικής επεκτασιμότητας αλλά και την αποφυγή λαθών (bugs).

## 2.5 Ajax (Asynchronous Javascript And XML)

Είναι η τεχνική με την οποία δίνεται στον προγραμματιστή η δυνατότητα να δημιουργεί σημαντικά πιο γρήγορες και δυναμικές ιστοσελίδες περιορίζοντας τον όγκο δεδομένων που ανταλλάσσει ο server με τον browser του χρήστη. Η ιδιαίτερη τεχνική αυτή επιτρέπει την ανανέωση περιεχομένων μιας ιστοσελίδας χωρίς αυτή να ανανεωθεί ολόκληρη. Η υλοποίηση της τεχνικής AJAX γίνεται με τον συνδυασμό του αντικειμένου (object) XMLHttpRequest (πραγματοποιεί την ασύγχρονη επικοινωνία με τον server), την Javascript/DOM (αλληλεπιδρά με τα δεδομένα και τα εμφανίζει), την CSS (μορφοποιεί τα προς εμφάνιση δεδομένα) και την XML (συχνά χρησιμοποιείται για την μεταφορά δεδομένων). Η χρήση της τεχνικής AJAX επιτρέπει σε μια ιστοσελίδα να ανανεώνεται ασύγχρονα (asynchronously) ανταλλάσσοντας στο παρασκήνιο μικρού όγκου δεδομένα με τον server επιτρέποντας να ανανεώνονται μέρη της ιστοσελίδας (ένα div για παράδειγμα), χωρίς να ανανεώνεται η σελίδα. Η τεχνική AJAX χρησιμοποιείται πολύ συχνά σε σελίδες που περιέχουν φόρμες που άλλοτε θα έστελναν “παραδοσιακά” αιτήματα σε server.

## 2.6 Apache Web Server

Ο «Apache Web Server» είναι αυτό ακριβώς που δηλώνει το όνομά του. Πρόκειται δηλαδή για έναν εξυπηρετητή «server» του παγκόσμιου Ιστού «Web». Με τον όρο «server» εννοούμε ηλεκτρονικούς υπολογιστές που φιλοξενούν ιστοσελίδες. Ο όρος αυτός χρησιμοποιείται και για το μηχάνημα εξυπηρετητή «hardware» αλλά και για το πρόγραμμα «software».

**Χαρακτηριστικά και λειτουργίες του Apache HTTP**

Ο Apache διαθέτει ποικιλία χαρακτηριστικών και μπορεί να υποστηρίξει μια μεγάλη γκάμα εφαρμογών με τις οποίες και συνεργάζεται. Οι δυνατότητες του προγράμματος αυτού καθαυτού και τα χαρακτηριστικά του δεν είναι και τόσο πολλά. Ένα από τα βασικότερα χαρακτηριστικά του όμως, το οποίο και του δίνει μεγάλες δυνατότητες, είναι ότι μπορεί να προσαρμόσει επάνω του πολλές προσθήκες προγραμμάτων «modules», τα οποία με τη σειρά τους παρέχουν διαφορετικές λειτουργίες. Μερικά από τα πιο γνωστά «modules» του «Apache HTTP» είναι τα «modules» πιστοποίησης, όπως για παράδειγμα τα mod\_access, mod\_auth, mod\_digest κ.λπ. Παρέχει επίσης «SSL» σε «TLS» μέσω των mod\_ssl, και mod\_proxy, πραγματοποιεί ανακατευθύνσεις διευθύνσεων «URL rewrites» μέσω του mod\_rewrite, καταγραφές συνδέσεων μέσω του mod\_log\_config, συμπίεση αρχείων μέσω του mod\_gzip και πολλά άλλα modules τα οποία διατίθενται είτε από το «Apache Software Foundation», είτε από τρίτες εταιρίες λογισμικού. Ένα άλλο χαρακτηριστικό – δυνατότητα του «Apache HTTP», είναι ότι μπορεί να εγκατασταθεί σε διάφορα λειτουργικά συστήματα. Ο «Apache HTTP» υποστηρίζει επίσης αρκετές διάσημες εφαρμογές και γλώσσες προγραμματισμού όπως MySQL, PHP, Perl, Python κ.λπ.

## 2.7 Γλώσσα Προγραμματισμού PHP

Η PHP είναι μία γλώσσα σεναρίων, που τρέχει στον web server που βρίσκεται εγκατεστημένο το site. Χρησιμοποιείται για τον προγραμματισμό διαδικτυακών εφαρμογών και ιστοσελίδων με δυναμικό περιεχόμενο. Δεν διαμορφώνει αισθητικά τη σελίδα, αλλά παρέχει δυνατότητες όπως η εγγραφή χρηστών σε μία βάση δεδομένων και η εμφάνιση της τοπικής ώρας του server. Η εκτέλεση της PHP γίνεται με τη σειρά εμφάνισης του κώδικα, εκτός από τις συναρτήσεις που εκτελούνται όταν κληθούν. Ο κώδικας γράφεται μεταξύ των ετικετών αρχής “”. Για τις μεταβλητές δεν χρειάζεται να δηλωθεί ο τύπος τους και ξεκινούν με το σύμβολο για το δολάριο ($). Μεγάλη προσοχή δίνεται μεταξύ κεφαλαίων και πεζών, καθώς η PHP είναι case sensitive. Η PHP μπορεί να ενσωματωθεί στα HTML αρχεία, τα οποία αποθηκεύονται και καλούνται με την κατάληξη .php . Η ενσωμάτωση τους μπορεί να γίνει είτε σε οποιοδήποτε σημείο του HTML αρχείου, με τη δυνατότητα να εκτελείται υπό συγκεκριμένες συνθήκες, όπως για παράδειγμα το πάτημα ενός κουμπιού, είτε με την εντολή “include” με την οποία θα εκτελείται ένα εξωτερικό αρχείο PHP κάθε φορά που θα ανοίγει το HTML αρχείο. Όταν ζητηθούν τα αρχεία με κατάληξη .php από τον Client, ο web server καλεί τον PHP Interpreter και του παρέχει το .php αρχείο και πληροφορίες από τις κεφαλίδες τις HTTP αίτησης. Εκτελείται ο κώδικας PHP και το αποτέλεσμα του μπαίνει στον HTML κώδικα, ο οποίος μεταφέρεται στον buffer εξόδου. Ο τελευταίος μαζί με τις πληροφορίες κεφαλίδων της HTTP αίτησης, δίνονται στον Web Server και αυτός τις στέλνει στον Client με χρήση του πρωτοκόλλου HTTP. Αποτέλεσμα αυτών είναι η εμφάνιση στον browser μίας κοινής HTML σελίδας.

## 2.8 Βάση δεδομένων

Μία βάση δεδομένων είναι μία οργανωμένη συλλογή δεδομένων σε ψηφιακή μορφή. Τα δεδομένα είναι οργανωμένα σε ένα μοντέλο που αναπαριστά σχετικές όψεις της πραγματικότητας, π.χ. η διαθεσιμότητα των δωματίων σε ένα ξενοδοχείο, με τρόπο που υποστηρίζει επεξεργασίες που απαιτούν πληροφορίες, όπως η αναζήτηση ξενοδοχείων που έχουν διαθεσιμότητα. Ο όρος βάση δεδομένων εφαρμόζεται στα δεδομένα και τις δομές δεδομένων τους. Η χρήση τους είναι ευρέως διαδεδομένη σε σχεδόν κάθε τεχνολογία και όλες οι γλώσσες server τις υποστηρίζουν εγγενώς διαθέτοντας συναρτήσεις για διάφορα συστήματα βάσεων δεδομένων.

## 2.9 Σύστημα διαχείρισης βάσης δεδομένων (DBMS)

Πρόκειται για ένα πακέτο λογισμικού με προγράμματα που ελέγχουν τη δημιουργία, τη συντήρηση και τη χρήση βάσης δεδομένων. Δίνει τη δυνατότητα σε οργανισμούς να αναπτύσσουν με ευκολία βάσεις δεδομένων με διάφορες εφαρμογές και σε διαφορετικά προγράμματα χρηστών να έχουν ταυτόχρονη πρόσβαση στην ίδια βάση δεδομένων. Το DBMS χρησιμοποιεί πληθώρα από μοντέλα βάσεων δεδομένων, όπως το σχεσιακό (RDBMS) ή το αντικειμενοστραφές (OODBMS) για την καταλληλότερη περιγραφή των εφαρμογών που αναπτύσσουν. Τυπικά υποστηρίζει γλώσσες ερωτημάτων (SQL), οι οποίες είναι γλώσσες υψηλού επιπέδου που απλοποιούν αισθητά τη συγγραφή εφαρμογών σε βάσεις δεδομένων, αλλά και την οργάνωση όπως και την ανάκτηση και την παρουσίαση της πληροφορίας. Παρέχει δυνατότητες για έλεγχο πρόσβασης και ασφάλειας στα δεδομένα ενισχύοντας την ακεραιότητά τους διαχειρίζοντας τις ταυτόχρονες λειτουργίες και ανακτώντας τη ΒΔ ύστερα από σφάλματα στο λογισμικό.

## 2.10 MySQL / MariaDB

Η MySQL είναι ένα σχεσιακό σύστημα διαχείρισης βάσης δεδομένων (RDBMS) που “τρέχει” ως εξυπηρετητής παρέχοντας πρόσβαση σε πλειάδα χρηστών σε βάσεις δεδομένων. Είναι το πιο διαδεδομένο ανοικτού κώδικα σύστημα διαχείρισης βάσης δεδομένων και ανήκει στην οικογένεια δομημένων ερωτημάτων (SQL). Θεωρείται ως ένα αρκετά ισχυρό, γρήγορο, αξιόπιστο και ευέλικτο RDBMS, παρέχοντας υψηλή απόδοση με πολυνηματικές λειτουργίες. Χρησιμοποιείται για διασύνδεση με web εφαρμογές και προτιμάται για την ταχύτητα και την ασφάλειά του ικανό να ανταπεξέρχεται σε ερωτήματα με μεγάλα μεγέθη δεδομένων.

## 2.11 Το Laravel Framework

Μιας που η περιπλοκότητα μιας εφαρμογής ηλεκτρονικής κράτησης είναι κάτι που λαμβάνεται σοβαρά υπόψη τόσο από άποψης ποσότητας χαρακτηριστικών όσο και από την πλευρά της ακεραιότητας των δεδομένων και των σωστών λειτουργιών, έγινε αναζήτηση ενός framework, το οποίο θα μπορούσε να μειώσει την ποσότητα κώδικα, ενώ θα δημιουργούσε ένα επιπλέον επίπεδο αφαιρετικότητας με σκοπό την απλοποίηση προγραμματιστικών ρουτινών και θα επέτρεπε τη συγκέντρωση στα κρισιμότερα μέρη της εφαρμογής μεταφέροντας την ευθύνη για τα υπόλοιπα μέρη από τον προγραμματιστή προς το framework. To Laravel επιλέχθηκε ως το framework για την εργασία λόγω της μεγάλης δημοφιλίας του. Επιταχύνει την ανάπτυξη κώδικα, έχει υπερ-αναλυτική τεκμηρίωση, επιτρέπει τη δημιουργία δομημένου και αντικειμενοστραφούς κώδικα, ενώ μεγάλη πληθώρα από κλασικά κομμάτια κώδικα, όπως η διαχείριση των αιτημάτων από και προς το server και η ταυτοποίηση των χρηστών είναι ήδη υλοποιημένα και δεν απαιτείται η δημιουργία του τροχού εκ νέου.

Το Laravel σαν εφαρμογή αποτελεί ένα σύγχρονο web PHP framework ανοιχτού κώδικα (open source) και χρησιμοποιείται για την κατασκευή διαδικτυακών εφαρμογών. Η αρχιτεκτονική του βασίζεται στο MVC πρότυπο. Το MVC σύμφωνα με τον Hardik Dangar (2013) είναι ένα αρχιτεκτονικό πρότυπο με κύριο χαρακτηριστικό το διαχωρισμός λογικής κατά την γραφή και τοποθέτηση κώδικα. Ουσιαστικά ο κώδικας που γράφετε διαχωρίζεται βάση των τριών οντοτήτων που ακολουθούν. Τα Models, είναι το στρώμα που μεσολαβεί ανάμεσα στο χρήστη και στα δεδομένα, τη βάση δεδομένων. Τα Views είναι η οπτική απεικόνιση της εφαρμογής, το στρώμα στο οποίο ανήκουν αρχεία τα οποία εμφανίζονται στο φυλλομετρητή και παράγουν το εικαστικό αποτέλεσμα. Οι Controllers είναι το στρώμα σύνδεσης μεταξύ Models και Views. Πρωταρχικός ρόλος τους είναι χρήση των request που έρχονται από τα Views και η μεταφορά δεδομένων στα Models και αντίστροφα.

Το πλαίσιο εργασίας παρέχει στους χρήστες την δυνατότητα να υλοποιούν εύκολα και γρήγορα ολοκληρωμένες εφαρμογές χρησιμοποιώντας τις βιβλιοθήκες και τα έτοιμα εργαλεία που διαθέτει καθώς και να προσθέτει νέα ανάλογα με τις απαιτήσεις της εφαρμογής. Παρέχει ασφάλεια σε κακόβουλες επιθέσεις SQL injection και Cross Site Scripting (XSS). Επιθέσεις SQL injection είναι οι εντολές SQL που εισάγονται σε πεδία εισόδου στον ιστότοπο μας με σκοπό την κακόβουλη ανάκτηση, αλλαγή, διαγραφή δεδομένων από την βάση. Οι XSS επιθέσεις είναι ο κώδικας που εισάγεται στα πεδία διαδικτυακής φόρμας μιας ιστοσελίδας μέσω του οποίου οι κακόβουλοι χρήστες μπορούν να έχουν πρόσβαση στα cookies, session και άλλα απόρρητα δεδομένα στο φυλλομετρητής (browser) του χρήστη.

## 2.12 Δυνατότητες του Laravel

Το Laravel αποτελεί ένα framework διαδικτυακών εφαρμογών το οποίο προσπαθεί να διευκολύνει τη διαδικασία ανάπτυξης, απλοποιώντας επαναληπτικές διεργασίες που χρησιμοποιούνται στις περισσότερες διαδικτυακές εφαρμογές, περιλαμβάνοντας, αλλά χωρίς να περιορίζεται, το routing, την αυθεντικοποίηση, το caching και τα sessions. Καθώς καταφέρνει να κάνει όλες τις ουσιώδεις εργασίες, από διαδικτύωση και διαχείριση βάσεων δεδομένων μέχρι και παραγωγή HTML περιεχομένου, το Laravel ονομάζεται και full-stack framework,δηλαδή, framework που δουλεύει σε όλες τις στοίβες λειτουργιών. Η συγκεκριμένη κάθετη υλοποίηση (υλοποίηση στοίβας) του περιβάλλοντος ανάπτυξης διαδικτυακών εφαρμογών έχει σκοπό να προσφέρει ένα βελτιωμένο και εύχρηστο μοτίβο ανάπτυξης για τον προγραμματιστή. Αντίθετα με άλλα περιβάλλοντα που υλοποιούν τη λογική στοίβας στο ενσωματωμένο περιβάλλον ανάπτυξής τους, το Laravel είναι μοναδικό με τον τρόπο του, καθώς δίνει προτεραιότητα στις προγραμματιστικές συμβάσεις, σε σχέση με τη διαμόρφωση ρυθμίσεων. Στην πραγματικότητα, αντίθετα με άλλα PHP frameworks τα οποία απαιτούν αρκετή διαμόρφωση πριν την εκκίνηση του κανονικού project, το Laravel χρειάζεται μόνο λίγες γραμμές από PHP κώδικα που να χρειάζονται διόρθωση, πριν να είναι έτοιμο προς χρήση. Αποφεύγοντας εξ' ολοκλήρου, ή χρησιμοποιώντας ένα πολύ μικρό αριθμό από αρχεία ρυθμίσεων, το Laravel επιτυγχάνει να διατηρεί παρόμοια δομή σε κάθε project, πράγμα που τα καθιστά παρόμοια μεταξύ τους, άρα και εύκολα στην αναγνώριση και ανάγνωση. Παρά το γεγονός ότι αυτό μπορεί στην αρχή να φανεί ως περιορισμός στον τρόπο με τον οποίο ο προγραμματιστής θα δομήσει την διαδικτυακή εφαρμογή του, αυτοί οι περιορισμοί στην πραγματικότητα επιτρέπουν πλήρη ελαστικότητα και ευκολία στην κατασκευή εφαρμογών. Στην ουσία, όλα τα νέα projects στο Laravel είναι εξοπλισμένα με ένα πλήρες δομημένο δίκτυο από φακέλους, καθώς και πολλά “placeholder” αρχεία. Η συγκεκριμένη δομή επιτρέπει την γρήγορη εκκίνηση, ανάπτυξη και προτυποποίηση ιδεών και κατά συνέπεια της πραγματικής προγραμματιστικής διαδικασίας. Παρόλα αυτά, αυτή η δομή είναι απόλυτα παραμετροποιήσιμη.

## 2.13 Composer

Ένα άλλο χαρακτηριστικό που κάνει το Laravel να ξεχωρίζει από τα άλλα frameworks, είναι ότι είναι προετοιμασμένο για το Composer. Στην πραγματικότητα, το Laravel το ίδιο είναι ένα μίγμα διαφόρων συστατικών του Composer, πράγμα το οποίο βοηθά στην πολύ σημαντική ανάγκη της διαλειτουργικότητας ανάμεσα στο Composer και στο Laravel. Το Composer είναι ένα εργαλείο διαχείρισης των εξαρτημένων πακέτων για την PHP. Ουσιαστικά, ο κύριος ρόλος του Composer στο Laravel framework είναι ότι διαχειρίζεται την εξάρτηση των εξαρτήσεων των project. Για παράδειγμα, εάν μία βιβλιοθήκη που χρησιμοποιείται σε ένα project εξαρτάται από τρεις άλλες βιβλιοθήκες, και υπάρχει η ανάγκη αναβαθμίσεις των τριών αυτών βιβλιοθηκών, τότε δεν απαιτείται η εύρεση και αναβάθμισή τους. Καθίσταται δυνατό να ανανεωθούν και οι τέσσερεις βιβλιοθήκες μέσω μίας μόνο εντολής, η οποία είναι “$ composer update”. Το Composer έχει τη δυνατότητα να διαχειριστεί μία εξάρτηση μέχρι και στο επίπεδο “ν”, το οποίο σημαίνει πως όλες οι εξαρτήσεις ενός project μπορούν να διαχειριστούν με ένα μόνο εργαλείο, πράγμα το οποίο είναι πολύ βολικό σαν διαθέσιμη επιλογή όταν ο προγραμματιστής έχει να κάνει με ένα μεγάλο σύνολο βιβλιοθηκών. Άλλο ένα πλεονέκτημα της χρήσης του Composer είναι ότι παράγει και χειρίζετε ένα αυτοδιαχειριζόμενο αρχείο στη ρίζα του φακέλου του project του προγραμματιστή, και το οποίο διαθέτει όλες τις εξαρτήσεις του project οι οποίες φορτώνουν αυτόματα τις κλάσεις, όταν είναι ενσωματωμένες σε ένα PHP script. Αυτό αφαιρεί από τον προγραμματιστή την υποχρέωση του να θυμάται τα μονοπάτια όλων των εξαρτήσεων και να τα περιλαμβάνει σε κάθε αρχείο του project. Το μόνο που χρειάζεται να κάνει είναι να περιλαμβάνει το αρχείο αυτόματης φόρτωσης που παρέχεται από το Composer.

Στην τεχνική του μορφή, το Composer είναι εγκατεστημένο ως PHP εκτελέσιμου αρχείου το οποίο είναι προστεθημένο στη μεταβλητή PATH του περιβάλλοντος του μηχανήματος του χρήστη. Η μεταβλητή PATH είναι ουσιαστικά μία λίστα με τις τοποθεσίες που διασχίζονται, όταν μία εντολή εκτελείται στο τερματικό. Δηλαδή, αντί να τρέχει ο προγραμματιστής την εντολή μέσω του απόλυτου μονοπατιού της, μπορεί να περιλαμβάνει τις απόλυτες διαδρομές των εντολών στη μεταβλητή PATH, και απλώς να εκτελεί μόνο το όνομα της μεταβλητής. Όταν το Composer είναι σωστά εγκατεστημένο, ο προγραμματιστής μπορεί να το εκτελέσει μέσα από τη γραμμή εντολών από οποιοδήποτε μέρος στο σύστημα αρχείων, χρησιμοποιώντας την εντολή “$composer”. Το project και οι εξαρτήσεις του είναι ορισμένα μέσα σε ένα αρχείο JSON, το οποίο ονομάζεται composer.json. Ουσιαστικά το Composer αποτελεί εξελεκτικό βήμα για την κοινότητα της PHP, καθώς υπάρχουν χιλιάδες πακέτα λογισμικού επέκτασης τα οποία είναι ήδη δοκιμασμένα και διαθέσιμα στο αρχείο του πακέτου Composer. Το Laravel framework έχει σχεδιαστεί με τέτοιο τρόπο ώστε να ενσωματώνει εύκολα πακέτα του Composer. Το μόνο που χρειάζεται το Composer είναι να διαβάσει τα περιεχόμενα του composer.json αρχείου και να συνδεθεί στον αντίστοιχο διαχειριστή πακέτων, ο οποίος είναι ένα online αποθετήριο πακέτων, ώστε να διευθετήσει όλες τις απαιτήσεις των πακέτων, αναδρομικά, δηλαδή, να δει τις ανάγκες κάθε πακέτου που ζητήθηκε, καθώς και τις ανάγκες των πακέτων για αυτά τα πακέτα, κοκ. Μετέπειτα αυτά τα πακέτα κατεβαίνουν στον τοπικό φάκελο που ονομάζεται vendor/ και η κατάστασή τους καταγράφεται σε ένα αρχείο που ονομάζεται composer.lock. Το Laravel framework, συνδυασμένο με τη δύναμη του Composer, δίνει στον προγραμματιστή πολύ περισσότερη ελευθερία στο να επιλέγει τι είδους πακέτα λογισμικού επιθυμεί να χρησιμοποιήσει στην διαδικτυακή του εφαρμογή. Για παράδειγμα, εάν δεν του αρέσει η προεπιλεγμένη εφαρμογή για τη διαχείριση του Mail που είναι εγκατεστημένη στο Laravel, και η οποία λέγεται Swift Mailer, και επιθυμεί να την αντικαταστήσει με ένα διαφορετικό πακέτο λογισμικού, όπως για παράδειγμα το πακέτο PHPMailer το οποίο είναι συμβατό με το Laravel, τότε μπορεί να το κάνει. Άρα, η αλλαγή ανάμεσα σε δύο πακέτα αποτελεί πολύ εύκολο εγχείρημα. Ο προγραμματιστής μπορεί να αντικαταστήσει πακέτα κώδικα όποτε το επιθυμεί και με ευκολία, όταν υπάρχει η ανάγκη, μέσω του Composer και με των κατάλληλων, γρήγορων και απλών ρυθμίσεων του Laravel.

## 2.14 Model, View, Controller

Το Laravel Framework είναι βασισμένο στο αρχιτεκτονικό πρότυπο MVC (Model - View - Controller) καθώς τα Models, τα Views και οι Controllers βρίσκονται σε ξεχωριστά αρχεία και σε ξεχωριστούς καταλόγους. Στο πρότυπο MVC διαχωρίζεται η λογική της εφαρμογής από την λογική της παρουσίασης που συνδέεται με ένα γραφικό περιβάλλον χρήστη (Graphical User Interface). Το Model είναι μόνιμο χαρακτηριστικό και αποθηκεύεται εκτός της εφαρμογής, συνήθως σε μια Βάση Δεδομένων. Στο Model κατασκευάζουμε πολύπλοκους αλγόριθμους, προγραμματίζουμε σύνθετες πράξεις και πραγματοποιούμε διάφορες ενέργειες σε Βάσεις Δεδομένων (ενημέρωση, εισαγωγή, κλπ..). Το συστατικό στο οποίο ασχολούμαστε με τον τρόπο παρουσίασης των πληροφοριών στο χρήστη είναι το View. Το επίπεδο του view είναι υπεύθυνο για τη δημιουργία ενός περιβάλλοντος χρήστη (User Interface) βασισμένου στα δεδομένα του Model. Το συστατικό το οποίο βρίσκεται μεταξύ του Model και του View και τα συντονίζει είναι ο Controller. Ο Controller είναι υπεύθυνος για την επεξεργασία των αιτημάτων, για το ποια ενέργεια να πραγματοποιήσει. Για παράδειγμα, όταν δεχθεί ένα αίτημα για το οποίο απαιτούνται να γίνουν σύνθετες διεργασίες επικοινωνεί αρχικά με το Model και στη συνέχεια προωθεί το αποτέλεσμα στο View για να το εμφανίσει. Παρακάτω θα παρουσιάσουμε τη λειτουργία ενός αιτήματος που υποβάλει ο χρήστης. Αρχικά, ο Browser στέλνει ένα αίτημα, το οποίο λαμβάνει ένας web server και το μεταφέρει στη Routing Engine του Laravel. Στη συνέχεια, ο router του Laravel λαμβάνει το αίτημα και το προωθεί στον κατάλληλο Controller. Μόλις ο Controller λάβει το αίτημα από το Laravel Router, σε ορισμένες περιπτώσεις απαντάει αμέσως με ένα view, το οποίο είναι ένα πρότυπο που μετατρέπεται σε HTML και αποστέλλεται πίσω στον Browser. Τις περισσότερες όμως φορές, ο Controller πρώτα αλληλοεπιδρά με ένα στοιχείο της web εφαρμογής που είναι υπεύθυνο για την επικοινωνία με τη Βάση Δεδομένων (model) και στη συνέχεια στέλνει ένα view που επιστρέφει πίσω στον Browser ως ολοκληρωμένη Web Σελίδα (HTML, CSS, εικόνες). Η δομή αυτή επιτρέπει στον προγραμματιστή να δημιουργεί εφαρμογές με ευκολία και με ταχύτητα, καθώς ο κώδικας είναι οργανωμένος και κατανοητός.

## 2.15 Model

Ένα model αποτελεί την μέθοδο μέσω της οποίας ο προγραμματιστής μπορεί να διαχειριστεί τα δεδομένα της βάσης. Αποτελείται από ένα επίπεδο που βρίσκεται ανάμεσα στα δεδομένα και την εφαρμογή. Τα ίδια τα δεδομένα μπορούν να αποθηκευτούν σε διάφορους τύπους βάσεων δεδομένων, όπως για παράδειγμα MySQL, PostgresSQL, ή ακόμη και σε απλή XML ή και αρχεία Excel. Η κύρια εργασία των μοντέλων η μετατροπή των δεδομένων των πινάκων της βάσης σε αντικείμενα με ιδιότητες και η επιτροπή της ανάγνωσης και γενικά της διαχείρισης αυτών των αντικειμένων από τα controllers.

## 2.16 Views

Τα views είναι η οπτική αναπαράσταση της διαδικτυακής εφαρμογής (επίπεδο παρουσίασης / presentation layer) και είναι υπεύθυνα για την παρουσίαση των δεδομένων που ο controller έλαβε από το model (business logic). Μπορούν εύκολα να κατασκευαστούν χρησιμοποιώντας το Blade, την γλώσσα κατασκευής των templates που πηγαίνει μαζί με το Laravel, ή με κώδικα php. Το Blade περιλαμβάνει κληρονομικότητα στα templates και τομείς (sections). Όταν το Laravel εμφανίζει αυτά τα Views, πρώτα εξετάζει τον τύπο του αρχείου, και, εξαρτάται από το εάν καταλήγει σε “.blade.php” ή απλά σε “.php”, αποφασίζει εάν θα το αντιμετωπίσει το view ως template του Blade ή όχι.

## 2.17 Controller

Η βασική λειτουργικότητα του Controller (ελεγκτή) είναι να διαχειρίζεται τα αιτήματα και να περνάει δεδομένα στο Model και στα Views. Άρα ένας Controller μπορεί να θεωρηθεί ως ο συνδετικός κρίκος ανάμεσα στο Model και τα Views. Ο προγραμματιστής έχει την επιλογή να γράψει τη δικιά του επιχειρηματική λογική, είτε στους Routers (δρομολογητές) είτε στους Controllers. Οι Routers μπορούν να είναι χρήσιμοι όταν η διαδικτυακή εφαρμογή είναι μικρή, σε γρήγορα αναπτύσσιμες και στατικές σελίδες. Άρα το να γράφει κανείς Controllers για κάθε μοναδική σελίδα της διαδικτυακής εφαρμογής, δεν είναι απαραίτητο. 6.3 Ανάπτυξη templates με το Bootstrap Το Bootstrap είναι αναμφισβήτητα το πιο δημοφιλές frontend framework ανάπτυξης ιστοσελίδων της βιομηχανίας. Προσφέρει μια πλήρη σειρά φιλικών προς το χρήστη, cross-platform και δοκιμασμένα κομμάτια κώδικα για συχνώς επαναλαμβανόμενα μοτίβα και συμβάσεις στο σχεδιασμό διεπαφών χρήστη. Το Bootstrap επιταχύνει σημαντικά την διαδικασία της οικοδόμησης μιας front-end διαδικτυακής διεπαφής λόγω των έτοιμων, ελεγμένων διεξοδικά μείγμα των σημάνσεων της HTML, CSS φύλλων, και συμπεριφοράς της Javascript. Με αυτά τα αναγκαία θεμέλια να ρυθμίζονται εύκολα και γρήγορα, η διεπαφή μπορεί εύκολα να δέχεται τροποποίηση. Υπάρχουν πολλοί τρόποι για να κατέβει το Bootstrap, αλλά δεν είναι όλοι αυτοί οι τρόποι λήψης του Bootstrap ισάξιοι. Παρόλα αυτά, μπορεί κανείς να χρησιμοποιήσει το Initializr το οποίο παράγει templates βασισμένα στην HTML5 Boilerplate, επιτρέποντας τον developer να επιλέξει ποια συστατικά αυτός ή αυτή επιθυμεί από αυτό. Κατεβάζοντας τα συστατικά που επιθυμεί, μπορεί να πάρει την παρακάτω δομή φακέλων και αρχείων, ομαδοποιώντας λογικά τα κοινά assets και παρέχοντας compiled και ελαχιστοποιημένες εκδόσεις των αρχείων αυτών.

## 2.18 Τεχνικές υλοποίησης και συνδυασμός όλων των προαναφερθέντων τεχνολογιών

Στο πλαίσιο της παρούσας εργασίας θα γίνει η μελέτη, η ανάλυση απαιτήσεων, ο σχεδιασμός και η έναρξη μέρους της υλοποίησης μίας εφαρμογής Ιστού, η οποία θα απευθύνεται σε τελικούς χρήστες. Το μοντέλο HTTP που επιλέχθηκε είναι το πλέον διαδεδομένο server-client, όπου η εφαρμογή εκτελείται (runs) από έναν διακομιστή/εξυπηρετητή (server) και προβάλλεται από ένα πρόγραμμα πελάτη (client) μέσω του περιηγητή Ιστού (browser) του χρήστη. Ο διακομιστής που θα χρησιμοποιηθεί έχει το όνομα Apache, το οποίο διαθέτει ένα σύνολο οδηγιών αναφορικά με την επεξεργασία και διεκπεραίωση των αιτημάτων (requests) των clients από και προς το διακομιστή. Το λειτουργικό σύστημα (Operating System) που θα «στεγάζεται» ο server είναι το Linux, το πλέον δημοφιλές σύστημα για φιλοξενία ιστοσελίδων όπως και δυναμικότερων εφαρμογών.

Η εφαρμογή που θα υλοποιηθεί χαρακτηρίζεται ως δυναμική ιστοσελίδα, με την έννοια ότι τα δεδομένα που προβάλλονται στους χρήστες προέρχονται από τη βάση δεδομένων που φιλοξενείται στο server και αυτή είναι η ουσιώδης διαφορά με τις στατικές ιστοσελίδες, δηλαδή η προέλευση των δεδομένων και η ενημέρωση αυτών μέσα από τη βάση δεδομένων. Οι δυναμικές ιστοσελίδες απαιτούν για την εκτέλεσή τους ένα μεγάλο πλήθος τεχνολογιών, οι οποίες θεωρούνται σήμερα απαραίτητες για την κάλυψη διαφόρων αναγκών, όπως για παράδειγμα η ταχύτητα φόρτωσης της εφαρμογής, η υποστήριξη για κινητές συσκευές και η φιλικότητα αλλά και η συνολική εμπειρία που λαμβάνει ο χρήστης.

Οι δυναμικές εφαρμογές απαιτούν τη χρήση γλώσσας διακομιστή, όπως τις Ruby, PHP, JSP κλπ. Η PHP επιλέχθηκε για την εφαρμογή, λόγω της ταχύτητας υλοποίησης, της ευκολίας που προσφέρει, των δυνατότητων που παρέχει και κυρίως λόγω της ευρείας διάδοσής της, αφού η συντριπτική πλειοψηφία των web εφαρμογών έχουν υλοποιηθεί με την εν λόγω γλώσσα. Οι γλώσσες διακομιστή ερμηνεύονται και δρομολογούνται από την τεχνολογία Apache, όπου η τελευταία αναλαμβάνει να εκτελέσει στη μηχανή τα αιτήματα των πελατών.

Η αρχιτεκτονική του έργου που θα δημιουργηθεί ακολουθεί το μοντέλο MVC (Model View Controller) και αποτελεί μια στρατηγική απόφαση για τη φύση αυτού, αφού η εφαρμογή θα ακολουθεί τη δομή του συγκεκριμένου μοντέλου. Το MVC θεωρείται μία κλασική και αποδοτική μέθοδος οργάνωσης του έργου (project) και των μεμονωμένων αρχείων κώδικα και ο πρωταρχικός σκοπός της είναι το όφελος προς τον προγραμματιστή αναφορικά με την περιπλοκότητα και τη μελλοντική ενημέρωση (Continuous Integration) της εφαρμογής. Τα αρχικά MVC προέρχονται από τις λέξεις: Model, View, Controller, που αποτελούν τα 3 κύρια και διακριτά μέρη της εφαρμογής. Το model αφορά την οργάνωση της λογικής του έργου και αποτελεί τον ενδιάμεσο που επιτρέπει τη μεταφορά των δεδομένων από τον controller στο view και αντίστροφα, από τη βάση δεδομένων στον controller. Εδώ είναι ο χώρος όπου γράφονται τα ερωτήματα προς τη βάση δεδομένων. Ο controller έχει άμεση επικοινωνία με το model και αναλαμβάνει να επεξεργαστεί και να εκτελέσει τα δεδομένα επιστρέφονται από αυτό. Η τελική του εργασία είναι να παραδώσει τα δεδομένα στο view τα οποία θα προβληθούν στον client. Το view είναι ο τελικός κώδικας που βλέπει ο χρήστης και αλληλοεπιδρά μόνο με αυτό. Αυτή η λογική επιτρέπει τη μεγαλύτερη ασφάλεια των εφαρμογών, καθώς ο χρήστης έχει πρόσβαση μόνα στα δεδομένα που αποφασίζεται να εμφανιστούν σε αυτόν.

Τα δεδομένα της εφαρμογής «κατοικούν» σε μια βάση δεδομένων, με τη MySQL να έχει επιλεχθεί γι’ αυτόν το σκοπό. Η MySQL είναι ένα σχεσιακό σύστημα βάσεων δεδομένων, όπου αυτά είναι οργανωμένα σε γραμμές που βρίσκονται μέσα σε πίνακες, οι οποίοι έχουν δυνατή ή χαλαρή συσχέτιση με άλλους πίνακες. Για παράδειγμα, οι διαδρομές που θα προβάλλονται στους επισκέπτες της ιστοσελίδας έχουν αποθηκευτεί στον πίνακα: app\_route που ορίζονται κατ’ ελάχιστον από τις στήλες (πεδία/columns) του ονόματος και της περιγραφής. Οι χρήστες και οι κωδικοί αυτών βρίσκονται αποθηκευμένοι στον πίνακα των χρηστών, τόσο από πλευρά οργάνωσης των δεδομένων όσο και από πλευρά της ασφάλειας της πληροφορίας. Με λίγα λόγια οι πιο σημαντικές πληροφορίες της εφαρμογής οργανώνονται σε πίνακες αναφορικά με το είδος και τον τύπο της πληροφορίας. Τα δεδομένα φορτώνονται, εισάγονται, ενημερώνονται και διαγράφονται στη βάση μέσω της χρήσης και εκτέλεσης ερωτημάτων SQL που γίνονται προς αυτήν.

Η γλώσσα εμφάνισης των δεδομένων της εφαρμογής είναι η γλώσσα σήμανσης (markup) HTML με τα μεμονωμένα στοιχεία της εφαρμογής να οργανώνονται σε κόμβους (nodes) XML. Η HTML περιλαμβάνει ορισμένα στοιχεία αλληλεπίδρασης με το χρήστη, όπως τις φόρμες και εμφάνισης των δεδομένων όπως τους πίνακες. Αυτά τα στοιχεία ναι μεν χρησιμοποιούνται από την πλειοψηφία των εφαρμογών, αλλά η απλή προβολή τους θεωρείται ξεπερασμένη, γι’ αυτό και χρησιμοποιούνται επιπλέον βιβλιοθήκες, όπως το Bootstrap, το οποίο και επιλέχθηκε για την υλοποίηση της εφαρμογής. To Bootstrap είναι ένα εργαλείο (framework) για τη δημιουργία έργων προγραμματισμού στο Web. Απαιτεί πολύ λίγο κώδικα από τον προγραμματιστή, καθώς αναλαμβάνει να κάνει τη δύσκολη σχεδιαστική δουλειά γι’ αυτόν. Η κύρια εργασία είναι η βελτιστοποίηση της εμφάνισης των στοιχείων HTML με αποτέλεσμα να κάνει πιο άνετη και φιλική την ιστοσελίδα προς το χρήστη. Τα φύλλα στυλ CSS υπάρχουν πλέον σε όλες τις ιστοσελίδες και ο κύριος σκοπός τους είναι η σχεδίαση και μορφοποίηση των δεδομένων σε κατάλληλες ενότητες και τελικά αποδίδουν μια ξεχωριστή προσωπικότητα σε κάθε ιστότοπο και τον καθιστούν μοναδικό και διαφορετικό σε σχέση με άλλους. Μεγάλο μέρος της βιβλιοθήκης έχει γραφτεί για την υποστήριξη των κινητών συσκευών, οι οποίες αποτελούν σήμερα σημαντικό μέρος της δραστηριότητας στον Ιστό.

Οι σύγχρονες εφαρμογές είναι απαιτητικές από πλευράς φιλικότητας προς το χρήστη και διαχείρισης της πληροφορίας, γι’ αυτό και χρησιμοποιείται η γλώσσα client, Javascript. Η εν λόγω γλώσσα επιτρέπει την εκτέλεση σεναρίων στον υπολογιστή του χρήστη. Το κύριο έργο τους είναι η οργάνωση της πληροφορίας και της διαδρομής του χρήστη όσον την περιήγηση του χρήστη στην ιστοσελίδα. Αναλαμβάνουν επίσης την αλληλεπίδραση με το χρήστη και συνολικά προσφέρουν μια βελτιωμένη εμπειρία χρήσης.

## 2.19 Visual Studio Code

Το Visual Studio Code είναι ένα πρόγραμμα επεξεργασίας κώδικα που διατίθεται από τη Microsoft για τα λειτουργικά συστήματα Windows, Linux και macOS. Χρησιμοποιείται από προγραμματιστές από όλον τον κόσμο για τη γραφή κώδικα σε μια μεγάλη πληθώρα γλωσσών προγραμματισμού. Αυτό το επιτυγχάνει μέσω της επισήμανσης σύνταξης (syntax highlighting), έξυπνης συμπλήρωσης κώδικα (intelligent code completion) και refactoring κώδικα μέσω της μετονομασίας κλάσεων, μεθόδων, ιδιοτήτων και μεταβλητών. Επιπλέον έχει υποστήριξη για αποσφαλμάτωση κώδικα (debugging), ενώ επιτρέπει στους χρήστες να το προσαρμόσουν αλλά και να το τροποποιήσουν μέσω των επεκτάσεων (extensions) που διαθέτει στις δικές τους ανάγκες. Με λίγα λόγια πρόκειται για ένα δυναμικό με άνετο και φιλικό γραφικό περιβάλλον πρόγραμμα διαθέτοντας πολλές δυνατότητες στους χρήστες τόσο από πλευράς τροποποίησης της εμφάνισης και των λειτουργιών του όσο και από πλευράς επεκτασιμότητας και μάλιστα διατίθεται δωρεάν. Στην περίπτωση της πτυχιακής εργασίας χρησιμοποιήθηκε για τη γραφή κώδικα όλων των απαραίτητων αρχείων της εφαρμογής και συγκεκριμένα για την κωδικοποίηση HTML, CSS, Javascript και PHP.

# ΚΕΦΑΛΑΙΟ 3 – Τεχνικά δομικά στοιχεία του έργου

## 3.1 Διαδικασία ανάπτυξης

Για την υλοποίηση της εφαρμογής κρίθηκε απαραίτητη η εγκατάσταση ενός τοπικού web εξυπηρετητή ο οποίος θα φιλοξενήσει το σύστημα. Ιδανικότερη λύση θεωρήθηκε τον Apache Web Server, ο οποίος διατίθεται δωρεάν. Έγινε εγκατάσταση του XAMPP, το οποίο είναι μια σουίτα προγραμμάτων ελεύθερου ανοικτού κώδικα λογισμικού και ανεξαρτήτου πλατφόρμας, το οποίο περιλαμβάνει το εξυπηρετητή ιστοσελίδων Apache και ένα διερμηνέα για κώδικα γραμμένο σε γλώσσες προγραμματισμού PHP. Μετά την εγκατάσταση του XAMPP, έγινε η εγκατάσταση του Laravel μέσω γραμμής εντολών (PowerShell) χρησιμοποιώντας τις εντολές composer (composer create-project --prefer-dist laravel/laravel) στον φάκελο C:/xampp/htdocs. O φάκελος αυτός αναγνωρίζεται από το Xampp ως localhost για την εκτέλεση της εφαρμογής. Σε αυτό το σημείο ξεκίνησε η παραγωγή κώδικα. . Το Laravel, όπως και τα περισσότερα PHP Framework διαθέτουν το αρχείο .env που βρίσκεται στον αρχικό φάκελο του έργου, στο οποίο ο προγραμματιστής μπορεί να ορίσει ευαίσθητα δεδομένα, τα οποία είναι προσβάσιμα μόνο από το ίδιο το framework. Ορίστηκαν τα στοιχεία σύνδεσης με τη βάση δεδομένων στο αρχείο .env στο root φάκελο. Μέσα στο αρχείο μπορούν να αποθηκεύονται άλλα στοιχεία, τα οποία είναι καθολικές μεταβλητές, όπως για παράδειγμα η σύνδεση σε mail server ή σε άλλες υπηρεσίες και APIs (Application Programming Interfaces). Στην παρακάτω εικόνα βλέπουμε ένα μέρος του PHP project όπως διαμορφώθηκε κατά την ολοκλήρωσή του:

Text

Description automatically generated

## 3.2 Περιγραφή του σχήματος της βάσης δεδομένων

Η βάση δεδομένων ονοματίστηκε ως eshop και αποτελείται από 10 πίνακες. Το σχήμα της δημιουργήθηκε με το εργαλείο Navicat Premium, το οποίο είναι ένα MySQL client και επιτρέπει την εύκολη δημιουργία πινάκων και στηλών, καθώς το ενσωματωμένο εργαλείο PHPMyAdmin που έρχεται μαζί με την εγκατάσταση του Xampp είναι δύσχρηστο και μη φιλικό προς το χρήστη. Παρακάτω βλέπουμε το σχεσιακό διάγραμμα της βάσης και τις συσχετίσεις μεταξύ των πινάκων:

Diagram

Description automatically generated

Ακολουθεί η ανάλυση των πινάκων:

**Πίνακας users (στοιχεία χρηστών)**

Στήλη id: Η στήλη id υπάρχει σε όλους τους πίνακες στη βάση και αποτελεί τον αύξοντα αριθμό κάθε εγγραφής και συμπληρώνεται αυτομάτως από τη MySQL με κάθε προσθήκη εγγραφής, δηλαδή έχει επιλογή αύξουσας αρίθμησης με κάθε εγγραφή νεότερη έχει μεγαλύτερο id από την αμέσως προηγούμενής της. Είναι ακέραιος 11 ψηφίων και αποτελεί τη μοναδική ταυτότητα κάθε εγγραφής του πίνακα, δηλαδή το αναγνωριστικό της.

Στήλη name: Είναι τύπος varchar 255 χαρακτήρων, δηλαδή συμβολοσειρά κι εδώ αποθηκεύουμε το όνομα χρήστη (username) που αυτός εισάγει κατά την εγγραφή του.

Στήλη email: varchar 255 χαρακτήρων, το email του χρήστη.

Στήλη fname: varchar 255 χαρακτήρων, το όνομα του χρήστη.

Στήλη lname: varchar 255 χαρακτήρων, το επώνυμο του χρήστη.

Στήλη address: varchar 255 χαρακτήρων, η διεύθυνση του χρήστη.

Στήλη password: varchar 255 χαρακτήρων, ο κωδικός του χρήστη, ο οποίος είναι κατακερματισμένος για λόγους ασφαλείας για την αποφυγή κακόβουλων επιθέσεων. Ένας κακόβουλος χρήστης εάν αποκτήσει με κάποιο τρόπο πρόσβαση στη βάση δε θα μπορεί να μαντέψει τον κωδικό των χρηστών και αυτό θα τον αποτρέψει από περαιτέρω κακόβουλες ενέργειες.

Στήλη created\_at: Είναι τύπου timestamp, δηλαδή χρονοσφραγίδα που συμπληρώνεται αυτομάτως από τη MySQL κατά την εισαγωγή εγγραφής, δηλαδή έχει ως προεπιλεγμένη τιμή την SQL εντολή: current\_timestamp().

Στήλη updated\_at: Είναι τύπου timestamp και συμπληρώνεται από την εφαρμογή όταν ενημερώνεται η εγγραφή στον πίνακα.

**Πίνακας password\_resets (επαναφορά κωδικών χρηστών)**

Σε αυτόν τον πίνακα αποθηκεύονται οι ενέργειες επαναφοράς των κωδικών των χρηστών.

Στήλη email: varchar 255 χαρακτήρων, το email του χρήστη.

Στήλη token: varchar 255 χαρακτήρων, μια μοναδική συμβολοσειρά που περιέχεται στο URL που αποστέλλεται στο email του χρήστη για την επαναφορά του κωδικού του.

**Πίνακας categories**

Σε αυτόν τον πίνακα αποθηκεύονται οι κατηγορίες στις οποίες ανήκουν τα προιόντα.

Στήλη id: Βλ. πίνακα users

Στήλη name: varchar 255 χαρακτήρων, η ονομασία της κατηγορίας

Στήλη description: varchar 255 χαρακτήρων, η περιγραφή της κατηγορίας

Στήλη image: varchar 255 χαρακτήρων, η εικόνα της κατηγορίας. Εδώ αποθηκεύεται το όνομα του αρχείου της εικόνας.

Στήλη created\_at: Βλ. πίνακα users

Στήλη updated \_at: Βλ. πίνακα users

**Πίνακας migrations**

Πρόκειται για πίνακα αναγκαίος για το Laravel και χρησιμοποιείται για τη μεταφορά των μοντέλων από PHP κώδικα στο Laravel project προς τη βάση δεδομένων με την εντολή: php artisan migrate

**Πίνακας failed\_jobs**

Πρόκειται για πίνακα αναγκαίος για το Laravel και χρησιμοποιείται για την εκτέλεση cron jobs (επαναλαμβανόμενων ρουτινών). Δε χρησιμοποιούνται στην παρούσα εφαρμογή cron jobs.

**Πίνακας products**

Εδώ αποθηκεύονται οι πληροφορίες για τα προιόντα.

Στήλη id: Τύπος bigint, δηλαδή πολύ μεγάλος ακέραιος. Βλ. πίνακας users

Στήλη cate\_id: Τύπος bigint, το id της κατηγορίας στην οποία ανήκει το προιόν στον πίνακα categories.

Στήλη name: Τύπος varchar. Η ονομασία του προιόντος.

Στήλη small\_description: Τύπος mediumtext. Δηλαδή αρκετά μεγάλο κείμενο για την περιγραφή του προιόντος.

Στήλη price: Τύπος varchar. Η τιμή του προιόντος. Παρόλο που είναι συμβολοσειρά, το Laravel μετατρέπει το πεδίο σε δεκαδικό αριθμό αυτόματα για εμάς.

Στήλη qty: Τύπος varchar. Η ποσότητα του προιόντος. Παρόλο που είναι συμβολοσειρά, το Laravel μετατρέπει το πεδίο σε ακέραιο αριθμό αυτόματα για εμάς.

Στήλη image: Τύπος varchar. Η εικόνα του προιόντος. Παρόμοια στήλη με την ομώνυμη στον πίνακα κατηγοριών.

Στήλη sku: Τύπος varchar. Το SKU του προιόντος. Το μοναδικό αναγνωριστικό προιόντος.

Στήλη created\_at: Βλ. πίνακα users

Στήλη updated \_at: Βλ. πίνακα users

**Πίνακας cart**

Εδώ αποθηκεύονται τα καλάθια των χρηστών

Στήλη id: Βλ. πίνακας products.

Στήλη user\_id: Τύπος varchar. Το id του χρήστη και συσχετίζεται με τη στήλη id στον πίνακα users.

Στήλη prod\_qty: Τύπος varchar. Η ποσότητα ενός προιόντος στο καλάθι.

Στήλη prod\_id: Τύπος varchar. Το id του προιόντος και συσχετίζεται με τη στήλη id στον πίνακα products.

Στήλη created\_at: Βλ. πίνακα users

Στήλη updated \_at: Βλ. πίνακα users

**Πίνακας orders**

Εδώ αποθηκεύουμε τις παραγγελίες των πελατών.

Στήλη id: Βλ. πίνακας users.

Στήλη user\_id: Τύπος varchar. Το id του χρήστη και συσχετίζεται με τη στήλη id στον πίνακα users.

Στήλη fname: Βλ. πίνακας users.

Στήλη lname: Βλ. πίνακας users.

Στήλη address: Βλ. πίνακας users.

Στήλη tracking\_no: Τύπος varchar. Ο αριθμός αποστολής της παραγγελίας

Στήλη payment\_type: Τύπος varchar. Ο τρόπος πληρωμής της παραγγελίας

Στήλη total\_price: Τύπος int, ακέραιος. Το συνολικό ποσό της παραγγελίας.

Στήλη created\_at: Βλ. πίνακα users

Στήλη updated \_at: Βλ. πίνακα users

**Πίνακας order\_items**

Εδώ αποθηκεύουμε το κάθε είδος που σχετίζεται με μια παραγγελία.

Στήλη id: Βλ. πίνακας users.

Στήλη order\_id: Τύπος varchar. Το id της παραγγελίας που συσχετίζεται με τη στήλη id στον πίνακα orders.

Στήλη prod\_id: Τύπος varchar. Το id του προιόντος που συσχετίζεται με τη στήλη id στον πίνακα των προιοντων.

Στήλη qty: Τύπος varchar. Η ποσότητα του παραγγελθέντος είδος.

Στήλη price: Τύπος varchar. Η τιμή του είδους την ώρα της παραγγελίας.

Στήλη created\_at: Βλ. πίνακα users

Στήλη updated \_at: Βλ. πίνακα users

# ΚΕΦΑΛΑΙΟ 4 – Τεχνική ανάλυση εφαρμογής

## 4.1 Routes (δρομολόγια αιτημάτων) και middlewares

Το επόμενο βήμα ήταν η ρύθμιση των routes, δηλαδή των δρομολογίων κάθε πόρου, δηλαδή ποιο controller ή ποιο view θα αναλάβει να επεξεργαστεί τη διαδρομή πόρου που αιτείται ο χρήστης. Παρακάτω βλέπουμε το περιεχόμενο του αρχείου routes/web.php:

Route::group(['middleware' => ['web']], function () {

    Route::get('/', [ProductController::class, 'index'])->name('welcome');

    Route::get('view-category/{id}', [ProductController::class, 'viewCategory']);

    Route::get('register', [RegisterController::class, 'viewRegisterForm']);

    Route::post('register', [RegisterController::class, 'registerNewUser']);

    Route::get('login', [SessionsController::class, 'viewLoginForm'])->name('login');

    Route::post('login', [SessionsController::class, 'userLogin']);

    Route::get('logout', [SessionsController::class, 'userLogOut']);

    Route::get("details/{id}", [ProductController::class, 'viewProductDetails']);

    Route::post('add-to-cart', [CartController::class, 'addProduct']);

    Route::get("cart", [CartController::class, 'viewcart']);

    Route::post("update-cart", [CartController::class, 'updatecart']);

    Route::post("remove-cart-item", [CartController::class, 'removeFromCart']);

    Route::get("checkout", [CheckoutController::class, 'index']);

    Route::post("place-order", [CheckoutController::class, 'placeorder']);

    Route::get('myorders', [UserController::class, 'viewMyOrders']);

    Route::get('view-order/{id}', [UserController::class, 'view']);

    Route::get('about-us', [UserController::class, 'viewAboutUs']);

    Route::group(['middleware' => ['auth', 'isAdmin']], function () {

        Route::get('admin/dashboard', [AdminController::class, 'index'])->name('adminIndex');

        Route::get('add-category', [AdminController::class, 'viewCategoryInsertForm']);

        Route::post('insert-category', [AdminController::class, 'insertCategory']);

        Route::get('view-categories', [AdminController::class, 'categoriesIndex']);

        Route::get('edit-category/{id}', [AdminController::class, 'editCategory']);

        Route::post('update-category/{id}', [AdminController::class, 'updateCategory']);

        Route::get('delete-category/{id}', [AdminController::class, 'deleteCategory']);

        Route::get('view-products', [AdminController::class, 'productsIndex']);

        Route::get('add-product', [AdminController::class, 'viewProductInsertForm']);

        Route::post('insert-product', [AdminController::class, 'insertProduct']);

        Route::get('edit-product/{id}', [AdminController::class, 'editProduct']);

        Route::post('update-product/{id}', [AdminController::class, 'updateProduct']);

        Route::get('delete-product/{id}', [AdminController::class, 'deleteProduct']);

    });

});

Όλα τα routes ανήκουν στο middleware web, που αποτελούν το προεπιλεγμένο middleware του Laravel. Δεν είναι απαραίτητο να εισάγουμε τα routes στο web middleware, αλλά είναι καλή πρακτική να εισάγουμε τα routes σε αυτό καθώς προτείνεται η χρήση των middleware για προ-επεξεργασία των αιτημάτων πριν αυτά φθάσουν στο controller. Επίσης, το συγκεκριμένο δίνει πρόσβαση σε μια σειρά από άλλα middlewares, τα οποία είναι σημαντικά αλλά βρίσκονται έξω από την εμβέλεια της εργασίας.

Θα ξεκινήσουμε με την περιγραφή των routes που ανήκουν μόνο στο πρώτο middleware και αφορά όλα τα δημόσια routes στα οποία έχει πρόσβαση ο κάθε χρήστης είτε είναι συνδεδεμένος ή μη, δηλαδή είναι επισκέπτης. Στην εφαρμογή έχουμε 2 ειδών routes, το POST και το GET. Σε ένα GET route, η εφαρμογή θα ανακατευθύνει το χρήστη στον πόρο που αιτείται. Σε ένα POST route, ο χρήστης συνήθως υποβάλλει μια φόρμα και το Laravel θα κατευθύνει την επεξεργασία των δεδομένων της φόρμας στο αίτημα με το οποίο αυτή συσχετίζεται. Το αίτημα αναλαμβάνει να το διεκπεραιώσει ένα controller. Την ονομασία του πόρου τη βλέπουμε μετά τον τύπο του εκάστοτε route. Το πρώτο route είναι ο αρχικός κατάλογος και συμβολίζεται με το slash. Ύστερα σε ένα array παράσχουμε το controller που θα αναλάβει να διαχειριστεί το αίτημα και τη μέθοδο που θα το αναλάβει συγκεκριμένα. Προαιρετικά σε ένα route καλούμε τη μέθοδο name για να δώσουμε ένα ψευδώνυμο (alias) σε ένα route για χρήση σε όλα τα views. Στο επόμενο route έχουμε ένα πόρο που καταλήγει σε ένα named parameter με ονομασία id. Το named parameter περιέχεται σε άγκιστρα. Στη συγκεκριμένη περίπτωση, όταν ο χρήστης πλοηγηθεί σε μια κατηγορία, αυτή έχει ένα id στο URL, το οποίο είναι συνήθως ένας ακέραιος. Αυτός λοιπόν ο ακέραιος είναι το id. Τα υπόλοιπα routes δε χρειάζονται επεξήγηση καθώς ομοιάζουν τα προηγούμενα δύο που περιγράψαμε.

Συνεχίζουμε στα routes τα οποία ανήκουν στα middlewares με ονομασία auth και isAdmin. Το πρώτο middleware ανήκει στον πυρήνα του Laravel και αφορά στην αυθεντικοποίηση του χρήστη. Ο χρήστης λοιπόν πρέπει να είναι συνδεδεμένος για να έχει πρόσβαση στα routes που ανήκουν στο εν λόγω middleware και αυτό είναι κάτι που ελέγχει το Laravel για εμάς. Το isAdmin middleware δημιουργήθηκε για χάρη της εργασίας και τα routes που ανήκουν σε αυτό απαιτείται να έχει προηγηθεί ταυτοποίηση του χρήστη ως διαχειριστής. Τα middlewares τα ορίζουμε στο φάκελο app/Middleware κι εκεί θα βρούμε το AdminMiddleware και παρακάτω βλέπουμε το σώμα της μεθόδου handle, η οποία είναι η αρμόδια μέθοδος σε όλα τα middlewares για να διαχειριστεί ένα αίτημα:

 public function handle(Request $request, Closure $next)

    {

        if (auth()->user()?->email != 'admin@admin.com') {

            abort(403);

        }

        return $next($request);

    }

Καταλαβαίνουμε ότι ένας χρήστης είναι διαχειριστής εάν έχει ταυτοποιηθεί και το email του είναι [admin@admin.com](mailto:admin@admin.com). Εάν δεν ισχύει αυτό, καλούμε τη μέθοδο abort με κωδικό λάθους (HTTP error code) 403, το οποίο σηματοδοτεί την απαγόρευση πρόσβασης και τότε ο browser θα αναλάβει να δείξει σχετικό μήνυμα στο χρήστη. Σε κάθε άλλη περίπτωση, επιτρέπεται η πρόσβαση στο διαχειριστή.

## 4.2 Models

Στο φάκελο app/Models ορίζουμε τα μοντέλα της εφαρμογής και σχετίζονται συνήθως με έναν πίνακα στη βάση. Σε MVC frameworks δε συνηθίζουμε να κάνουμε raw SQL ερωτήματα στη βάση καθώς θεωρείται κακή πρακτική και πιθανό να καταλήξει σε λάθη. Επομένως, έχουμε τα εξής μοντέλα:

1. Cart: Αφορά το καλάθι. Στη μεταβλητή $fillable ορίζουμε τα πεδία στη βάση που επιτρέπεται η συμπλήρωση στη βάση δεδομένων. Όσα πεδία δεν βρίσκονται σε αυτή τη μεταβλητή, τότε δε συμπληρώνονται (με εξαίρεση τα πεδία χρονοσφραγίδων). Σε μοντέλα ορίζουμε συσχετίσεις μεταξύ άλλων πινάκων (χρησιμοποιώντας τη στήλη του εν λόγω πίνακα και τη στήλη στον πίνακα συσχέτισης) και εδώ ορίσαμε τη συσχέτιση products, δηλαδή ένα καλάθι σχετίζεται με ένα προιόν.
2. Category: Αφορά τις κατηγορίες
3. Order: Αφορά τις παραγγελίας. Θα βρούμε τη σχέση orderitems, δηλαδή μια παραγγελία έχει παραγγελθέντα είδη (πίνακας order\_items).
4. OrderItem: Αφορά τα παραγγελθέντα είδη. Θα βρούμε τη σχέση products, καθώς ένα είδος που έχει παραγγελθεί σχετίζεται με ένα προιόν.
5. Product: Αφορά τα προιόντα. Θα βρούμε τη σχέση category, καθώς ένα προιόν σχετίζεται με μια κατηγορία.
6. User: Αφορά τους χρήστες. Θα βρούμε τη μεταβλητή $hidden, που σημαίνει ότι τα πεδία που είναι μέσα στη μεταβλητή, δε φαίνονται σε ερωτήματα που περιλαμβάνουν εγγραφές από χρήστες (είτε άμεσα είτε έμμεσα μέσω κάποιας σχέσης). Στη συγκεκριμένη περίπτωση, το πεδίο του κωδικού δεν είναι επιθυμητό να εμφανίζεται.

## 4.3 Εγγραφή χρήστη

Οι controllers ενός συστήματος στεγάζονται στο φάκελο app/HTTP/Controllers. Το RegisterController αναλαμβάνει την εγγραφή των χρηστών. Η πρώτη μέθοδος με ονομασία viewRegisterForm θα αναλάβει να εμφανίσει το view με όνομα registerForm. Όλα τα views ανήκουν στο φάκελο resources/views. Το εν λόγω view έχει HTML περιεχόμενο και περιλαμβάνει PHP κώδικα σε Blade γλώσσα, η οποία είναι μια μηχανή προτύπων και αναλαμβάνει να ερμηνεύσει τον PHP κώδικα όταν υπάρχει σε αρχεία με κατάληψη blade.php. Μέσα σε views μπορούμε να κάνουμε εισάγουμε άλλα views με την εντολή @extends ορίζοντας το φάκελο που βρίσκεται το view και το όνομα του αρχείου. Εδώ εισάγουμε το αρχείο layouts/master που περιέχει βασικό κώδικα HTML και γίνεται εισαγωγή όλων των στατικών αρχείων της εφαρμογής, δηλαδή των CSS και των Javascript αρχείων. Τα αρχεία που κάνουμε εισαγωγή αφορούν τις βιβλιοθήκες Bootstrap, jQuery, custom CSS και Javascript αρχεία και Google fonts. Επίσης θα βρούμε την οδηγία {{ csrf\_token() }}, το οποίο παράγει ένα μοναδικό CSRF token που θα εξηγήσουμε πιο μετά. Επιστρέφουμε στο πρώτο view και βρίσκουμε την οδηγία @section, μέσω της οποία δίνουμε είτε δίνουμε 2 παραμέτρους με την 1η παράμετρο να αφορά το όνομά της και τη 2η την τιμή της και αναλαμβάνει να εμφανίζει την τιμή το view στο οποίο έγινε το extend. Όταν η οδηγία δίνεται χωρίς παράμετρο, τότε λέμε στο Laravel να εμφανίσει το περιεχόμενο που ακολουθεί μετά την οδηγία και τελειώνει με την οδηγία @endsection. Στη περίπτωσή μας, το περιεχόμενο θα αποδοθεί από το master view με την εντολή @yield('content'), δηλαδή συνδέουμε τα 2 view μεταξύ τους χρησιμοποιώντας την ίδια συμβολοσειρά ανάμεσα στα 2.

Επίσης θα βρούμε την οδηγία @error και μια συμβολοσειρά ως παράμετρο. Είναι μια οδηγία που γίνεται ερμηνεία από το Laravel και αναλαμβάνει να ελέγξει εάν υπάρχουν λάθη στη φόρμα. Όταν υπάρχουν, τότε εμφανίζουμε το λάθος (μεταβλητή $message). Οι οδηγίες των λαθών δέχονται ως παράμετρο μια συμβολοσειρά με την οποία σχετίζεται με ένα πεδίο στη φόρμα στο οποίο εντοπίστηκε το λάθος. Σε διπλά άγκιστρα εμφανίζουμε μια μεταβλητή PHP σε Blade αρχεία. Τέλος, θα βρούμε το παρακάτω block κώδικα:

@if ($message = Session::get('success'))

            <div class="alert alert-success">

                <strong>{{ $message }}</strong>

            </div>

        @endif

Θα γίνει έλεγχος εάν ο χρήστης έχει ήδη υποβάλλει τη φόρμα και ο PHP κώδικας επεξεργάστηκε το αίτημα του χρήστη με επιτυχία κι έτσι θα εμφανίσουμε το μήνυμα της επιτυχούς εγγραφής στο χρήστη.

Όταν ο χρήστης υποβάλλει τη φόρμα εγγραφής, καλείται η μέθοδος registerNewUser και αυτό φαίνεται από τα routes που είδαμε νωρίτερα. Στο σώμα της μεθόδου θα βρούμε το ακόλουθο block:

  $attributes = request()->validate(

            [

                'name' => 'required',

                'email' => 'required|unique:users,email',

                'password' => 'required'

            ],

            [

                'email.unique' => 'Το email υπάρχει ήδη',

                'email.required' => 'Το email είναι υποχρεωτικό',

            ]

        );

Ορίζουμε 4 κανόνες που αφορούν το αίτημα της εγγραφής μαζί με τα μηνύματα λάθους για λάθη που σχετίζονται με το email:

1. Το όνομα είναι υποχρεωτικό
2. Το email είναι υποχρεωτικό
3. Ο κωδικός είναι υποχρεωτικός
4. Το email πρέπει να είναι μοναδικό, να μην υπάρχει στον πίνακα των χρηστών

Το Laravel θα ελέγξει για εμάς τους κανόνες που τηρούνται σε κάθε αίτημα εγγραφής και θα εμφανίσει τυχόν λάθη. Θα γίνει κατακερματισμός του κωδικού του χρήστη και θα αποθηκευτεί σε αυτή τη μορφή στη βάση δεδομένων και τελικά ο χρήστης θα ενημερωθεί με μήνυμα επιτυχίας.

## 4.4 Είσοδος χρήστη

Το SessionsController είναι το αρμόδιο controller που αναλαμβάνει την ταυτοποίηση των χρηστών στο σύστημα. Η μέθοδος viewLoginForm θα εμφανίσει τη φόρμα εισόδου στο χρήστη που βρίσκεται στο αρχείο sessions/loginForm. Το συγκεκριμένο view έχει παρόμοια χαρακτηριστικά με το register view στην προηγούμενη ενότητα και δεν υπάρχει κάτι νέο άξιο περιγραφής. Με τη μέθοδο userLogin πραγματοποιείται η είσοδος του χρήστη και ξεκινά με τον ορισμό κανόνων στα πεδία εισόδου όπως φαίνεται στο παρακάτω block:

 $attributes = request()->validate([

            'email' => 'required|exists:users,email',

            'password' => 'required'

        ]);

Το email και o κωδικός είναι υποχρεωτικά πεδία και το 1ο πεδίο πρέπει να υπάρχει στον πίνακα των χρηστών, δηλαδή ο χρήστης πρέπει να έχει ήδη εγγραφεί. Με την εντολή auth()->attempt($attributes) θα γίνει η επαλήθευση των κανόνων και η είσοδος του χρήστη εφόσον βρει ταίριασμα του email και του κωδικού ανάμεσα στα πεδία της φόρμας και στα πεδία της βάσης. Θα παραχθεί νέα συνεδρία χρήστη (cookie) για να μπορέσουμε να γνωρίζουμε σε κάθε σελίδα για ποιον χρήστη αφορά η κάθε συνεδρία. Εάν ο χρήστης είναι διαχειριστής (έχοντας το email [admin@admin.com](mailto:admin@admin.com)) τότε θα τον ανακατευθύνουμε στη σελίδα της διαχείρισης. Διαφορετικά ανακατευθύνεται στο root κατάλογο και δη στην αρχική σελίδα.

Με τη μέθοδο userLogOut γίνεται αποσύνδεση του χρήστη, διαγράφονται οι συνεδρίες του και γίνεται ανακατεύθυνση στην αρχική σελίδα.

## 4.5 Αρχική σελίδα (όλα τα προιόντα)

Η μέθοδος index θα διαχειριστεί το αίτημα στην αρχική σελίδα και θα κάνει ερώτημα στον πίνακα products μέσω του μοντέλου Product και θα επιστραφούν όλα τα προιόντα του ηλεκτρονικού καταστήματος. Θα εμφανιστεί το view με όνομα home δίνοντας ως παραμέτρους τις μεταβλητές search (θα την περιγράψουμε μετά) και products έτσι ώστε να τις γνωρίζει και να εμφανίσει τα περιεχόμενά τους.

Στο view μέσω ενός βρόγχου foreach θα κάνουμε επανάληψη όλα τα προιόντα (μεταβλητή products) και θα εμφανίσουμε τις εξής στήλες:

1. Κατηγορία στην οποία το προιόν
2. Ποσότητα του προιόντος: όταν είναι θετική εμφανίζεται, διαφορετικά εμφανίζεται το λεκτικό Out of Stock.
3. Εικόνα του προιόντος
4. Τιμή του προιόντος
5. SKU του προιόντος
6. Κουμπί προσθήκης στο καλάθι όταν η ποσότητα είναι θετική.
7. Σύνδεσμος προς την κατηγορία του προιόντος
8. Σύνδεσμος προς τη σελίδα του προιόντος

## 4.5.1 Αναζήτηση προϊόντος

Δίνεται η δυνατότητα στο χρήστη να αναζητήσει για προιόντα με βάση το όνομα ή το SKU τους και η υπεύθυνη μέθοδος για αιτήματα αναζήτησης είναι η index όπου θα γίνει έλεγχος εάν το πεδίο στη φόρμα αναζήτησης που έχει την ονομασία search είναι συμπληρωμένο. Τότε θα γίνει ερώτημα επιλογής στον πίνακα των προιόντων με τον τελεστή LIKE, δηλαδή ο όρος που αναζητά ο χρήστης εντοπίζεται σε οποιοδήποτε στήλη (είτε στο όνομα είτε στο SKU). Τέλος, το κουμπί Reset που εμφανίζεται δεξιά από τον όρο αναζήτησης ανακατευθύνει το χρήστη στην αρχική σελίδα.

## 4.6 Σελίδα προϊόντος

Η μέθοδος που θα εμφανίσει πληροφορίες για ένα προιόν είναι η viewProductDetails και αναμένει για την παράμετρο $id, η οποία εντοπίζεται στο τελευταίο τμήμα το URL (μετά το τελευταίο slash), κάτι που περιγράψαμε στην ενότητα των routes. Θα γίνει ερώτημα επιλογής του προιόντος με βάση το id του που βρίσκεται στο όρισμα της μεθόδου και το αποτέλεσμα του ερωτήματος θα δοθεί στο view productDetails, το οποίο δε διαθέτει νέα στοιχεία και ουσιαστικά είναι αντιγραφή του block του κάθε προιόντος στην αρχική σελίδα.

## 4.7 Σελίδα κατηγορίας

Η μέθοδος που θα αναλάβει να εμφανίσει τα προιόντα μιας κατηγορίας που επιλέγει ο χρήστης είναι η viewCategory, η οποία αναμένει για την παράμετρο $id, δηλαδή το id της κατηγορίας. Αρχικά θα γίνει έλεγχος ότι η κατηγορία υπάρχει στη βάση δεδομένων και εάν αυτό είναι αληθές θα ανακτηθεί η κατηγορία με βάση το id της όπως επίσης θα γίνει ανάκτηση και των προιόντων που ανήκουν σε αυτή την κατηγορία με βάση τη στήλη cate\_id. Οι μεταβλητές $category και $products θα γνωστοποιηθούν στο view με όνομα αρχείου viewCategory. Στο συγκεκριμένο view δε θα δούμε κάποιο νέο στοιχείο καθώς ομοιάζει με τα 2 αμέσως προηγούμενα views.

## 4.8 Προσθήκη στο καλάθι

Η προσθήκη στο καλάθι απαιτεί 2 μέρη, το client και το server και κάθε μέρος θα αναλυθεί ξεχωριστά αλλά συνεργάζονται μεταξύ τους.

## 4.8.1 Όψη Javascript

Εδώ ανοίγουμε μια παρένθεση καθώς είναι απαραίτητο στο Laravel όλα τα αιτήματα τύπου POST να περιέχουν στην επικεφαλίδα τους (headers) το CSRF token, μέσω του οποίου ελέγχεται η εγκυρότητα μιας φόρμας και αποτελεί μεγάλο χαρακτηριστικό ασφαλείας και αποτρέπει σημαντικό ποσοστό κακόβουλων επιθέσεων. Το CSRF ελέγχεται από το Laravel με μόνη απαίτηση από εμάς να το εισάγουμε στον HTML κώδικα της σελίδας. Για να μπορέσουμε να κάνουμε ασύγχρονα αιτήματα Ajax προς το server από Javascript κώδικα είναι απαραίτητο να εισάγουμε το ακόλουθο block:

$.ajaxSetup({

    headers: {

        "X-CSRF-TOKEN": $('meta[name="csrf-token"]').attr("content"),

    },

});

Ανακτούμε το CSRF token που βρίσκεται σε meta κόμβο με όνομα csrf-token και έτσι όλα τα αιτήματα Ajax το «κουβαλάνε» στις επικεφαλίδες τους.

Το client side έχει προγραμματιστεί με jQuery κώδικα (βιβλιοθήκη jQuery), ο οποίος είναι ουσιαστικά Javascript κώδικας, αλλά πιο εύκολος στην ανάγνωση και λιγότερο φλύαρος σε εντολές.

Ορίζουμε έναν ακροατή συμβάντος click στο κουμπί με κλάση addToCartBtn, δηλαδή στο κουμπί «Add to cart» και θα ανακτήσουμε το id του προιόντος το οποίο βρίσκεται σε έναν κόμβο input ο οποίος είναι κρυφός, δε φαίνεται δηλαδή. Ύστερα θα κάνουμε ασύγχρονο αίτημα Ajax τύπου POST προς το server και συγκεκριμένα στον πόρο add-to-cart εισάγοντας στο φορτίο (payload) του αιτήματος το product\_id και το product\_qty, το οποίο είναι 1. Τα αιτήματα Ajax διαθέτουν ακροατές συμβάντος επιτυχίας, λάθους και ολοκλήρωσης. Εμείς χρησιμοποιούμε τον ακροατή συμβάντος επιτυχίας, δηλαδή η επικοινωνία με το server ήταν επιτυχημένη όπου και θα ανακτήσουμε και θα εμφανίσουμε το μήνυμα επιτυχίας (το οποίο είναι σε μορφή JSON), το οποίο προέρχεται από το server. Το μήνυμα εξαφανίζεται μετά από 2.5 δευτερόλεπτα και γίνεται επαναφόρτωση της σελίδας. Σημειώνεται ότι η προσθήκη στο καλάθι προϋποθέτει ότι ο χρήστης είναι συνδεδεμένος.

## 4.8.2 Όψη PHP

Η μέθοδος που θα αναλάβει να διαχειριστεί το αίτημα προσθήκης στο καλάθι είναι η addProduct όπου θα λάβει το φορτίο του αιτήματος (product\_id και product\_qty) και έπειτα θα κάνει έλεγχο εάν ο χρήστης είναι συνδεδεμένος. Σε περίπτωση που δεν είναι, δεν προβλέπεται η προσθήκη στο καλάθι και εμφανίζεται μήνυμα ότι απαιτείται η είσοδός του. Διαφορετικά, θα ανακτήσουμε το προιόν από τη βάση με βάση το id του και έπειτα θα ελέγξουμε εάν το προιόν υπάρχει ήδη στο καλάθι (πίνακας cart) με βάση το id του προιόντος και το id του συνδεδεμένου χρήστη καθώς δε θέλουμε να προσθέσουμε το ίδιο προιόν ξανά στο καλάθι και θα εμφανιστεί ανάλογο μήνυμα στο χρήστη σε αυτή την περίπτωση. Διαφορετικά, θα δημιουργηθεί νέα εγγραφή στον πίνακα carts με βάση δίνοντας το id του χρήστη, το id του προιόντος και τη ζητούμενη ποσότητα η οποία είναι 1 και ένα μήνυμα επιτυχίας θα επιστραφεί σε μορφή JSON.

## 4.8.3 Αυξομείωση ποσότητας στο καλάθι

Ο χρήστης βρισκόμενος στη σελίδα του καλαθιού μπορεί να αυξομειώσει την ποσότητα για το κάθε προιόν.

## 4.8.3.1 Όψη Javascript

Ορίζουμε έναν ακροατή συμβάντος click στα κουμπιά συν και πλην του στη γραμμή του προιόντος και θα μετατρέψουμε τη ζητούμενη ποσότητα σε ακέραιο και θα αυξήσουμε ή θα μειώσουμε την υφιστάμενη ποσότητα κατά μία μονάδα αναλόγως της ενέργειας του χρήστη.

Έχουμε ορίσει έναν δεύτερο ακροατή συμβάντος όπου θα ανακτήσουμε το id του προιόντος και την υφιστάμενη ποσότητα και θα τα στείλουμε ως φορτίο στο server μέσω ενός ασύγχρονου αιτήματος προς το server στον πόρο update-cart. Ο ορισθείς ακροατής συμβάντος επιτυχίας του αιτήματος απλά θα κάνει επαναφόρτωση της σελίδας.

## 4.8.3.2 Όψη PHP

Η αρμόδια μέθοδος που θα αναλάβει το αίτημα ενημέρωσης του καλαθιού είναι η updatecart και το σώμα της οποίας δε διαφέρει πολύ από τη μέθοδο addProduct. Θα ανακτηθεί το καλάθι του χρήστη και απλά θα ενημερώσει τη νέα ζητούμενη ποσότητα και ανάλογο μήνυμα επιτυχίας σε μορφή JSON θα επιστραφεί στον client.

## 4.8.4 Διαγραφή προιόντος από το καλάθι

Όταν ο χρήστης πατήσει στο κουμπί “Remove” στη γραμμή του προιόντος τότε επιχειρεί διαγραφή του προιόντος από το καλάθι και θα γίνει ασύγχρονο αίτημα Ajax στον πόρο remove-cart-item με το φορτίο του αιτήματος να περιέχει το id του προιόντος. Τη σκυτάλη θα λάβει τώρα ο server και θα γίνει κλήση της μεθόδου removeFromCart όπου έχοντας το id του προιόντος θα ανακτηθεί το καλάθι και εάν υπάρχει τότε θα γίνει ερώτημα διαγραφής της γραμμής με βάση το id του προιόντος και το id του χρήστη. Η απάντηση θα επιστραφεί στον client σε μορφή JSON.

## 4.8.5 Προβολή καλαθιού

Η μέθοδος viewcart θα αναλάβει να εμφανίσει το καλάθι του χρήστη στο view viewCart χρησιμοποιώντας το id του συνδεδεμένου χρήστη.

## 4.8.6 Παραγγελία

Ο χρήστης έχει πατήσει στο κουμπί “Checkout” και μεταφέρεται στη σελίδα επιβεβαίωσης της παραγγελίας όπου θα εμφανιστεί το view viewCheckoutForm (βρισκόμαστε στο CheckoutController). Η μέθοδος απλά θα ανακτήσει το καλάθι του χρήστη και θα το εμφανίσει στο view μέσω ενός βρόγχου foreach προβάλλοντας το όνομα, την τιμή, την ποσότητα και το μερικό σύνολο για κάθε προιόν. Ο χρήστης βλέπει μια φόρμα όπου καλείται να συμπληρώσει το ονοματεπώνυμό του, τη διεύθυνσή του και τον τρόπο πληρωμής (εικονικός). Την παραγγελία θα τη διεκπεραιώσει η μέθοδος placeorder το CheckoutController. Θα δημιουργηθεί νέο αντικείμενο Order (εγγραφή στον πίνακα orders) συμπληρώνοντας τα στοιχεία που πληκτρολόγησε ο χρήστης στη φόρμα, το τελικό σύνολο και έναν τυχαίο αριθμό μεταξύ 111111 και 999999 που αποτελεί τον αριθμό αποστολής της παραγγελίας για παράδοση στον πελάτη. Ύστερα θα εισάγουμε στον πίνακα order\_items (αντικείμενο OrderItem) τις αντίστοιχες εγγραφές του πίνακα cart για να γνωρίζουμε τι είδη και ποσότητες παρήγγειλε ο πελάτης συμπληρώνοντας επίσης το order\_id που λάβαμε προηγουμένως από την αποθήκευση της παραγγελίας. Θα ενημερώσουμε την ποσότητα κάθε είδους αφαιρώντας την υφιστάμενη ποσότητα από την παραγγελθείσα ποσότητα για να έχουμε ενημερωμένο απόθεμα. Θα ενημερώσουμε επίσης τη διεύθυνση και το ονοματεπώνυμο του πελάτη στον πίνακα στον πίνακα users και τελικά θα διαγράψουμε το καλάθι του χρήστη και μήνυμα επιτυχίας θα εμφανιστεί στο χρήστη συνοδευόμενο με το id της νέας παραγγελίας, ο οποίος ανακατευθύνεται στην αρχική σελίδα.

## 4.9 Διαχείριση

Το AdminController θα διαχειριστεί όλα τα αιτήματα του διαχειριστή. Μετά την επιτυχή είσοδό του στο σύστημα, η μέθοδος index θα εμφανίσει την αρχική οθόνη της διαχείρισης με τους συνδέσμους που μπορεί να μεταβεί (view: admin/index). Η είσοδος στη σελίδα της διαχείρισης χρησιμοποιώντας το ψευδώνυμο και κωδικό (ίδια): [admin@admin.com](mailto:admin@admin.com). Η εγγραφή του διαχειριστή στον πίνακα users έχει μπει χειροκίνητα στη βάση (δεν έγινε με εγγραφή μέσω της φόρμας της εφαρμογής).

## 4.9.1 Διαχείριση κατηγοριών

Η μέθοδος categoriesIndex θα ανακτήσει όλες τις κατηγορίες και θα τις γνωστοποιήσει στο view admin/categoryIndex, όπου μέσω ενός βρόγχου foreach θα γίνει επανάληψη των κατηγοριών εμφανίζοντας την εικόνα, το id, την ονομασία, την περιγραφή της κάθε κατηγορίας. Σε αυτήν την οθόνη ο διαχειριστής μπορεί να επεξεργαστεί ή να διαγράψει την κατηγορία. Η επεξεργασία της κατηγορίας γίνεται από τη μέθοδο editCategory, η οποία αναμένει ως παράμετρο το id της κατηγορίας. Όταν ο χρήστης πατήσει το κουμπί «Επεξεργασία» στέλνεται ένα αίτημα στο server και τότε καλείται η μέθοδος updateCategory. Θα γίνει η ανάκτηση της κατηγορίας μέσω του μοντέλου Category (μέσω της παραμέτρου $id) και μετά θα γίνει έλεγχος εάν ο χρήστης έχει επιλέξει να ανεβάσει εικόνα. Εάν ισχύει αυτό, τότε θα διαγραφεί η παλιά εικόνα (ελέγχοντας πρώτα εάν υπάρχει) και θα αποδοθεί ένα νέο όνομα στην εικόνα χρησιμοποιώντας την τρέχουσα χρονοσφραγίδα του Unix (μεγάλος ακέραιος) ενωμένο με την επέκταση αρχείου και τελικά θα αποθηκευτεί στο δημόσιο φάκελο (public) στην τοποθεσία images/category. Τέλος, θα γίνει ενημέρωση της ονομασίας και της περιγραφής της κατηγορίας μέσω του μοντέλου Category και o server θα ανακατευθύνει το χρήστη στη σελίδα προβολής όλων των κατηγοριών,

Η διαγραφή κατηγορίας θα γίνει μέσω της μεθόδου deleteCategory όταν ο χρήστης πατήσει το κουμπί “Delete” σε μια κατηγορία. Η μέθοδος θα ανακτήσει την κατηγορία βάσει του id της, θα διαγράψει την αντίστοιχη εικόνα και τελικά θα διαγραφεί η κατηγορία από τον πίνακα categories.

Η προσθήκη κατηγορίας ομοιάζει με την ενημέρωση κατηγορίας και η αρμόδια μέθοδος να διαχειριστεί το αίτημα είναι η updateCategory, όπου γίνονται οι ίδιες ενέργειες με τη διαφορά ότι εδώ γίνεται προσθήκη νέας εγγραφής.

## 4.9.2 Διαχείριση προιόντων

Ο διαχειριστής προβάλλει τα προϊόντα του καταστήματος (μέθοδος ProductsIndex και view admin/productIndex) όπου απλά θα γίνει ερώτημα επιλογής όλων των προιόντων και μέσω ενός βρόγχου foreach θα γίνει επανάληψη των προιόντων και θα προβληθούν οι στήλες: id, ονομασία, κατηγορία, περιγραφή, τιμή, ποσότητα (εάν είναι θετική, διαφορετικά το λεκτικό Out of Stock), εικόνα και οι ενέργειες επεξεργασίας και διαγραφής.

Όταν ο χρήστης πατήσει το κουμπί “Edit” ο χρήστης θα μεταφερθεί στη σελίδα επεξεργασίας του προιόντος και η μέθοδος editProduct έχοντας ως όρισμα το id του προιόντος θα κάνει ερώτημα επιλογής όλων των κατηγοριών και ερώτημα επιλογής του συγκεκριμένου προιόντος και οι μεταβλητές θα γνωστοποιηθούν στο view admin/editProduct. Σε αυτή τη σελίδα, ο διαχειριστής μπορεί να διαχειριστεί όλα τα πεδία του προιόντος όπως επίσης μπορεί να αλλάξει την κατηγορία μέσω ενός dropdown στοιχείου, το οποίο περιέχει όλες τις κατηγορίες με επιλεγμένη την ενεργή κατηγορία του προιόντος. Όταν ο χρήστης υποβάλλει τη φόρμα της σελίδας θα αναλάβει το αίτημα να το διαχειριστεί η μέθοδος updateProduct όπου θα ανακτήσει το προιόν με βάση το id του, θα συμπληρώσει τις στήλες του προιόντος με τα αντίστοιχα πεδία της φόρμας και θα αλλάξει την εικόνα εάν ο χρήστης έχει επιλέξει να ανεβάσει νέα (ίδια διαδικασία όπως την επεξεργασία κατηγορίας).

Η διαγραφή κατηγορίας γίνεται όταν ο διαχειριστής πατήσει το κουμπί “Delete” στη λίστα των προιόντων και τότε το αίτημα θα το διαχειριστεί η μέθοδος deleteProduct, η οποία έχοντας το id του προιόντος θα το διαγράψει από τη βάση όπως και την αντίστοιχη εικόνα εάν υπάρχει.

Η προσθήκη προιόντος θα πραγματοποιηθεί όταν ο διαχειριστής βρισκόμενος στη σελίδα Εισαγωγή προιόντος πατήσει το κουμπί “Submit” υποβάλλοντας τη φόρμα της σελίδας. Το αίτημα θα το αναλάβει η μέθοδος insertProduct και θα δημιουργήσει νέο αντικείμενο Product και θα αντιστοιχίσει τις τιμές της φόρμες με τα πεδία στη βάση του πίνακα products και θα προσθέσει νέα εικόνα στο δημόσιο φάκελο στην τοποθεσία images/products με τον ίδιο τρόπο που γίνεται η προσθήκη εικόνας σε κατηγορία.

## 4.10 Σελίδα: Σχετικά με εμάς

Το controller UserController και η μέθοδος viewAboutUs θα αναλάβει να εμφανίσει στοιχεία για το ηλεκτρονικό κατάστημα (view about-us, όπου περιέχεται βασικός HTML κώδικας).

## 4.11 Παραγγελίες πελάτη

Το controller UserController και η μέθοδος viewMyOrders θα αναλάβει να εμφανίσει τις παραγγελίες που έχει πραγματοποιήσει ο συνδεδεμένος πελάτης. Θα γίνει ερώτημα επιλογής στον πίνακα orders και το αποτέλεσμα του ερωτήματος θα εμφανιστεί στο view myOrders μέσω ενός βρόγχου foreach, ο οποίος κάνει επανάληψη όλες τις παραγγελίες και για κάθε παραγγελία εμφανίζονται οι εξής στήλες:

1. ID Παραγγελίας
2. Tracking number
3. Τρόπος Πληρωμής
4. Τελικό Ποσό
5. Ενέργεια (προβολή λεπτομερειών παραγγελίας)

Όταν ο χρήστης πατήσει το κουμπί «Ενέργεια», θα μεταφερθεί στη σελίδα των λεπτομερειών της παραγγελίας με τη βοήθεια της μεθόδου view, η οποία δεχόμενη το id της παραγγελίας και χρησιμοποιώντας το id του συνδεδεμένου χρήστη, θα λάβει τις πληροφορίες της παραγγελίας και το αποτέλεσμα θα αποδοθεί στο view singleOrderDetails. Θα εμφανιστούν τα στοιχεία επικοινωνίας και διεύθυνσης του χρήστη και μέσω ενός βρόγχου foreach θα εμφανιστούν τα παραγγελθέντα είδη και συγκεκριμένα η ονομασία, η παραγγελθείσα ποσότητα, η εικόνα και η τιμή του είδους όπως και το τελικό σύνολο της παραγγελίας.

# ΚΕΦΑΛΑΙΟ 5 – Στιγμιότυπα χρήσης

## 5.1 Αρχική οθόνη

Graphical user interface, application, website

Description automatically generated

## 5.2 Αναζήτηση

Graphical user interface, application

Description automatically generated

## 5.3 Εγγραφή

Graphical user interface, application

Description automatically generated

## 5.4 Είσοδος

Graphical user interface, application, Teams

Description automatically generated

## 5.5 Σελίδα προιόντος

Graphical user interface, application, website

Description automatically generated

## 5.6 Σελίδα κατηγορίας

Graphical user interface, application, website

Description automatically generated

## 5.7 Σελίδα Σχετικά με εμάς

Graphical user interface, Word

Description automatically generated

## 5.8 Καλάθι

Graphical user interface, application

Description automatically generated

## 5.9 Παραγγελία

Graphical user interface

Description automatically generated

## 5.10 Επιτυχής παραγγελία

A picture containing graphical user interface

Description automatically generated

## 5.11 Παραγγελίες πελάτη

Graphical user interface, text, application

Description automatically generated

## 5.12 Κεντρική οθόνη διαχείρισης

Graphical user interface, text, application

Description automatically generated

## 5.13 Διαχείριση κατηγοριών

Graphical user interface, application

Description automatically generated

## 5.13 Επεξεργασία κατηγορίας

Graphical user interface, application

Description automatically generated

## 5.14 Εισαγωγή κατηγορίας

Graphical user interface, text, application

Description automatically generated

## 5.15 Διαχείριση προϊόντων

Graphical user interface, application

Description automatically generated

## 5.16 Επεξεργασία προϊόντος

Graphical user interface, application, Teams

Description automatically generated

## 5.17 Εισαγωγή προϊόντος

A screenshot of a computer

Description automatically generated