ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Σχολή Τεχνολογίας

Τμήμα Ψηφιακών Συστημάτων

Ανάπτυξη Ηλεκτρονικού Κατα-στήματος με php Laravel

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΓΕΩΡΓΙΟΣ ΚΟΛΩΝΑΣ (ΑΜ: 3119152)

Επιβλέπων: Φώτιος Κόκκορας, *Επίκουρος Καθηγητής*

ΛΑΡΙΣΑ,ΜΑΡΤΙΟΣ 2024

UNIVERSITY OF THESSALY

School of Technology

Digital Systems Department

"Developing an E-commerce Site with PHP Laravel

BACHELOR THESIS

GEORGIOS KOLONAS (RN: 3119152)

Supervisor: ***Fotis Kokkoras, Assistant Professor***

LARISSA, March 2024

Υπεύθυνη Δήλωση περί Ακαδημαϊκής Δεοντολογίας και Πνευματικών Δικαιωμάτων

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα πτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον, και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο/Η Δηλών/ούσα

(Υπογραφή)

Γεώργιος Κολώνας

13/3/2024

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων/πουσα Φώτιος Κόκκορας
Επίκουρος καθηγητής,
Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας

Μέλος **Γεώργιος Κακαρόντζας**
Καθηγητής , Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο σαλίας

Ημερομηνία έγκρισης: 15-3-2024

Περίληψη

Η πτυχιακή αυτή εργασία επικεντρώνεται στην κατασκευή ενός ηλεκτρονικού καταστήματος χρησιμοποιώντας το php framework Laravel. Ο σκοπός της εργασίας είναι να αναπτυχθεί ένα πλήρες, λειτουργικό, και ασφαλές ηλεκτρονικό κατάστημα που θα επιτρέπει στους χρήστες την ασφαλής εγγραφή τους , την αναζήτηση προϊόντων για ευκολία και γρήγορη πρόσβαση , την αξιολόγηση προϊόντων από τους χρήστες για τη δημιουργία εμπιστοσύνης, την προσθήκη προϊόντων στο καλάθι αγορών και τέλος με την ολοκλήρωση συναλλαγής όπου θα γίνεται η αγορά τών προιόντων .Με αυτήν την προσέγγιση, η πτυχιακή αυτή θα καλύψει κάποιες από τις βασικές λειτουργίες που απαιτούνται για τη δημιουργία και τη λειτουργία ενός ηλεκτρονικού καταστήματος.

Abstract

This bachelor's thesis centers around the development of an electronic store using the PHP framework Laravel. The objective of the project is to create a complete, functional, and secure online store that enables users to register securely, facilitates easy and quick product searches, allows user ratings for building trust, supports the addition of products to the shopping cart, and concludes with the completion of transactions for product purchases. With this approach, the thesis aims to encompass fundamental functionalities necessary for establishing and operating an electronic commerce platform.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου προς τον καθηγητή μου για την ευκαιρία που μου έδωσε να εργαστώ πάνω σε ένα τέτοιο θέμα. Η ευκαιρία αυτή μου επέτρεψε να εξερευνήσω και να αναπτύξω τις ικανότητές μου σε έναν τομέα που με ενδιαφέρει ιδιαίτερα.

Επιπλέον είμαι ευγνώμων προς όλους εκείνους που μοιράζονται την εμπειρία και τη γνώση τους μέσα από βίντεο και από διάφορα site με τον κόσμο, καθώς η διαθεσιμότητα αυτών των πόρων μου έδωσε την ευκαιρία να εξελίξω τις ικανότητές μου πάνω στην ανάπτυξη ιστοσελιδών . Αυτή η εμπειρία με έκανε πιο αυτόνομο και αυτοδιδακτο, και ελπίζω ότι θα έχει μακροπρόθεσμα θετική επίδραση στην προσωπική και επαγγελματική μου ανάπτυξη.

Κολώνας Γέωργιος

ημερομηνία

Περιεχόμενα

1 Εισαγωγή

Στον σημερινό ψηφιακό κόσμο, έχει καταστεί κρίσιμο για τις επιχειρήσεις να δημιουργήσουν μια διαδικτυακή παρουσία προκειμένου να προσεγγίσουν μεγαλύτερο κοινό και να αυξήσουν τις πωλήσεις. Ένας από τους πιο συνηθισμένους τρόπους για να το κάνουν αυτό οι επιχειρήσεις είναι να δημιουργήσουν έναν ιστότοπο, ο οποίος λειτουργεί ως εικονική βιτρίνα για να περιηγηθούν οι πελάτες και να αγοράσουν προϊόντα ή υπηρεσίες. Ένας ιστότοπος ηλεκτρονικού εμπορίου, που συνήθως αναφέρεται ως ηλεκτρονικό κατάστημα , είναι ένας τύπος ιστότοπου που ειδικεύεται στην πώληση αγαθών ή υπηρεσιών απευθείας σε πελάτες μέσω του Διαδικτύου. Αυτοί οι ιστότοποι προσφέρουν συνήθως μια σειρά λειτουργιών για τη βελτίωση της εμπειρίας αγορών, όπως σύστημα καλαθιού αγορών, ασφαλείς επιλογές πληρωμής και λειτουργίες αποστολής και διαχείρισης.

Η δημιουργία και η διαχείριση ηλεκτρονικών καταστημάτων απαιτεί συχνά συνδυασμό τεχνολογιών ανάπτυξης ιστού, όπως HTML, CSS, JavaScript, PHP και δημοφιλών πλαισίων όπως το Laravel, για να διασφαλιστεί ότι ο ιστότοπος είναι φιλικός προς το χρήστη, ασφαλής και επεκτάσιμος. Για παράδειγμα, η HTML και η CSS χρησιμοποιούνται για τη δομή και το στυλ της διάταξης και του σχεδιασμού του ιστότοπου, ενώ η JavaScript παρέχει διαδραστικότητα και λειτουργικότητα, όπως αναδυόμενα παράθυρα, κινούμενα σχέδια και δυναμικά στοιχεία σελίδας. Η PHP χρησιμοποιείται συνήθως για το χειρισμό της επεξεργασίας από την πλευρά του διακομιστή, όπως οι υποβολές φορμών, οι αλληλεπιδράσεις με βάση δεδομένων και ο έλεγχος ταυτότητας χρήστη, που είναι απαραίτητες για ιστότοπους ηλεκτρονικού εμπορίου.

Με την ανάπτυξη των ηλεκτρονικών αγορών, τα ηλεκτρονικά καταστήματα έχουν γίνει ολοένα και πιο σημαντικό στοιχείο της ψηφιακής αγοράς. Επιτρέπουν στις επιχειρήσεις να επεκτείνουν την εμβέλειά τους πέρα από τις φυσικές βιτρίνες και παρέχουν στους πελάτες μια βολική και εξατομικευμένη εμπειρία αγορών. Τα ηλεκτρονικά καταστήματα μπορούν να κυμαίνονται από μικρές επιχειρήσεις που πωλούν εξειδικευμένα προϊόντα έως μεγάλες πολυεθνικές εταιρείες με έναν τεράστιο κατάλογο προϊόντων. Ωστόσο,

ανεξάρτητα από το μέγεθος ή την εμβέλειά τους, όλα τα ηλεκτρονικά καταστήματα μοιράζονται τον ίδιο θεμελιώδη στόχο να παρέχουν στους πελάτες μια ευχάριστη εμπειρία αγορών.

2 Διαδίκτυο και Παγκόσμιος Ιστός

2.1 Παγκόσμιος Ιστός

Ο Παγκόσμιος Ιστός (WWW) δημιουργήθηκε στα τέλη της δεκαετίας του 1980 και στις αρχές της δεκαετίας του 1990 από έναν Βρετανό επιστήμονα υπολογιστών ονόματι Tim Berners-Lee, ο οποίος εργαζόταν στο CERN. Στόχος του Berners-Lee ήταν να δημιουργήσει ένα σύστημα που θα επέτρεπε στους ερευνητές του CERN να μοιράζονται πληροφορίες πιο εύκολα. Ανέπτυξε το πρώτο πρόγραμμα περιήγησης ιστού, που ονομάζεται WorldWideWeb, και τον πρώτο διακομιστή ιστού, που κυκλοφόρησαν το 1991. Το πρόγραμμα περιήγησης WorldWideWeb επέτρεπε στους χρήστες να δημιουργούν, να επεξεργάζονται και να προβάλλουν ιστοσελίδες χρησιμοποιώντας μια γραφική διεπαφή, η οποία το έκανε πολύ πιο προσιτό από τα προηγούμενα συστήματα που βασιζόταν σε διεπαφές γραμμής εντολών. Η βασική καινοτομία του Παγκόσμιου Ιστού ήταν η χρήση υπερκειμένου, το οποίο επέτρεπε στους χρήστες να πλοηγούνται μεταξύ διαφορετικών ιστοσελίδων κάνοντας κλικ σε υπερσυνδέσμους. Αυτό διευκόλυνε πολύ την πρόσβαση και την οργάνωση πληροφοριών στο Διαδίκτυο και άνοιξε το δρόμο για την έκρηξη του περιεχομένου ιστού και του ηλεκτρονικού εμπορίου που βλέπουμε σήμερα.

Με την πάροδο του χρόνου, ο Παγκόσμιος Ιστός έχει εξελιχθεί για να γίνει ένα παγκόσμιο δίκτυο διασυνδεδεμένων εγγράφων υπερκειμένου, στα οποία έχει πρόσβαση μέσω του Διαδικτύου χρησιμοποιώντας προγράμματα περιήγησης Ιστού.

2.2 Web browser

Το πρόγραμμα περιήγησης Ιστού είναι ένας τύπος εφαρμογής λογισμικού που επιτρέπει στους χρήστες να έχουν πρόσβαση και να εμφανίζουν ιστοσελίδες στο Διαδίκτυο. Παρέχει μια διεπαφή που επιτρέπει στους χρήστες να περιηγούνται μεταξύ διαφορετικών ιστοσελίδων, να εισάγουν διευθύνσεις URL ή όρους αναζήτησης και να αλληλεπιδρούν με περιεχόμενο ιστού, όπως φόρμες και στοιχεία πολυμέσων. Τα προγράμματα περιήγησης Ιστού χρησιμοποιούν HTTP για να επικοινωνούν με διακομιστές Ιστού και να ανακτούν ιστοσελίδες, τις οποίες στη συνέχεια εμφανίζουν στην οθόνη του χρήστη.

Ερμηνεύουν τον κώδικα προγραμματισμού που χρησιμοποιείται για τη δημιουργία ιστοσελίδων, όπως HTML, CSS και JavaScript, και παρουσιάζουν το περιεχόμενο σε μια μορφή που μπορούν να προβληθούν και να αλληλεπιδράσουν οι χρήστες. Υπάρχουν πολλά διαφορετικά προγράμματα περιήγησης ιστού διαθέσιμα, το καθένα με τα δικά του μοναδικά χαρακτηριστικά, διεπαφή χρήστη και χαρακτηριστικά απόδοσης.

2.3 Εφαρμογές Διαδικτύου

Το Διαδίκτυο έχει επηρεάσει και διαμορφώσει τον τρόπο ζωής μας και προσφέρει πολλές βασικές εφαρμογές που καλύπτουν πολλούς τομείς. Ορισμένες από τις βασικές εφαρμογές του διαδικτύου περιλαμβάνουν:

Ηλεκτρονικό Ταχυδρομείο (Email): Το email είναι μια από τις παλαιότερες και βασικές υπηρεσίες του Διαδικτύου, που επιτρέπει την ανταλλαγή ηλεκτρονικών μηνυμάτων μεταξύ ατόμων σε ολόκληρο τον κόσμο.

Ιστός (World Wide Web - WWW): Ο Παγκόσμιος Ιστός είναι μια από τις κύριες εφαρμογές του Διαδικτύου, παρέχοντας πρόσβαση σε πληροφορίες, ιστοσελίδες, βίντεο, φωτογραφίες και πολλά άλλα.

Κοινωνικά Δίκτυα: Πλατφόρμες όπως το Facebook, το Twitter, το Instagram και το LinkedIn παρέχουν μέσα για τη δημιουργία και τη διατήρηση κοινωνικών σχέσεων, καθώς και την κοινοποίηση πληροφοριών και περιεχομένου.

Αναζήτηση (Search Engines): Υπηρεσίες όπως το Google, το Bing και το Yahoo παρέχουν δυνατότητα αναζήτησης για να βρίσκουμε πληροφορίες στον Παγκόσμιο Ιστό.

Ηλεκτρονικό Εμπόριο (E-commerce): Πλατφόρμες όπως το Amazon, το eBay και πολλές άλλες επιτρέπουν την αγορά και πώληση προϊόντων και υπηρεσιών μέσω του Διαδικτύου.

Υπηρεσίες Καταναλωτικής Πλειοδοσίας (Online Auctions): Πλατφόρμες όπως το eBay επιτρέπουν στους χρήστες να συμμετέχουν σε δημοπρασίες για την αγορά και πώληση αντικειμένων.

Υπηρεσίες Μουσικής/Βίντεο Κοινοποίησης: Πλατφόρμες όπως το YouTube, το Spotify, το Netflix παρέχουν πρόσβαση σε μουσική, βίντεο και άλλο πολυμέσο.

Υπηρεσίες Συνεργασίας (Collaboration Tools): Εφαρμογές όπως το Google Docs, το Microsoft Teams και το Slack διευκολύνουν τη συνεργασία μεταξύ ανθρώπων σε διάφορες τοποθεσίες.

2.4 Πρωτόκολλο HTTP

Το πρωτόκολλο HTTP (Hypertext Transfer Protocol) αποτελεί ένα από τα κύρια πρωτόκολλα επικοινωνίας στο Διαδίκτυο, υπεύθυνο για τη μεταφορά και τη διαχείριση υπερκειμένων δεδομένων, κυρίως κειμένου, μεταξύ ενός πελάτη (όπως ένας web browser) και ενός εξυπηρετητή (web server).

Ορισμένα βασικά χαρακτηριστικά του πρωτοκόλλου HTTP περιλαμβάνουν:

Request: Ο πελάτης στέλνει μια HTTP αίτηση σε έναν εξυπηρετητή για τη ζήτηση συγκεκριμένων δεδομένων. Η αίτηση περιλαμβάνει τον τύπο του αιτήματος, όπως GET για λήψη δεδομένων ή POST για αποστολή δεδομένων από τον πελάτη.

Response: Ο εξυπηρετητής ανταποκρίνεται στο αίτημα με μια HTTP απόκριση που περιλαμβάνει τα απαιτούμενα δεδομένα. Η απόκριση περιλαμβάνει επίσης έναν κωδικό κατάστασης που υποδεικνύει την επιτυχία ή την αποτυχία του αιτήματος.

Stateless: Το πρωτόκολλο HTTP είναι "χωρίς κατάσταση", κάθε αίτηση είναι ανεξάρτητη από τις προηγούμενες αιτήσεις, χωρίς διατήρηση πληροφοριών κατάστασης.

Hypertext: Το πρωτόκολλο επικεντρώνεται στη μεταφορά υπερκειμένων δεδομένων, όπως κείμενο, συνδέσμους, εικόνες και πολυμέσα, αποτελώντας τη βάση για τη λειτουργία του Παγκόσμιου Ιστού (WWW).

Simple Architecture: Η αρχιτεκτονική του HTTP είναι απλή, επιτρέποντας εύκολη κατανόηση και υλοποίηση.

Επιπλέον, το πρωτόκολλο HTTP υποστηρίζει πρωτόκολλα ασφαλείας όπως το HTTPS, εξασφαλίζοντας την ασφαλή μεταφορά δεδομένων μέσω κρυπτογραφίας. Εξελίσσεται διαρκώς για να ανταποκρίνεται στις ανάγκες του σύγχρονου διαδικτυακού περιβάλλοντος

2.5 Γενικές Κατηγορίες Ιστοσελίδων

2.5.1 Στατικές Ιστοσελίδες

Αποτελούνται από αρχεία HTML και CSS που καθορίζονται κατά τη δημιουργία της ιστοσελίδας.Το περιεχόμενο παραμένει σταθερό και δεν αλλάζει κατά την διάρκεια της προβολής της ιστοσελίδας από τον χρήστη και δεν αλληλεπιδρούν δυναμικά με τους χρήστες ή τους επισκέπτες.

2.5.2 Δυναμικές Ιστοσελίδες

Δημιουργούνται κατά την εκτέλεση, με τη χρήση γλωσσών προγραμματισμού όπως PHP, ASP, Python, ή JavaScript.Το περιεχόμενο μπορεί να αλλάζει δυναμικά ανάλογα με τις παραμέτρους ή τις ενέργειες των χρηστών καί επιτρέπουν διαχείριση δεδομένων και περιεχομένου σε πραγματικό χρόνο.

2.5.3 Ενεργές Ιστοσελίδες

Περιλαμβάνουν προγράμματα ή κώδικα σε γλώσσες σεναρίου (π.χ., JavaScript, Flash, Java Applets).Ο κώδικας εκτελείται στον υπολογιστή του χρήστη, επιτρέποντας δυναμική αλληλεπίδραση και αλλαγές στην εμφάνιση ή το περιεχόμενο. Μπορεί να χρησιμοποιούνται για παιχνίδια, εφαρμογές διαδραστικού περιεχομένου, κλπ.

Οι επιλογές μεταξύ στατικών, δυναμικών και ενεργών ιστοσελίδων εξαρτώνται συχνά από τις ανάγκες του έργου, την πολυπλοκότητα του περιεχομένου και τη διαδραστικότητα που απαιτείται.

2.6 HTML , CSS, JAVASCRIPT

Η HTML είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται για τη δημιουργία και τη δομή περιεχομένου για τον Παγκόσμιο Ιστό. Είναι απαραίτητο συστατικό όλων των ιστοσελίδων και ιστοσελίδων. HTML σημαίνει HyperText Markup Language και χρησιμοποιεί ετικέτες και χαρακτηριστικά για να ορίσει και να μορφοποιήσει κείμενο, εικόνες, συνδέσμους και άλλα στοιχεία περιεχομένου σε μια ιστοσελίδα. Οι ετικέτες HTML χρησιμοποιούνται για να περιγράψουν το περιεχόμενο και τη δομή μιας ιστοσελίδας και μπορούν να χρησιμοποιηθούν για τη δημιουργία επικεφαλίδων, παραγράφων, λιστών, πινάκων, φορμών και άλλων τύπων περιεχομένου.Όταν μια ιστοσελίδα ζητείται από το πρόγραμμα περιήγησης ενός χρήστη, ο κώδικας HTML ανακτάται από τον διακομιστή ιστού και αποδίδεται από το πρόγραμμα περιήγησης. Το πρόγραμμα περιήγησης χρησιμοποιεί τον κώδικα HTML για να ερμηνεύσει και να εμφανίσει το περιεχόμενο της ιστοσελίδας στον χρήστη. Με την πάροδο του χρόνου, η HTML έχει εξελιχθεί για να ανταποκρίνεται στις μεταβαλλόμενες ανάγκες του Ιστού. Η πιο πρόσφατη έκδοση του HTML είναι η HTML5, η οποία περιλαμβάνει νέες δυνατότητες όπως υποστήριξη πολυμέσων, βελτιωμένες φόρμες και καλύτερη προσβασιμότητα. Η HTML είναι μια ευέλικτη και ισχυρή γλώσσα που είναι απαραίτητη για όποιον θέλει να δημιουργήσει περιεχόμενο για τον Ιστό. Η HTML μπορεί επίσης να συνδυαστεί με άλλες γλώσσες προγραμματισμού όπως CSS και JavaScript για να προσθέσει στυλ και διαδραστικότητα σε ιστοσελίδες.

CSS

Η CSS βασίζεται σε ένα σύνολο κανόνων που καθορίζουν τον τρόπο με τον οποίο πρέπει να διαμορφώνονται τα στοιχεία σε ένα έγγραφο. Αυτοί οι κανόνες μπορούν να εφαρμοστούν σε μεμονωμένα στοιχεία ή ομάδες στοιχείων σε μια ιστοσελίδα. Τα προγράμματα περιήγησης ιστού χρησιμοποιούν CSS για να ερμηνεύσουν και να εφαρμόσουν

αυτούς τους κανόνες στυλ, επιτρέποντας μια συνεπή εμφάνιση και αίσθηση σε πολλές σελίδες . Διαχωρίζοντας το περιεχόμενο και την παρουσίαση, το CSS διευκολύνει τη συντήρηση και την ενημέρωση του σχεδιασμού ενός ιστότοπου. Επιτρέπει επίσης μεγαλύτερη ευελιξία και ανταπόκριση, επιτρέποντας στις ιστοσελίδες να προσαρμόζονται σε διαφορετικά μεγέθη οθόνης και συσκευές. Με την πάροδο του χρόνου, το CSS έχει εξελιχθεί για να περιλαμβάνει νέες δυνατότητες όπως κινούμενα σχέδια και μεταβάσεις, παρέχοντας ακόμη περισσότερες δυνατότητες στους σχεδιαστές να δημιουργούν ελκυστικές και λειτουργικές ιστοσελίδες.

JAVASCRIPT

Οι προγραμματιστές χρησιμοποιούν JavaScript για να δημιουργήσουν ένα ευρύ φάσμα λειτουργιών, από απλή επικύρωση φόρμας έως πολύπλοκα κινούμενα σχέδια και παιχνίδια. Η γλώσσα χρησιμοποιεί ένα σύνολο κανόνων και σύνταξης για να καθορίσει πώς γράφεται και εκτελείται ο κώδικας. Ο κώδικας JavaScript μπορεί να προστεθεί σε ένα έγγραφο HTML ή να αποθηκευτεί σε ξεχωριστό αρχείο και να συνδεθεί με το έγγραφο. Όταν ένας χρήστης ζητά μια ιστοσελίδα, το πρόγραμμα περιήγησης χρησιμοποιεί JavaScript για να ερμηνεύσει και να εκτελέσει τον κώδικα που περιέχεται στη σελίδα. Αυτό επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές Ιστού που ανταποκρίνονται στις εισροές των χρηστών, να ενημερώνουν περιεχόμενο σε πραγματικό χρόνο και να αλληλεπιδρούν με σενάρια διακομιστή. Η JavaScript έχει εξελιχθεί με την πάροδο του χρόνου για να περιλαμβάνει νέες δυνατότητες, όπως ασύγχρονο προγραμματισμό και API για αλληλεπίδραση με λειτουργίες του προγράμματος περιήγησης.

2.7 PHP

Η PHP είναι μια γλώσσα δέσμης ενεργειών ανοιχτού κώδικα που χρησιμοποιείται συνήθως για την ανάπτυξη Ιστού. Επιτρέπει στους προγραμματιστές να δημιουργούν δυναμικές και διαδραστικές ιστοσελίδες, καθώς εκτελούνται από την πλευρά του διακομιστή και όχι από το πρόγραμμα περιήγησης από την πλευρά του πελάτη. Όταν ένας χρήστης ζητά μια ιστοσελίδα, ο κώδικας PHP εκτελείται στον διακομιστή για τη δημιουργία HTML που στη συνέχεια αποστέλλεται στο πρόγραμμα περιήγησης του χρήστη. Αυτό επιτρέπει τη δημιουργία ιστοτόπων που μπορούν να εκτελούν μια σειρά

λειτουργιών, όπως η επεξεργασία των εισροών των χρηστών, η αλληλεπίδραση με βάσεις δεδομένων και η δημιουργία περιεχομένου με βάση τις προτιμήσεις των χρηστών. Η PHP χρησιμοποιείται συχνά σε συνδυασμό με άλλες τεχνολογίες ανάπτυξης ιστού, συμπεριλαμβανομένων των HTML, CSS και JavaScript, και είναι συμβατή με διάφορους διακομιστές ιστού και λειτουργικά συστήματα. Με την ευελιξία και την απλότητά της, η PHP έχει γίνει μια δημοφιλής επιλογή για την ανάπτυξη δυναμικών και ισχυρών διαδικτυακών εφαρμογών, όπως ιστοσελίδες ηλεκτρονικού εμπορίου, συστήματα διαχείρισης περιεχομένου και πλατφόρμες μέσων κοινωνικής δικτύωσης.

2.8 MYSQL

Η MySQL αποτελεί ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS) που χρησιμοποιείται ευρέως για την αποθήκευση, την ανάκτηση και τη διαχείριση δεδομένων. Είναι ένα RDBMS ανοιχτού κώδικα, προσφέροντας τη δυνατότητα προσαρμογής και χρήσης χωρίς κόστος αδειοδότησης. Με την ταχύτητα και την αποδοτικότητά του, το MySQL χρησιμοποιείται ευρέως σε διάφορες εφαρμογές, από μικρές ιστοσελίδες έως μεγάλες επιχειρησιακές πλατφόρμες.

Το σύστημα αυτό παρέχει επίσης υψηλή ασφάλεια με δικαιώματα χρηστών και δυνατότητες κρυπτογράφησης δεδομένων. Είναι ευέλικτο και συμβατό με τη γλώσσα SQL, καθιστώντας το ιδανικό για εφαρμογές που απαιτούν αποθήκευση, ανάκτηση και επεξεργασία δεδομένων. Η ενεργή κοινότητά του παρέχει στήριξη και συνεχή εξέλιξη, ενισχύοντας τη φήμη και τη δημοτικότητά του στον κόσμο της ανάπτυξης λογισμικού.

2.9 APACHE

Ο Apache HTTP Server, κοινώς γνωστός ως Apache, είναι ένα ευρέως χρησιμοποιούμενο λογισμικό διακομιστή web ανοιχτού κώδικα που αναπτύχθηκε από το Apache Software Foundation. Η κύρια λειτουργία του είναι να εξυπηρετεί ιστοσελίδες και να διευκολύνει την παράδοση περιεχομένου μέσω του πρωτοκόλλου HTTP .Οι περισσότερες απο τις web εφαρμογές κατασκευάζονται με βάση τα χαρακτηριστικά που παρέχει ο Apache. Χρησιμοποιείτε με PHP,Perl,Python.

2.10 XML

Η XML είναι μια ευέλικτη γλώσσα σήμανσης που έχει σχεδιαστεί για την αποθήκευση και τη μεταφορά δεδομένων. Χρησιμοποιεί ετικέτες για να ορίσει στοιχεία, παρόμοια με την HTML, αλλά είναι πιο ευέλικτο καθώς οι χρήστες μπορούν να ορίσουν τις δικές τους ετικέτες. Η XML χρησιμοποιείται ευρέως για την αναπαράσταση δομημένων δεδομένων σε αναγνώσιμη και ανεξάρτητη από μηχανή μορφή. Χρησιμοποιείται συχνά στην ανάπτυξη ιστού, την ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων και τα αρχεία διαμόρφωσης. Τα έγγραφα XML είναι αναγνώσιμα από τον άνθρωπο, καθιστώντας τα εύκολα κατανοητά και η ιεραρχική τους δομή επιτρέπει την αποτελεσματική οργάνωση των πληροφοριών. Διαδραματίζει κρίσιμο ρόλο στη διευκόλυνση της επικοινωνίας και της ανταλλαγής δεδομένων μεταξύ διαφορετικών εφαρμογών και πλατφορμών.

2.11 Σύστημα Διαχείρισης Περιεχομένου (CMS):

Σύστημα Διαχείρισης Περιεχομένου (CMS): Το CMS είναι μια εφαρμογή λογισμικού που απλοποιεί τη δημιουργία, τη διαχείριση και την τροποποίηση ψηφιακού περιεχομένου, καθιστώντας το προσβάσιμο σε χρήστες χωρίς εκτεταμένη τεχνική εξειδίκευση. Οι πλατφόρμες CMS επιτρέπουν στους χρήστες να δημοσιεύουν, να επεξεργάζονται και να οργανώνουν εύκολα διάφορους τύπους περιεχομένου, όπως κείμενο, εικόνες και πολυμέσα, μέσω μιας φιλικής προς τον χρήστη διεπαφής. Τα δημοφιλή συστήματα CMS περιλαμβάνουν το WordPress, το Joomla και το Drupal.

2.12 CMS και ηλεκτρονικό εμπόριο (eShop)

Ορισμένες πλατφόρμες CMS, όπως το WordPress με WooCommerce ή το Magento, επεκτείνουν τις δυνατότητές τους για να περιλαμβάνουν λειτουργίες ηλεκτρονικού εμπορίου. Αυτό σημαίνει ότι μπορούν να χρησιμεύσουν ως βάση για την κατασκευή ηλεκτρονικών καταστημάτων ή ηλεκτρονικών καταστημάτων. Οι λειτουργίες ηλεκτρονικού εμπορίου που είναι ενσωματωμένες σε ένα CMS επιτρέπουν στους χρήστες να παρουσιάζουν και να πουλούν προϊόντα, να διαχειρίζονται το απόθεμα, να επεξεργάζονται πληρωμές και να χειρίζονται άλλες πτυχές του διαδικτυακού λιανικού εμπορίου. Αυτή η ενοποίηση παρέχει μια απρόσκοπτη εμπειρία, επιτρέποντας στις επιχειρήσεις

να διαχειρίζονται τόσο το περιεχόμενό τους όσο και τις λειτουργίες ηλεκτρονικού εμπορίου σε μια ενιαία πλατφόρμα. Απλοποιεί τη διαδικασία δημιουργίας και διατήρησης διαδικτυακής παρουσίας, συνδυάζοντας τη δημοσίευση περιεχομένου και τη διαχείριση προϊόντων

2.12.1Wordpress

Το WordPress είναι ένα ευρέως χρησιμοποιούμενο σύστημα διαχείρισης περιεχομένου ανοιχτού κώδικα (CMS) γνωστό για τη φιλική προς τον χρήστη διεπαφή και το εκτεταμένο οικοσύστημά του. Αρχικά σχεδιασμένο για blogging, το WordPress έχει εξελιχθεί σε μια ευέλικτη πλατφόρμα κατάλληλη για την κατασκευή διαφόρων τύπων ιστοσελίδων. Προσφέρει μια σειρά θεμάτων, προσθηκών και προσαρμόσιμων λειτουργιών, καθιστώντας το δημοφιλές μεταξύ ιδιωτών, bloggers και επιχειρήσεων για τη δημιουργία και τη διαχείριση της παρουσίας τους στο διαδίκτυο

2.12.2WooCommerce

Το WooCommerce είναι ένα ισχυρό πρόσθετο ηλεκτρονικού εμπορίου που έχει σχεδιαστεί για να ενσωματώνεται απρόσκοπτα με το WordPress, μετατρέποντάς το σε ένα πλήρως λειτουργικό ηλεκτρονικό κατάστημα. Αναπτύχθηκε από την Automattic, την εταιρεία πίσω από το WordPress.com, το WooCommerce φημίζεται για την απλότητα και την ευελιξία του. Επιτρέπει στους χρήστες να προσθέτουν καταχωρίσεις προϊόντων, να διαχειρίζονται το απόθεμα, να επεξεργάζονται πληρωμές και να χειρίζονται την αποστολή, όλα μέσα στο οικείο περιβάλλον του WordPress. Με μια τεράστια βιβλιοθήκη επεκτάσεων, το WooCommerce μπορεί να προσαρμοστεί για να ταιριάζει στις συγκεκριμένες ανάγκες διαφορετικών επιχειρήσεων, από μικρές νεοφυείς επιχειρήσεις έως μεγαλύτερες επιχειρήσεις.

2.12.3Magento

Το Magento είναι μια ισχυρή πλατφόρμα ηλεκτρονικού εμπορίου ανοιχτού κώδικα που απευθύνεται σε επιχειρήσεις όλων των μεγεθών. Γνωστό για την επεκτασιμότητα και την ευελιξία του, το Magento προσφέρει μια ολοκληρωμένη σειρά λειτουργιών για τη δημιουργία και τη διαχείριση ηλεκτρονικών καταστημάτων. Παρέχει ισχυρά εργαλεία για τη διαχείριση προϊόντων, τον έλεγχο του αποθέματος, την επεξεργασία παραγγελιών και ένα προσαρμόσιμο καλάθι αγορών. Με έμφαση στην απόδοση και την επεκτασιμότητα,

το Magento επιλέγεται συχνά από μεγαλύτερες επιχειρήσεις και επιχειρήσεις με πολύ-πλοκες απαιτήσεις ηλεκτρονικού εμπορίου

3 Εξέλιξη τών ηλεκτρονικών κατα-στημάτων

Ενδεικτική χρήση των στυλ επικεφαλίδας (Heading). Τα επίπεδα 4 και 5 μπορεί να μπουν εμβόλιμα σε οποιαδήποτε ενότητα μεγαλύτερου επιπέδου. Οι ενότητες εδώ έχουν λίγο κείμενο γιατί είναι παραδείγματα χρήσης. Μην φτιάχνετε συστηματικά μικρές ενότητες. Είναι δείγμα κακής οργάνωσης από μέρους σας.

3.1 Ιστορική Εξέλιξη των Ηλεκτρονικών Καταστημάτων

Η ιστορία του ηλεκτρονικού εμπορίου και των ηλεκτρονικών αγορών έχει υποστεί μια αξιοσημείωτη εξέλιξη, αναδιαμορφώνοντας τον τρόπο με τον οποίο οι άνθρωποι αγοράζουν και πωλούν αγαθά. Στη δεκαετία του 1970, τα πρώτα πειράματα στην ηλεκτρονική ανταλλαγή δεδομένων έθεσαν τις βάσεις για τις ηλεκτρονικές συναλλαγές. Η δεκαετία του 1980 έγινε μάρτυρας της εμφάνισης των ηλεκτρονικών αγορών, κυρίως στη σφαίρα των συναλλαγών B2B. Ωστόσο, μόλις το 1990, με τη δημιουργία του προγράμματος περιήγησης WorldWideWeb από τον Tim Berners-Lee, το διαδίκτυο έγινε προσβάσιμο στο κοινό, σηματοδοτώντας την πραγματική γέννηση του ηλεκτρονικού εμπορίου για τους καταναλωτές.

Το 1994 έγινε η πρώτη ασφαλής ηλεκτρονική αγορά, συμβάλλοντας στην εδραίωση της εμπιστοσύνης των καταναλωτών στις ηλεκτρονικές συναλλαγές. Στα μέσα της δεκαετίας του 1990 παρατηρήθηκε η άνοδος διαδικτυακών αγορών όπως το Amazon και το eBay, δίνοντας τη δυνατότητα σε ιδιώτες και επιχειρήσεις να συμμετέχουν σε εικονικές αγορές και πωλήσεις. Στα τέλη της δεκαετίας του 1990 γνώρισε την έκρηξη των dot-com, που οδήγησε σε άνοδο των startups του ηλεκτρονικού εμπορίου και στην καθιέρωση διαδικτυακών παρουσιών από παραδοσιακούς λιανοπωλητές.

Στις αρχές της δεκαετίας του 2000 είδε τη διεύρυνση των οριζόντων του ηλεκτρονικού εμπορίου, με μια ποικιλία προϊόντων να γίνεται διαθέσιμη για ηλεκτρονική αγορά. Στα μέσα της δεκαετίας του 2000, η έλευση των smartphones εγκαινίασε το κινητό εμπόριο (m-commerce), επιτρέποντας στους καταναλωτές να κάνουν αγορές χρησιμοποιώντας τις κινητές συσκευές τους. Η κυριαρχία των κολοσσών του ηλεκτρονικού εμπορίου όπως η

Amazon και η Alibaba έγινε εμφανής τη δεκαετία του 2010, οδηγώντας σε μείωση των παραδοσιακών καταστημάτων λιανικής πώλησης.

Σήμερα, το ηλεκτρονικό εμπόριο συνεχίζει να ευδοκιμεί, προσφέροντας διάφορα επιχειρηματικά μοντέλα, όπως συνδρομητικές υπηρεσίες, dropshipping και κοινωνικό εμπόριο. Καινοτομίες όπως η επαυξημένη πραγματικότητα (AR) για εικονικές δοκιμές και η τεχνητή νοημοσύνη (AI) για εξατομικευμένες προτάσεις έχουν διαμορφώσει περαιτέρω το τοπίο του ηλεκτρονικού εμπορίου. Η ιστορία των ηλεκτρονικών καταστημάτων αντικατοπτρίζει μια συνεχή εξέλιξη, με γνώμονα τις τεχνολογικές εξελίξεις, τις αλλαγές στη συμπεριφορά των καταναλωτών και την προσαρμογή των επιχειρήσεων στην ψηφιακή εποχή. Το ηλεκτρονικό εμπόριο έχει γίνει μια αναπόσπαστη και μεταμορφωτική δύναμη στο παγκόσμιο εμπόριο, παρέχοντας ευκολία και προσβασιμότητα στους καταναλωτές σε όλο τον κόσμο

3.2 Σημαντικά χαρακτηριστικά των ηλεκτρονικών καταστημάτων

Τα επιτυχημένα ηλεκτρονικά καταστήματα ενσωματώνουν μια σειρά από βασικές λειτουργίες για τη βελτιστοποίηση της εμπειρίας του χρήστη και τη διευκόλυνση των ασφαλών συναλλαγών. Μια φιλική προς το χρήστη διεπαφή με διαισθητική πλοήγηση και καθαρή διάταξη είναι θεμελιώδης, διασφαλίζοντας ότι οι πελάτες μπορούν να βρίσκουν εύκολα προϊόντα. Η ανταπόκριση σε κινητά είναι ζωτικής σημασίας για την εξυπηρέτηση των χρηστών σε διάφορες συσκευές.

Ο κατάλογος προϊόντων είναι ένα βασικό στοιχείο, με εικόνες υψηλής ποιότητας, λεπτομερείς περιγραφές και μια ισχυρή λειτουργία αναζήτησης για αποτελεσματική εξερεύνηση. Οι κριτικές και οι αξιολογήσεις πελατών συμβάλλουν στην κοινωνική απόδειξη, βοηθώντας τους πιθανούς αγοραστές. Ένα καλά σχεδιασμένο καλάθι αγορών επιτρέπει στους χρήστες να διαχειρίζονται τις επιλογές τους πριν προχωρήσουν σε μια ασφαλή διαδικασία ολοκλήρωσης αγοράς.

Η ασφάλεια είναι πρωταρχικής σημασίας σε όλη τη διαδικασία ολοκλήρωσης αγοράς, με κρυπτογράφηση SSL και υποστήριξη για πολλαπλές επιλογές πληρωμής, συμπεριλαμβανομένων πιστωτικών καρτών και ψηφιακών πορτοφολιών. Ένα σύστημα παρακολούθησης παραγγελιών ενισχύει τη διαφάνεια και την ικανοποίηση των πελατών.

Επιπλέον, οι λειτουργίες διαχείρισης λογαριασμού προσφέρουν στους χρήστες τη δυνατότητα να δημιουργούν προφίλ, να αποθηκεύουν προτιμήσεις και να παρακολουθούν το ιστορικό παραγγελιών. Αυτά τα συνδυασμένα χαρακτηριστικά δημιουργούν μια ολοκληρωμένη και απρόσκοπτη εμπειρία διαδικτυακών αγορών, ενισχύοντας την εμπιστοσύνη και την αφοσίωση των πελατών.

3.3 Η ποικιλομορφία των πλατφορμών ηλεκτρονικού εμπορίου

Στο συνεχώς εξελισσόμενο τοπίο του διαδικτυακού εμπορίου, η επιλογή της κατάλληλης πλατφόρμας ηλεκτρονικού εμπορίου παίζει καθοριστικό ρόλο στη διαμόρφωση της επιτυχίας και της αποτελεσματικότητας ενός ηλεκτρονικού καταστήματος. Με τα χρόνια, έχουν εμφανιστεί διάφορες πλατφόρμες, καθεμία από τις οποίες προσφέρει μοναδικά χαρακτηριστικά και δυνατότητες για να καλύψει τις διαφορετικές ανάγκες των διαδικτυακών επιχειρήσεων.

Μεταξύ της πληθώρας των διαθέσιμων πλατφορμών ηλεκτρονικού εμπορίου, η καθεμία έχει τα δυνατά της σημεία και είναι προσαρμοσμένη σε συγκεκριμένες απαιτήσεις. Μια τέτοια πλατφόρμα είναι το WooCommerce που ανάφερα παραπάνω , ενσωματωμένο άψογα με το WordPress, παρέχοντας ευελιξία στο σχεδιασμό και τη λειτουργικότητα.. Καθώς και το Magento, με τις ισχυρές επιλογές προσαρμογής της, στοχεύει σε επιχειρήσεις ηλεκτρονικού εμπορίου μεγάλης κλίμακας.

Αυτές οι πλατφόρμες χρησιμεύουν ως η αρχή των διαδικτυακών επιχειρήσεων, προσφέροντας χαρακτηριστικά όπως φιλική προς τον χρήστη διεπαφή, ανταπόκριση σε κινητά και ασφαλείς επιλογές πληρωμής. Ένας καλά οργανωμένος κατάλογος προϊόντων, η αποτελεσματική λειτουργικότητα αναζήτησης και μια ασφαλής διαδικασία ολοκλήρωσης αγοράς συμβάλλουν σε μια θετική εμπειρία χρήστη. Επιπλέον, λειτουργίες όπως οι κριτικές πελατών και η παρακολούθηση παραγγελιών ενισχύουν τη διαφάνεια και χτίζουν εμπιστοσύνη μεταξύ των καταναλωτών.

Καθώς αναζητούμε τρόπους ανάπτυξης ιστοσελίδων ηλεκτρονικού εμπορίου, είναι σημαντικό να αναφέρουμε ένα απο τα ποιό γνώστα php framework εμπορίου με ένα ισχυρό πλαίσιο υποστήριξης. Το Laravel, ένα πλαίσιο PHP γνωστό για την κομψότητα και την απλότητά του . Η αξιοποίηση της ισχύος του Laravel στο backend επιτρέπει την απρόσκοπτη διαχείριση δεδομένων, την ασφαλή επεξεργασία συναλλαγών και τον αποτελεσματικό έλεγχο ταυτότητας χρήστη. Η Laravel φέρει ένα ολοκληρωμένο σύνολο

εργαλείων και λειτουργιών που συμπληρώνουν τέλεια τις δυνατότητες frontend δημοφιλών πλατφορμών ηλεκτρονικού εμπορίου. Η αρχιτεκτονική MVC, το ORM και η χειροτεχνική διεπαφή γραμμής εντολών βελτιστοποιούν τη διαδικασία ανάπτυξης. Η ευελιξία του Laravel επιτρέπει στους προγραμματιστές να το ενσωματώσουν με διάφορες βάσεις δεδομένων, διασφαλίζοντας επεκτασιμότητα και απόδοση.

4 Laravel php Framework

4.1 Laravel ως PHP framework

Το Laravel, ένα σημαντικό πλαίσιο εφαρμογών ιστού PHP, έχει αναδειχθεί ως φάρος στη σφαίρα της σύγχρονης ανάπτυξης Ιστού. Κατασκευασμένο από τον Taylor Otwell, η Laravel έκανε το ντεμπούτο της τον Ιούνιο του 2011 με αποστολή να επαναπροσδιορίσει την εμπειρία ανάπτυξης, δίνοντας έμφαση στην απλότητα, την κομψότητα και τη χαρά του προγραμματιστή.

4.2 Ιστορικό και γενική περιγραφή των χαρακτηριστικών που το καθιστούν δημοφιλές.

Από την έναρξή του, η Laravel έχει υποστεί μια αξιοσημείωτη εξέλιξη. Το ταξίδι ξεκίνησε με το Laravel 1, θέτοντας τις βάσεις για την ανάπτυξη διαδικτυακών εφαρμογών. Οι επόμενες εκδόσεις εισήγαγαν βασικά χαρακτηριστικά όπως το Eloquent ORM, τη μηχανή προτύπων Blade και την ενσωμάτωση του Composer, ενισχύοντας τη θέση της Laravel ως προοδευτικό και με επίκεντρο τους προγραμματιστές πλαίσιο.

Το Laravel 4, μια σημαντική αναμόρφωση, εισήγαγε μια εκλεπτυσμένη αρχιτεκτονική και τη διεπαφή γραμμής εντολών Artisan που βασίζεται σε Composer. Το Laravel 5, με τη σημασιολογική του εκδοχή, έφερε το Laravel Elixir για βελτιωμένη ανάπτυξη frontend. Το Laravel 6 παρουσίασε το Laravel Vapor για ανάπτυξη χωρίς διακομιστή, ενώ το Laravel 8 παρουσίασε το Laravel Jetstream, εργοστασιακές κατηγορίες μοντέλων και δυναμικά στοιχεία Blade.

Σε όλη τη διάρκεια της ιστορίας της, η Laravel όχι μόνο συμβαδίζει με τις σύγχρονες πρακτικές ανάπτυξης PHP, αλλά έχει πρωτοπορήσει και σε αρκετές καινοτομίες. Η εκφραστική σύνταξη του, σε συνδυασμό με χαρακτηριστικά όπως το Laravel Mix, οι εντολές Artisan και το δυναμικό πρότυπο Blade, έχει προσελκύσει μια ισχυρή κοινότητα προγραμματιστών.

Καθώς η Laravel συνεχίζει να εξελίσσεται, παραμένει μια απόδειξη της συγχώνευσης κομψότητας και λειτουργικότητας, προσφέροντας ένα δυναμικό οικοσύστημα που εξουσιοδοτεί τους προγραμματιστές να δημιουργούν εξελιγμένες διαδικτυακές εφαρμογές με

απαράμιλλη ευκολία. Αυτή η αφήγηση αποτυπώνει την ουσία της Laravel – ένα πλαίσιο που όχι μόνο διαμορφώνει το παρόν αλλά συνεχίζει να εμπνέει το μέλλον της ανάπτυξης της PHP

4.3 Model-View-Controller (MVC)

Η αρχιτεκτονική Model-View-Controller (MVC) είναι ένας τρόπος σχεδιασμού που χρησιμοποιείται ευρέως στην ανάπτυξη λογισμικού και η Laravel, ως πλαίσιο MVC, αξιοποιεί αυτό το μοτίβο για την δομή των εφαρμογών.

4.3.1 Model

Το Μοντέλο αντιπροσωπεύει τα δεδομένα και την επιχειρηματική λογική της εφαρμογής. Ενσωματώνει τις λειτουργίες που σχετίζονται με δεδομένα, όπως η αναζήτηση στη βάση δεδομένων, η επικύρωση δεδομένων και η εφαρμογή επιχειρηματικών κανόνων.

Περιέχει δομές δεδομένων που αντιπροσωπεύουν οντότητες ή αντικείμενα στην εφαρμογή.

Αλληλεπιδρά με τη βάση δεδομένων μέσω ενός συστήματος αντικειμενικής-σχεσιακής χαρτογράφησης (ORM) (π.χ. Eloquent στο Laravel) και διαχειρίζεται την επικύρωση δεδομένων, τους επιχειρηματικούς κανόνες και τις σχέσεις μεταξύ οντοτήτων.

4.3.2 View

Το View είναι υπεύθυνο για την παρουσίαση δεδομένων στον χρήστη. Αφορά τη λογική της παρουσίασης και τα στοιχεία διεπαφής χρήστη.

Αντιπροσωπεύει τις οπτικές πτυχές της εφαρμογής, συμπεριλαμβανομένων των HTML, CSS και JavaScript, καθώς εμφανίζει τα δεδομένα από το μοντέλο στον χρήστη καθώς και διατηρεί τη διεπαφή χρήστη ξεχωριστή από τη λογική της εφαρμογής.

4.3.3 Controller

Ο Ελεγκτής ενεργεί ως ενδιάμεσος μεταξύ του Model και του View. Λαμβάνει τα στοιχεία του χρήστη, τα επεξεργάζεται (ενδεχομένως αλληλεπιδρώντας με το Μοντέλο) και

ενημερώνει ανάλογα επίσης χειρίζεται τα αιτήματα και τις εισροές χρηστών καθώς και δίνει τη ροή δεδομένων μεταξύ του Μοντέλου και του View .

4.4 Πλεονεκτήματα αρχιτεκτονικής MVC

Πλεονεκτήματα της αρχιτεκτονικής MVC περιλαμβάνουν:

Separation of Concerns: Όπως αναφέρθηκε η αρχή αυτή επιτρέπει τον διαχωρισμό της λογικής της εφαρμογής σε τρεις διακριτικές στρώσεις (Μοντέλο, Προβολή, Ελεγκτής)

Reusability: Η διάκριση των στρωμάτων επιτρέπει την επαναχρησιμοποίηση των στοιχείων σε διάφορα μέρη της εφαρμογής ή ακόμη και σε άλλα έργα. Αυτό προωθεί έναν κώδικα που είναι πιο ευέλικτος και ευανάγνωστος.

Testabilit: Κάθε στρώμα του MVC μπορεί να υποβληθεί σε δοκιμές ανεξάρτητα, επιτρέποντας την ευκολότερη ανάπτυξη μονάδων δοκιμών. Αυτό διασφαλίζει τη σταθερότητα και την αποτελεσματικότητα του κώδικα.

Flexibility: Ο διαχωρισμός των λειτουργιών επιτρέπει ευκολότερες αλλαγές ή ενημερώσεις σε ένα στρώμα χωρίς να επηρεάζονται τα υπόλοιπα. Για παράδειγμα, μια αλλαγή στην εμφάνιση της εφαρμογής (Προβολή)

4.5 Project Structure in Laravel

4.5.1 App Directory

Ο κατάλογος εφαρμογών βρίσκεται στην καρδιά μιας εφαρμογής Laravel. Περιέχει τον βασικό κώδικα και είναι οργανωμένο σε υποκαταλόγους:

Http: Ελεγκτές, ενδιάμεσο λογισμικό και αιτήματα φορμών.

Providers:: Πάροχοι υπηρεσιών που εκκινούν διάφορα στοιχεία της εφαρμογής.

Console: Εντολές Artisan.

4.5.2 Bootstrap Directory

Ο κατάλογος bootstrap περιέχει το αρχείο app.php, το οποίο προετοιμάζει την εφαρμογή Laravel. Αυτός ο κατάλογος φιλοξενεί επίσης τον κατάλογο προσωρινής μνήμης για αρχεία πλαισίου που έχουν αποθηκευτεί στην κρυφή μνήμη.

4.5.3 Config Directory

Τα αρχεία διαμόρφωσης για διάφορες πτυχές της εφαρμογής αποθηκεύονται στον κατάλογο διαμόρφωσης. Οι προγραμματιστές μπορούν να προσαρμόσουν ρυθμίσεις όπως συνδέσεις βάσης δεδομένων, διαμορφώσεις κρυφής μνήμης και άλλα.

4.5.4 Database Directory

Ο κατάλογος της βάσης δεδομένων περιέχει μετεγκαταστάσεις και σπόρους για τη διαχείριση της βάσης δεδομένων. Οι μετεγκαταστάσεις χρησιμοποιούνται για τον έλεγχο της έκδοσης των αλλαγών σχήματος βάσης δεδομένων, ενώ τα προγράμματα seeders συμπληρώνουν τη βάση δεδομένων με αρχικά δεδομένα.

4.5.5 Public Directory

Ο public κατάλογος είναι η ρίζα εγγράφων του διακομιστή web. Περιέχει το αρχείο σημείου εισόδου index.php και χρησιμεύει ως τοποθεσία για στοιχεία όπως εικόνες, stylesheets και αρχεία JavaScript. Ο κατάλογος αυτός φιλοξενεί views, language files και πρωτογενή στοιχεία (πριν από τη μεταγλώττιση). Τα πρότυπα Blade της Laravel αποθηκεύονται στον υποκατάλογο προβολών.

4.5.6 Routes Directory

Ο κατάλογος διαδρομών είναι εκεί όπου ορίζονται τα routes . Web routes ορίζονται συνήθως στο web.php και τα API routes στο api.php. Οι διαδρομές παρέχουν μια αντιστοίχιση μεταξύ URI και Controllers:

4.5.7 Storage Directory

Η Laravel χρησιμοποιεί τον κατάλογο αποθήκευσης για να αποθηκεύει αρχεία που δημιουργούνται, όπως αρχεία καταγραφής, αρχεία που δημιουργούνται από πλαίσιο και μεταφορτωμένα αρχεία χρήστη. Ο υποκατάλογος εφαρμογών εντός του χώρου αποθήκευσης περιέχει αρχεία που θα πρέπει να διατηρηθούν μεταξύ των αναπτύξεων.

4.5.8 Vendor Directory

Ο κατάλογος προμηθευτή περιέχει εξαρτήσεις Composer. Δημιουργείται από το Composer με βάση το αρχείο composer.json.

4.5.9 Env File

Το αρχείο .env περιλαμβάνει ρυθμίσεις οσον αφορά την βάση δεδομένων, application key,και environment type.

4.5.10 Config Files

Τα αρχεία διαμόρφωσης της Laravel βρίσκονται στον κατάλογο ρυθμίσεων. Οι προγραμματιστές μπορούν να τροποποιήσουν αυτά τα αρχεία για να προσαρμόσουν τη συμπεριφορά διαφόρων στοιχείων Laravel. Η κατανόηση και η τήρηση αυτής της δομής του έργου διευκολύνει την οργάνωση κώδικα

4.6 PHP ARTISAN

Το Artisan είναι η διεπαφή γραμμής εντολών που περιλαμβάνεται στο Laravel, παρέχοντας στους προγραμματιστές μια σειρά από χρήσιμες εντολές για κοινές εργασίες και εργασίες ειδικά για τη Laravel. Απλοποιεί διάφορες πτυχές της ανάπτυξης, προσφέροντας δυνατότητες για δημιουργία κώδικα, μετεγκατάσταση βάσεων δεδομένων, δοκιμές και άλλα

Artisan Commands: Οι εντολές Artisan καλύπτουν ένα ευρύ φάσμα λειτουργιών.

Database Migrations: Το Artisan απλοποιεί τις μετεγκαταστάσεις βάσεων δεδομένων, επιτρέποντας στους προγραμματιστές να ορίζουν αλλαγές της βάσης δεδομένων και να τις εφαρμόζουν εύκολα

Database Seeding: Οι προγραμματιστές μπορούν να χρησιμοποιήσουν το Artisan για τη δημιουργία βάσεων δεδομένων με δοκιμαστικά ή προεπιλεγμένα δεδομένα

Artisan Serve: Η εντολή ’php artisan serve’ ξεκινά έναν διακομιστή ανάπτυξης, επιτρέποντας στους προγραμματιστές να εκτελούν την εφαρμογή τοπικά για δοκιμή και εντοπισμό σφαλμάτων

Testing: Το Artisan διευκολύνει τις δοκιμές παρέχοντας εντολές για την εκτέλεση δοκιμών. Η εντολή δοκιμής php artisan εκτελεί τη δοκιμαστική σουίτα, διασφαλίζοντας ότι η εφαρμογή συμπεριφέρεται όπως αναμένεται.

4.7 Security and Laravel

Η Laravel είναι γνωστή για τη δέσμευσή της στην ασφάλεια και διαθέτει πολλές ενσωματωμένες λειτουργίες που βοηθούν τους προγραμματιστές να δημιουργήσουν ασφαλείς εφαρμογές.

4.7.1 CSRF(Cross-Site Request Forgery)

Η Laravel περιλαμβάνει προστασία πλαστογράφησης αιτημάτων μεταξύ ιστότοπων (CSRF) δημιουργώντας μοναδικά διακριτικά CSRF για κάθε περίοδο λειτουργίας ενεργού χρήστη. Αυτά τα διακριτικά περιλαμβάνονται αυτόματα σε φόρμες και η Laravel τα επαληθεύει σε κάθε εισερχόμενο αίτημα POST.

4.7.2 XSS Protection

Laravel διαφεύγει αυτόματα τις μεταβλητές που μεταβιβάζονται στις προβολές, γεγονός που βοηθά στην προστασία από επιθέσεις Cross-Site Scripting (XSS). Επιπλέον, η μηχανή προτύπων Blade παρέχει οδηγίες για την έξοδο ακατέργαστου HTML όταν χρειάζεται

4.7.3 SQL Injection Protection:

Το Eloquent ORM και το Query Builder της Laravel παραμετροποιούν αυτόματα τα ερωτήματα, αποτρέποντας επιθέσεις SQL injection. Οι προγραμματιστές ενθαρρύνονται να χρησιμοποιούν αυτές τις μεθόδους πρόσβασης στη βάση δεδομένων αντί για απλά ερωτήματα SQL

4.7.4 Authentication and Authorization:

Το ενσωματωμένο σύστημα αυθεντικοποίησης της Laravel παρέχει ασφαλείς δυνατότητες σύνδεσης και εγγραφής χρήστη. Η εξουσιοδότηση μπορεί να διεκπεραιωθεί χρησιμοποιώντας πολιτικές και πύλες, επιτρέποντας λεπτομερή έλεγχο των αδειών των χρηστών.

Η Laravel χρησιμοποιεί ενάν αλγόριθμο bcrypt για να κρυπτογράφηση με ασφάλεια τους κωδικούς πρόσβασης χρηστών. Χρησιμοποιείται η συνάρτηση password_hash της PHP, και οι κωδικοί πρόσβασης κρυπτογραφίζονται αυτόματα πριν αποθηκευτούν στη βάση δεδομένων

4.7.5 Middleware

Το Middleware στο Laravel επιτρέπει στους προγραμματιστές να φιλτράρουν αιτήματα HTTP που εισέρχονται στην εφαρμογή. Η Laravel περιλαμβάνει ενδιάμεσο λογισμικό για την επαλήθευση του ελέγχου ταυτότητας του χρήστη, αποτρέποντας τη μη εξουσιοδοτημένη πρόσβαση σε διαδρομές.

4.7.6 Route Protection

Οι προγραμματιστές μπορούν να προστατεύουν διαδρομές χρησιμοποιώντας ενδιάμεσο λογισμικό, διασφαλίζοντας ότι μόνο οι πιστοποιημένοι χρήστες ή χρήστες με συγκεκριμένους ρόλους μπορούν να έχουν πρόσβαση σε ορισμένα μέρη της εφαρμογής

4.7.7 HTTPS

Η Laravel υποστηρίζει ασφαλή επικοινωνία μέσω HTTPS. Κατά την ανάπτυξη μιας εφαρμογής, συνιστάται η επιβολή HTTPS για όλη την επισκεψιμότητα ιστού.

4.8 Jetstream and LiveWire

Το Jetstream είναι ένα πακέτο που παρέχεται από την Laravel, είναι η βάση για τη δημιουργία συστημάτων ελέγχου πρόσβασης και εξουσιοδότησης χρηστών σε μια εφαρμογή Laravel.

Το Livewire επιτρέπει στους προγραμματιστές να δημιουργούν δυναμικές εφαρμογές, χωρίς την ανάγκη για ένα εκτεταμένο προγραμματισμό με JavaScript.

Το Livewire βοηθά στη διαχείριση των δυναμικών στοιχείων της εφαρμογής σας, ενώ το Jetstream παρέχει ένα έτοιμο σύστημα εξουσιοδότησης και άλλες λειτουργίες που μπορείτε να ενσωματώσετε στην εφαρμογή σας.

4.8.1 User Authentication

Είναι ένα σημαντικό στοιχείο σε πολλές εφαρμογές, καθώς επιτρέπει στους χρήστες να αυθεντικοποιούνται προτού αποκτήσουν πρόσβαση σε προστατευμένο περιεχόμενο ή λειτουργίες.

Στο πλαίσιο του Laravel, το Jetstream παρέχει ένα έτοιμο σύστημα εξουσιοδότησης χρηστών. Αυτό περιλαμβάνει λειτουργίες όπως:

Εγγραφή χρηστών: Οι χρήστες μπορούν να δημιουργήσουν λογαριασμούς με την παροχή στοιχείων όπως όνομα, email και κωδικό πρόσβασης.
Σύνδεση χρηστών: Οι χρήστες μπορούν να συνδεθούν στο σύστημα με τα διαπιστευτήριά τους (email και κωδικό πρόσβασης).
Ανάκτηση κωδικού πρόσβασης: Υπάρχουν μηχανισμοί για την ανάκτηση ξεχασμένων κωδικών πρόσβασης μέσω email ή άλλων μεθόδων επαλήθευσης.
Επίπεδα πρόσβασης: Μπορούν να οριστούν διάφορα επίπεδα πρόσβασης για τους χρήστες, όπως π.χ. "admin" και "user", με διαφορετικά δικαιώματα πρόσβασης σε λειτουργίες της εφαρμογής.

Το Jetstream χρησιμοποιεί την πλατφόρμα Blade του Laravel για τη δημιουργία των απαραίτητων προβολών (views) και παρέχει επίσης έτοιμους controllers και μοντέλα για τη διαχείριση της εξουσιοδότησης.

5 Δημιουργία Ηλεκτρονικού Καταστήματος με Χρήση του Laravel

5.1 Εισαγωγή στο eshop

Η παρούσα ενότητα επικεντρώνεται στη δημιουργία ενός ηλεκτρονικού καταστήματος μέσω της εφαρμογής του Laravel.

Στόχος αυτής της ενότητας είναι η κατανόηση των διαδικασιών ανάπτυξης ενός πλήρως λειτουργικού ηλεκτρονικού καταστήματος όπως αναφέραμε Θα γίνει ανάλυση στις βασικές λειτουργίες και τα χαρακτηριστικά που προσφέρει το Laravel για την ανάπτυξη τέτοιων εφαρμογών.

Ανάμεσα στα κύρια σημεία περιλαμβάνονται οι διαδικασίες εγκατάστασης του Laravel, η διαχείριση των προϊόντων και η διαχείριση των παραγγελιών τόσο από την πλευρά του χρήστη όσο και από την πλευρά του διαχειριστή. Επίσης, η ασφάλεια και η προστασία των δεδομένων των χρηστών και η διαδικασία πληρωμής.

5.1.1 Χρήστης και διαχειριστής (Admin – User)

Πλευρά Χρήστη

Συγκεκριμένα, θα υπάρχει η αρχική σελίδα χρησιμοποιώντας ένα πρότυπο για αυτό το εγχείρημα. Ο χρήστης θα μπορεί να δει τα προϊόντα και να τα προσθέσει στο καλάθι αγορών του. Για να προσθέσει το προϊόν στο καλάθι, ο χρήστης θα πρέπει να συνδεθεί. Στη συνέχεια, θα υπάρχει επιλογή σύνδεσης και εγγραφής στο σύστημα. Τέλος, θα υπάρχει επιλογή πληρωμής με PayPal ή Stripe, που περιλαμβάνει πληρωμή με Visa, MasterCard, πιστωτική ή χρεωστική κάρτα όπου χρήστης θα λαμβάνει επίσης επιβεβαίωση μέσω email κατά την ολοκλήρωση της αγοράς.

Πλευρά Διαχειριστή

Από την πλευρά του διαχειριστή, θα υπάρχει ένας πίνακας ελέγχου όπου ο διαχειριστής θα μπορεί να προσθέσει, να διαγράψει ή να ενημερώσει προϊόντα, να δει τις παραγγελίες, να τις ακυρώσει και να στείλει ειδοποιήσεις μέσω email στους πελάτες.

Πριν ξεκινήσει η εγκατάσταση της Laravel θα πρέπει να προσθεθούν 2 σημαντικά εργαλεία το Node.js και Composer (PhP)

Τόσο το Node.js όσο και το Composer είναι σημαντικά εργαλεία για τη δημιουργία ενός ηλεκτρονικού καταστήματος χρησιμοποιώντας το Laravel.

Node.js: Το Node.js είναι ένα περιβάλλον εκτέλεσης JavaScript που χρησιμοποιείται για την εκτέλεση JavaScript κώδικα στον διακομιστή. Στο πλαίσιο του έργου σας, το Node.js θα χρησιμοποιηθεί για την εκτέλεση του npm (Node Package Manager), το οποίο είναι ένα εργαλείο για τη διαχείριση εξαρτήσεων και πακέτων JavaScript που χρησιμοποιούνται στον πελάτη (π.χ. πρότυπα, βιβλιοθήκες JavaScript)

Composer: Το Composer είναι ένα εργαλείο διαχείρισης εξαρτήσεων για τη γλώσσα προγραμματισμού PHP. Στο πλαίσιο του έργου σας, το Composer θα χρησιμοποιηθεί για τη διαχείριση των εξαρτήσεων PHP του Laravel, όπως πακέτα, βιβλιοθήκες και πλαίσια που απαιτούνται για τη λειτουργία του καταστήματος.

Και τα δύο εργαλεία είναι ουσιώδη για τη δημιουργία και τη λειτουργία ενός σύγχρονου ηλεκτρονικού καταστήματος με χρήση του Laravel, καθώς εξασφαλίζουν τη διαχείριση των εξαρτήσεων και τη σωστή λειτουργία του έργου

5.2 Δημιουργία Laravel Project

Δημιουργούμε το project laravel με την εξής εντολή :

```
composer create-project laravel/laravel example-app
```

Αφού ολοκληρωθεί η εγκατάσταση χρησιμοποιείτε η εντολή **php artisan serve**

Η εντολή αυτήν εκκινεί έναν τοπικό διακομιστή ανάπτυξης για την εφαρμογή σας. Αυτός ο διακομιστής επιτρέπει την προεπισκόπηση και τον έλεγχο της εφαρμογής σε ένα τοπικό περιβάλλον πριν την ανάπτυξη σε έναν πραγματικό διακομιστή .

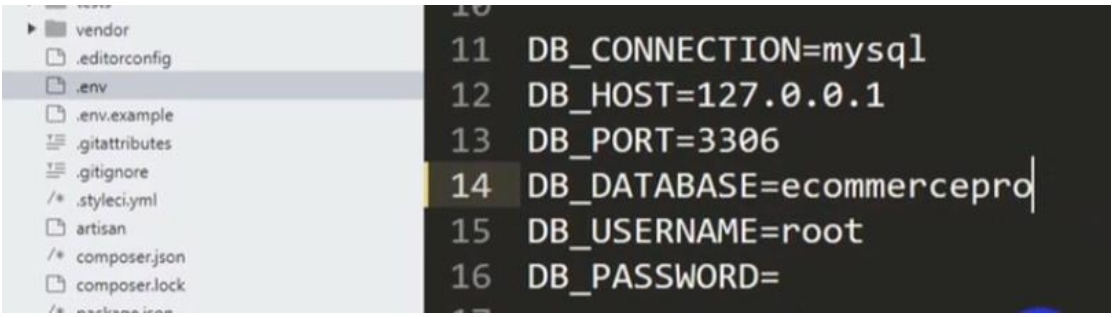
Στο επόμενο βήμα, θα δημιουργηθει η βάση δεδομένων όπου θα αποθηκεύονται όλα τα προϊόντα του ηλεκτρονικού καταστήματος, οι χρήστες, οι πελάτες, οι κατηγορίες και άλλες σχετικές πληροφορίες.

Η βάση δεδομένων αποτελεί τον πυρήνα του ηλεκτρονικού καταστήματος, καθώς εδώ αποθηκεύονται όλα τα δεδομένα που απαιτούνται για τη λειτουργία της εφαρμογής. Τα προϊόντα του καταστήματος, όπως περιγραφή, τιμή και διαθεσιμότητα, αποθηκεύονται εδώ, καθώς και πληροφορίες σχετικά με τους πελάτες και τους χρήστες που έχουν εγγραφεί στο σύστημα επιπλέον, οι κατηγορίες προϊόντων οργανώνονται επίσης στη βάση δεδομένων, επιτρέποντας στους χρήστες να περιηγούνται εύκολα στον κατάλογο των προϊόντων ανά κατηγορία.

5.2.1 Δημιουργία βάσης δεδομένων E Commerce

Η δημιουργία αυτής της βάσης δεδομένων είναι κρίσιμη για την επιτυχή λειτουργία του ηλεκτρονικού καταστήματος και θα μας επιτρέψει να διαχειριστούμε αποτελεσματικά τα προϊόντα, τους πελάτες και τις πωλήσεις.

Εγκατάσταση Βάσης Δεδομένων: Αφού ορίσαμε το όνομα της βάσης δεδομένων στο αρχείο .env ως "E Commerce Pro", δημιουργούμε τη βάση δεδομένων χρησιμοποιώντας τον PHPMyAdmin. Εκεί δημιουργούμε μια νέα βάση δεδομένων με το ίδιο όνομα και κλικάρουμε το "create". Αυτό θα δημιουργήσει τη βάση δεδομένων μας με το σωστό όνομα.



Άνοιγμα του Έργου στο vscode: Μετά την εγκατάσταση της βάσης δεδομένων, ανοίγουμε το έργο μας στο vscode. Από εκεί, μπορούμε να επεξεργαστούμε τα αρχεία του έργου και να συνεχίσουμε την ανάπτυξή μας.

5.2.2 Login και Register για User και Admin

Εντολές που χρησιμοποιήθηκαν :

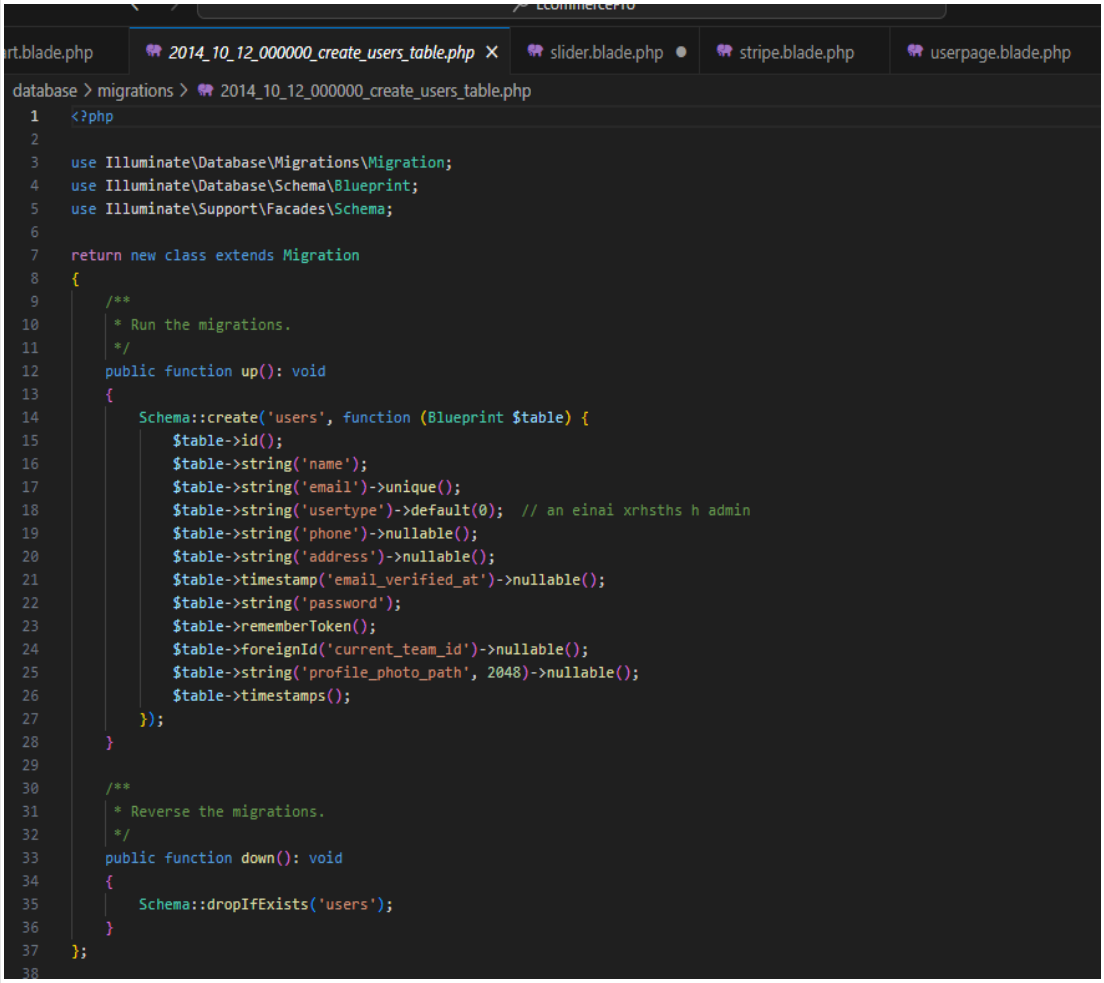
- Composer require Jetstream:install livewire
- php artisan Jetstream:install livewire
- npm install
- npm run dev

Αλλαγές σε user table στην βαση δεδομένων :

- php artisan migrate

Γίνεται χρήση των παραπάνω εντολών για την εγκατάσταση των πακέτων του Laravel Jetstream, που παρέχουν τη δυνατότητα πολυχρηστικής σύνδεσης. Μετά την εγκατάσταση αυτών των πακέτων, θα εγκαταστήσουμε το LiveWire για τη διαχείριση των διεπαφών χρήστη.

Στο επόμενο βήμα, θα γίνει επεξεργασία του πίνακα χρηστών στη βάση δεδομένων μας. Θα προσθέσουμε τρία νέα πεδία: "user_type", "phone" και "address", που θα χρησιμοποιηθούν για τη διάκριση μεταξύ διαχειριστών και χρηστών και την αποθήκευση πρόσθετων πληροφοριών.

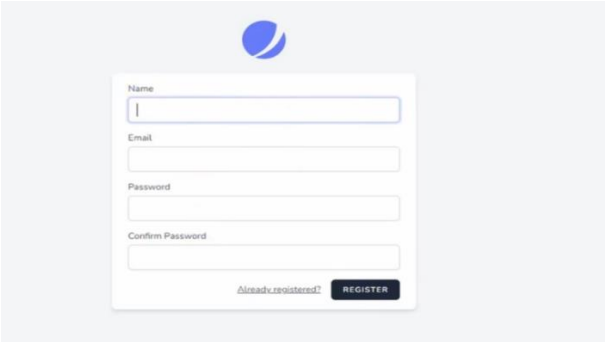


Τέλος, θα εκτελέσουμε την εντολή "php artisan migrate" στον terminal του vs code για να μεταφέρουμε τις αλλαγές στη βάση δεδομένων μας. Αυτό θα ενεργοποιήσει το πολυχρηστικό σύστημα σύνδεσης στο έργο μας.

Με αυτόν τον τρόπο, όταν ανανεώθεί η σελίδα, θα εμφανιστούν επιλογές για σύνδεση και εγγραφή. Με βάση την σύνδεση θα διαχωριστεί ο admin από τους απλούς χρήστες παρέχοντας τις κατάλληλες δυνατότητες πρόσβασης ανάλογα με το ρόλο η αλλιώς τύπο του κάθε χρήστη.

Ο τύπος χρήστη είναι εξ' ορισμού μηδέν. (database , user_type = 0)

Παρατηρούμε πλέον στο σίτε τα buttons σύνδεσης και εγγραφής.Αν πηγαίνετε στη σελίδα εγγραφής υπάρχουν τα πεδία ονόματος, ηλεκτρονικού ταχυδρομείου και κωδικού πρόσβασης .



Ωστόσο, στη βάση δεδομένων μας έχουμε επίσης τα πεδία τηλεφώνου και διεύθυνσης. Έτσι, πρέπει να δημιουργηθούν τα πεδία τηλεφώνου και διεύθυνσης για την εγγραφή. Στον φάκελο "resources", στον οποίο βρίσκονται οι πηγές, και μέσα σε αυτόν υπάρχει ένας φάκελος με το όνομα "views". Εντός αυτού, θα υπάρχει ένας φάκελος που ονομάζεται "auth".Εντός αυτού του φακέλου υπάρχει το αρχείο με το όνομα "register.blade.php".Εκεί, βρίσκονται τα πεδία ονόματος, ηλεκτρονικού ταχυδρομείου ,κωδικό πρόσβασης και επιβεβαίωση κωδικού πρόσβασης. Εδώ θα προσθεθεί το "ηλεκτρονικό ταχυδρομείο" και το "τηλέφωνο"ως προς το view του χρήστη . Αντίστοιχα στον φάκελο "app" μέσα στον φάκελο "models", στο αρχείο "User.php". Γίνεται πρόσθεση των πεδίων "τηλέφωνο" και "διεύθυνση" ως προς την λειτουργικότητα .

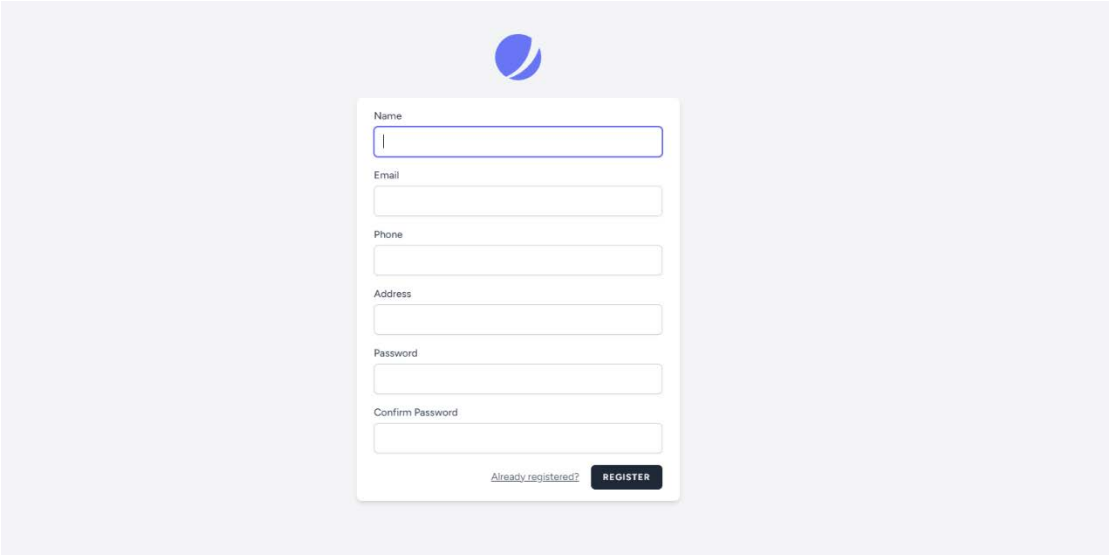
```php
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Fortify\TwoFactorAuthenticatable;
use Laravel\Jetstream\HasProfilePhoto;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable implements MustVerifyEmail
{
    use HasApiTokens;
    use HasFactory;
    use HasProfilePhoto;
    use Notifiable;
    use TwoFactorAuthenticatable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'phone',
        'address',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
        'two_factor_recovery_codes',
        'two_factor_secret',
    ];
```

showcart.blade.php | register.blade.php | slider.blade.php | stripe.blade.php | userpage.blade.php | sidebar.blade.php | home.blade.ph

views > auth > register.blade.php

```blade
<x-validation-errors class="mb-4" />

<form method="POST" action="{{ route('register') }}">
    @csrf

    <div>
        <x-label for="name" value="{{ __('Name') }}" />
        <x-input id="name" class="block mt-1 w-full" type="text" name="name" :value="old('name')" required autofocus autocomplete="name" />
    </div>

    <div class="mt-4">
        <x-label for="email" value="{{ __('Email') }}" />
        <x-input id="email" class="block mt-1 w-full" type="text" name="email" :value="old('email')" required autocomplete="username" />
    </div>


    <div class="mt-4">
        <x-label for="phone" value="{{ __('Phone') }}" />
        <x-input id="phone" class="block mt-1 w-full" type="text" name="phone" :value="old('phone')" required />
    </div>

    <div class="mt-4">
        <x-label for="address" value="{{ __('Address') }}" />
        <x-input id="address" class="block mt-1 w-full" type="text" name="address" :value="old('address')" required />
    </div>

    <div class="mt-4">
        <x-label for="password" value="{{ __('Password') }}" />
        <x-input id="password" class="block mt-1 w-full" type="password" name="password" required autocomplete="new-password" />
    </div>

    <div class="mt-4">
        <x-label for="password_confirmation" value="{{ __('Confirm Password') }}" />
        <x-input id="password_confirmation" class="block mt-1 w-full" type="password" name="password_confirmation" required autocomplete="new-password" />
    </div>
```

Δημιουργία ενός νέου χρήστη



Στο κλικ κουμπί "Εγγραφή", θα δημιουργηθεί ο χρήστης στη βάση δεδομένων με τα στοιχεία που δώθηκαν.

Εάν αποσυνδεθούμε από το σύστημα και δημιουργήσουμε έναν ακόμα χρήστη, όπου αυτός θα είναι ο διαχειριστής. Μετά από τη δημιουργία του λογαριασμού, θα χρειαστεί να αλλαχτεί ο τύπος χρήστη σε "1", προκειμένου να τον καθορίσουμε ως διαχειριστή.(Στην βάση δεδομένων επιλογή του χρήστη που θα είναι ο *διαχειριστής και αλλαγή του user_type = 1* έτσι θα διαφέρει ο ρόλος του από τους άλλους χρήστες).Έτσι διασφαλίζεται ότι εάν ο διαχειριστής προσπαθήσει να συνδεθεί, θα μεταφερθεί σε ένα διαφορετικό πίνακα ελέγχου(άλλο view - άλλη σελίδα) , διαφορετικό από αυτόν του κανονικού χρήστη.

Η παρούσα ενότητα επικεντρώνεται στη δημιουργία ενός ηλεκτρονικού καταστήματος μέσω της εφαρμογής του Laravel.

Στόχος αυτής της ενότητας είναι η κατανόηση των διαδικασιών ανάπτυξης ενός πλήρως λειτουργικού ηλεκτρονικού καταστήματος όπως αναφέραμε Θα γίνει ανάλυση στις βασικές λειτουργίες και τα χαρακτηριστικά που προσφέρει το Laravel για την ανάπτυξη τέτοιων εφαρμογών.

Ανάμεσα στα κύρια σημεία περιλαμβάνονται οι διαδικασίες εγκατάστασης του Laravel, η διαχείριση των προϊόντων και η διαχείριση των παραγγελιών τόσο από την πλευρά του χρήστη όσο και από την πλευρά του διαχειριστή. Επίσης, η ασφάλεια και η προστασία των δεδομένων των χρηστών και η διαδικασία πληρωμής.

Πλευρά Χρήστη

Συγκεκριμένα, θα υπάρχει η αρχική σελίδα χρησιμοποιώντας ένα πρότυπο για αυτό το εγχείρημα. Ο χρήστης θα μπορεί να δει τα προϊόντα και να τα προσθέσει στο καλάθι αγορών του. Για να προσθέσει το προϊόν στο καλάθι, ο χρήστης θα πρέπει να συνδεθεί. Στη συνέχεια, θα υπάρχει επιλογή σύνδεσης και εγγραφής στο σύστημα. Τέλος, θα υπάρχει επιλογή πληρωμής με PayPal ή Stripe, που περιλαμβάνει πληρωμή με Visa, MasterCard, πιστωτική ή χρεωστική κάρτα όπου χρήστης θα λαμβάνει επίσης επιβεβαίωση μέσω email κατά την ολοκλήρωση της αγοράς.

Πλευρά Διαχειριστή

Από την πλευρά του διαχειριστή, θα υπάρχει ένας πίνακας ελέγχου όπου ο διαχειριστής θα μπορεί να προσθέσει, να διαγράψει ή να ενημερώσει προϊόντα, να δει τις παραγγελίες, να τις ακυρώσει και να στείλει ειδοποιήσεις μέσω email στους πελάτες.

Πριν ξεκινήσει η εγκατάσταση της Laravel θα πρέπει να προσθεθούν 2 σημαντικά εργαλεία το Node.js και Composer (PhP)

Τόσο το Node.js όσο και το Composer είναι σημαντικά εργαλεία για τη δημιουργία ενός ηλεκτρονικού καταστήματος χρησιμοποιώντας το Laravel.

Node.js: Το Node.js είναι ένα περιβάλλον εκτέλεσης JavaScript που χρησιμοποιείται για την εκτέλεση JavaScript κώδικα στον διακομιστή. Στο πλαίσιο του έργου σας, το Node.js θα χρησιμοποιηθεί για την εκτέλεση του npm (Node Package Manager), το οποίο είναι ένα εργαλείο για τη διαχείριση εξαρτήσεων και πακέτων JavaScript που χρησιμοποιούνται στον πελάτη (π.χ. πρότυπα, βιβλιοθήκες JavaScript)

Composer: Το Composer είναι ένα εργαλείο διαχείρισης εξαρτήσεων για τη γλώσσα προγραμματισμού PHP. Στο πλαίσιο του έργου σας, το Composer θα χρησιμοποιηθεί για τη διαχείριση των εξαρτήσεων PHP του Laravel, όπως πακέτα, βιβλιοθήκες και πλαίσια που απαιτούνται για τη λειτουργία του καταστήματος.

Και τα δύο εργαλεία είναι ουσιώδη για τη δημιουργία και τη λειτουργία ενός σύγχρονου ηλεκτρονικού καταστήματος με χρήση του Laravel, καθώς εξασφαλίζουν τη διαχείριση των εξαρτήσεων και τη σωστή λειτουργία του έργου

Δημιουργία Laravel Project
Δημιουργούμε το project laravel με την εξής εντολή :

```
composer create-project laravel/laravel example-app
```

Αφού ολοκληρωθεί η εγκατάσταση χρησιμοποιείτε η εντολή **php artisan serve**

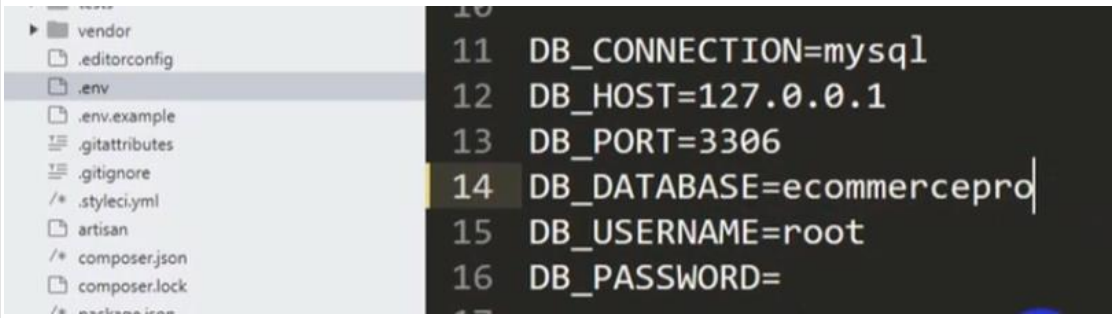
Η εντολή αυτήν εκκινεί έναν τοπικό διακομιστή ανάπτυξης για την εφαρμογή σας. Αυτός ο διακομιστής επιτρέπει την προεπισκόπηση και τον έλεγχο της εφαρμογής σε ένα τοπικό περιβάλλον πριν την ανάπτυξη σε έναν πραγματικό διακομιστή .

Στο επόμενο βήμα, θα δημιουργηθει η βάση δεδομένων όπου θα αποθηκεύονται όλα τα προϊόντα του ηλεκτρονικού καταστήματος, οι χρήστες, οι πελάτες, οι κατηγορίες και άλλες σχετικές πληροφορίες.

Η βάση δεδομένων αποτελεί τον πυρήνα του ηλεκτρονικού καταστήματος, καθώς εδώ αποθηκεύονται όλα τα δεδομένα που απαιτούνται για τη λειτουργία της εφαρμογής. Τα προϊόντα του καταστήματος, όπως περιγραφή, τιμή και διαθεσιμότητα, αποθηκεύονται εδώ, καθώς και πληροφορίες σχετικά με τους πελάτες και τους χρήστες που έχουν εγγραφεί στο σύστημα επιπλέον, οι κατηγορίες προϊόντων οργανώνονται επίσης στη βάση δεδομένων, επιτρέποντας στους χρήστες να περιηγούνται εύκολα στον κατάλογο των προϊόντων ανά κατηγορία.

Η δημιουργία αυτής της βάσης δεδομένων είναι κρίσιμη για την επιτυχή λειτουργία του ηλεκτρονικού καταστήματος και θα μας επιτρέψει να διαχειριστούμε αποτελεσματικά τα προϊόντα, τους πελάτες και τις πωλήσεις.

Εγκατάσταση Βάσης Δεδομένων: Αφού ορίσαμε το όνομα της βάσης δεδομένων στο αρχείο .env ως "E Commerce Pro", δημιουργούμε τη βάση δεδομένων χρησιμοποιώντας τον PHPMyAdmin. Εκεί δημιουργούμε μια νέα βάση δεδομένων με το ίδιο όνομα και κλικάρουμε το "create". Αυτό θα δημιουργήσει τη βάση δεδομένων μας με το σωστό όνομα.



Άνοιγμα του Έργου στο vscode: Μετά την εγκατάσταση της βάσης δεδομένων, ανοίγουμε το έργο μας στο vscode. Από εκεί, μπορούμε να επεξεργαστούμε τα αρχεία του έργου και να συνεχίσουμε την ανάπτυξή μας.

Login και Register για User και Admin

Εντολές που χρησιμοποιήθηκαν :

- Composer require Jetstream:install livewire
- php artisan Jetstream:install livewire
- npm install

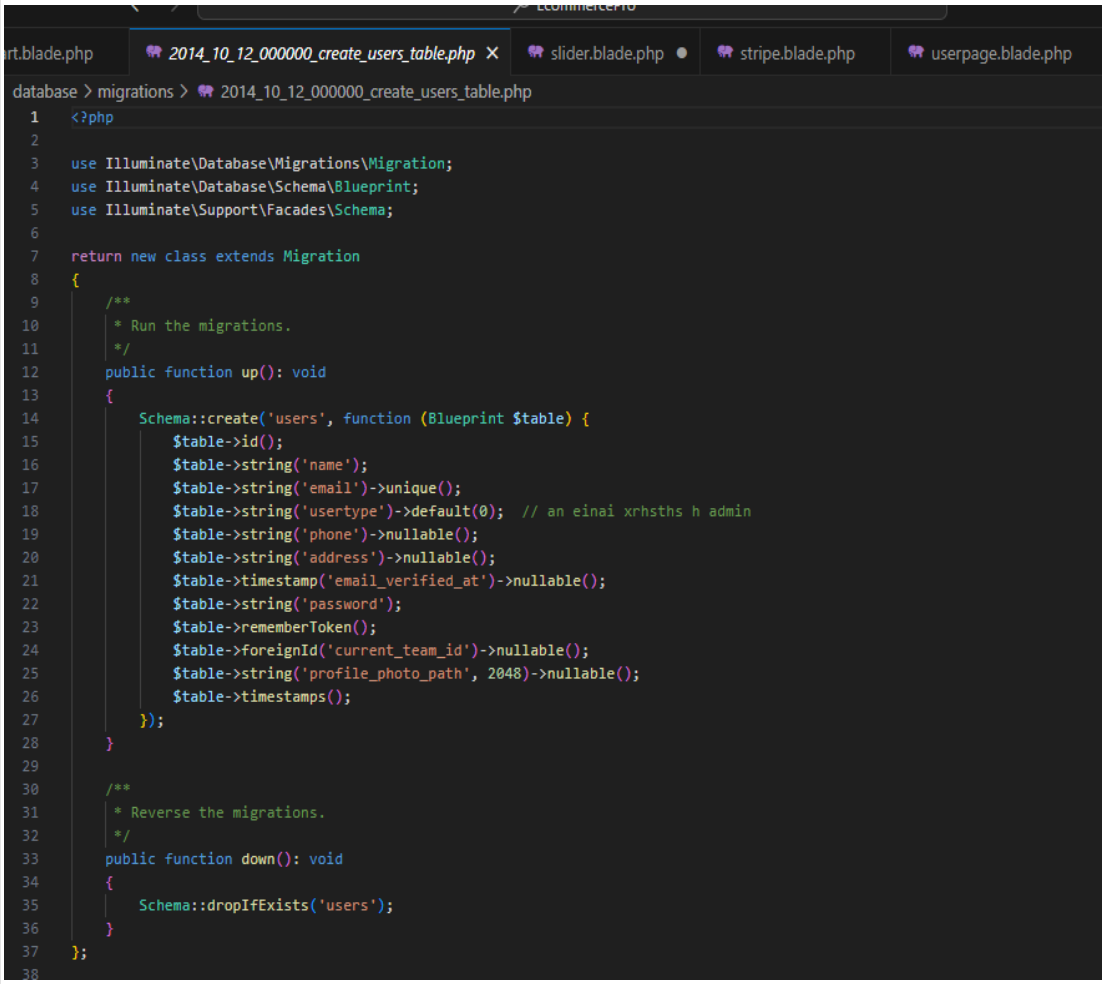
- npm run dev

Αλλαγές σε user table στην βαση δεδομένων :

- php artisan migrate

Γίνεται χρήση των παραπάνω εντολών για την εγκατάσταση των πακέτων του Laravel Jetstream, που παρέχουν τη δυνατότητα πολυχρηστικής σύνδεσης. Μετά την εγκατάσταση αυτών των πακέτων, θα εγκαταστήσουμε το LiveWire για τη διαχείριση των διεπαφών χρήστη.

Στο επόμενο βήμα, θα γίνει επεξεργασία του πίνακα χρηστών στη βάση δεδομένων μας. Θα προσθέσουμε τρία νέα πεδία: "user_type", "phone" και "address", που θα χρησιμοποιηθούν για τη διάκριση μεταξύ διαχειριστών και χρηστών και την αποθήκευση πρόσθετων πληροφοριών.



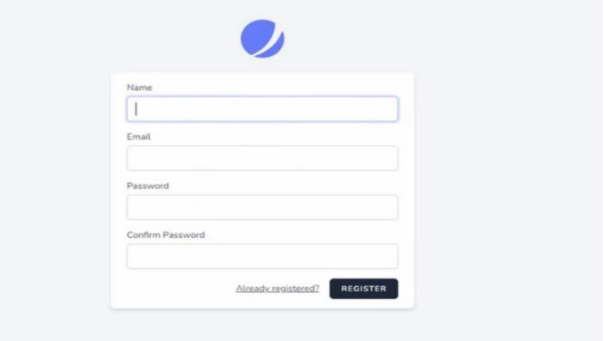
Τέλος, θα εκτελέσουμε την εντολή "php artisan migrate" στον terminal του vs code για να μεταφέρουμε τις αλλαγές στη βάση δεδομένων μας. Αυτό θα ενεργοποιήσει το πολυχρηστικό σύστημα σύνδεσης στο έργο μας.

Με αυτόν τον τρόπο, όταν ανανεώθεί η σελίδα, θα εμφανιστούν επιλογές για σύνδεση και εγγραφή. Με βάση την σύνδεση θα διαχωριστεί ο admin από τους απλούς χρήστες

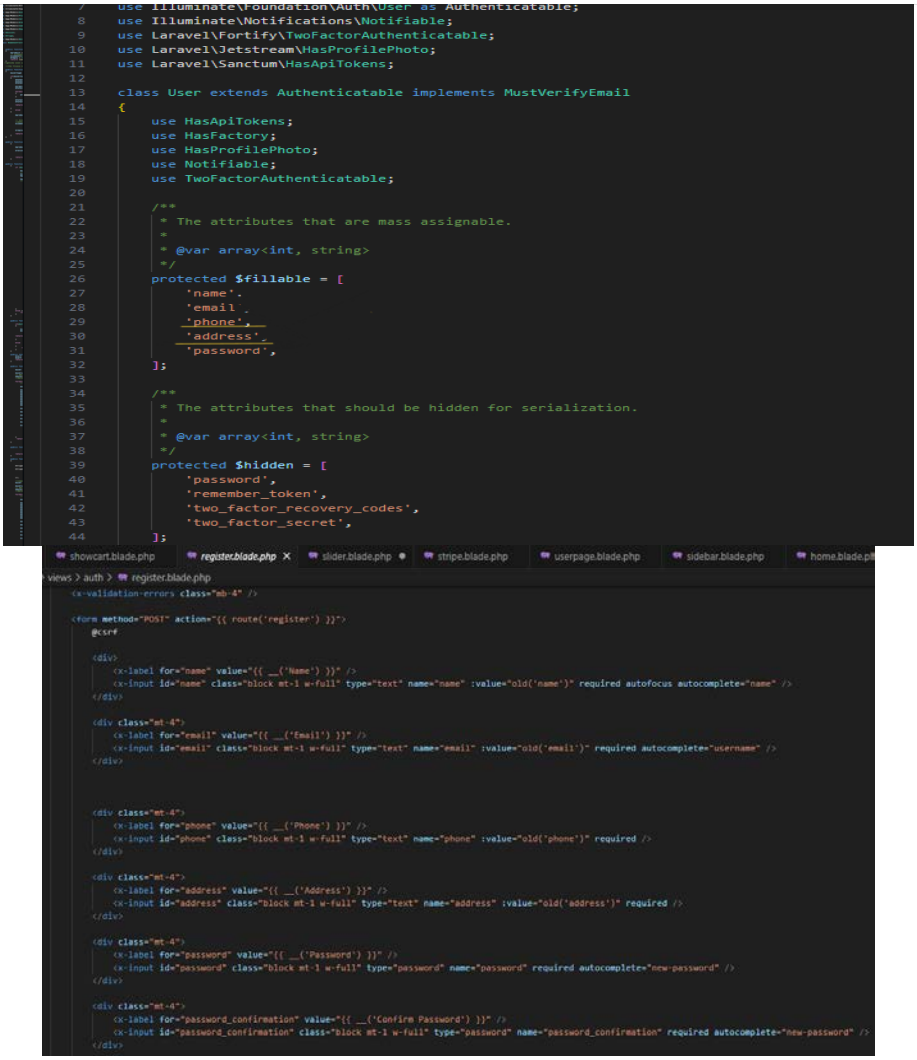
παρέχοντας τις κατάλληλες δυνατότητες πρόσβασης ανάλογα με το ρόλο η αλλιώς τύπο του κάθε χρήστη.

Ο τύπος χρήστη είναι εξ' ορισμού μηδέν. (database , user_type = 0)

Παρατηρούμε πλέον στο σίτε τα buttons σύνδεσης και εγγραφής.Αν πηγαίνετε στη σελίδα εγγραφής υπάρχουν τα πεδία ονόματος, ηλεκτρονικού ταχυδρομείου και κωδικού πρόσβασης .



Ωστόσο, στη βάση δεδομένων μας έχουμε επίσης τα πεδία τηλεφώνου και διεύθυνσης. Έτσι, πρέπει να δημιουργηθούν τα πεδία τηλεφώνου και διεύθυνσης για την εγγραφή. Στον φάκελο "resources", στον οποίο βρίσκονται οι πηγές, και μέσα σε αυτόν υπάρχει ένας φάκελος με το όνομα "views". Εντός αυτού, θα υπάρχει ένας φάκελος που ονομάζεται "auth".Εντός αυτού του φακέλου υπάρχει το αρχείο με το όνομα "register.blade.php".Εκεί, βρίσκονται τα πεδία ονόματος, ηλεκτρονικού ταχυδρομείου ,κωδικό πρόσβασης και επιβεβαίωση κωδικού πρόσβασης. Εδώ θα προσθεθεί το "ηλεκτρονικό ταχυδρομείο" και το "τηλέφωνο"ως προς το view του χρήστη . Αντίστοιχα στον φάκελο "app" μέσα στον φάκελο "models", στο αρχείο "User.php". Γίνεται πρόσθεση των πεδίων "τηλέφωνο" και "διεύθυνση" ως προς την λειτουργικότητα .



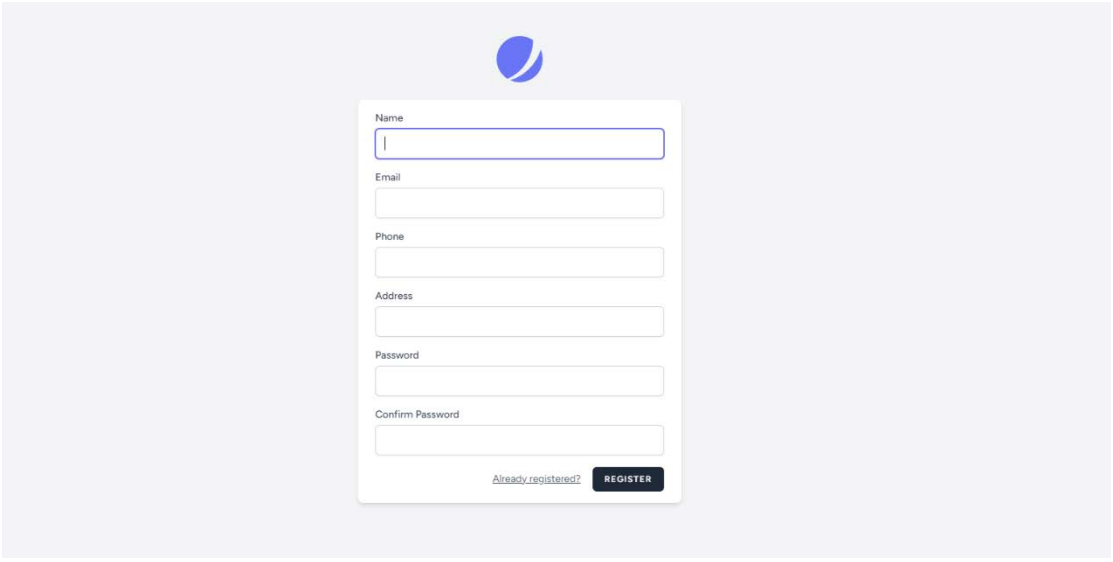
```php
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Fortify\TwoFactorAuthenticatable;
use Laravel\Jetstream\HasProfilePhoto;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable implements MustVerifyEmail
{
    use HasApiTokens;
    use HasFactory;
    use HasProfilePhoto;
    use Notifiable;
    use TwoFactorAuthenticatable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'phone',
        'address',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
        'two_factor_recovery_codes',
        'two_factor_secret',
    ];
```

showcart.blade.php | register.blade.php | slider.blade.php | stripe.blade.php | userpage.blade.php | sidebar.blade.php | home.blade.php

views > auth > register.blade.php

```blade
<x-validation-errors class="mb-4" />

<form method="POST" action="{{ route('register') }}">
    @csrf

    <div>
        <x-label for="name" value="{{ __('Name') }}" />
        <x-input id="name" class="block mt-1 w-full" type="text" name="name" :value="old('name')" required autofocus autocomplete="name" />
    </div>

    <div class="mt-4">
        <x-label for="email" value="{{ __('Email') }}" />
        <x-input id="email" class="block mt-1 w-full" type="text" name="email" :value="old('email')" required autocomplete="username" />
    </div>



    <div class="mt-4">
        <x-label for="phone" value="{{ __('Phone') }}" />
        <x-input id="phone" class="block mt-1 w-full" type="text" name="phone" :value="old('phone')" required />
    </div>

    <div class="mt-4">
        <x-label for="address" value="{{ __('Address') }}" />
        <x-input id="address" class="block mt-1 w-full" type="text" name="address" :value="old('address')" required />
    </div>

    <div class="mt-4">
        <x-label for="password" value="{{ __('Password') }}" />
        <x-input id="password" class="block mt-1 w-full" type="password" name="password" required autocomplete="new-password" />
    </div>

    <div class="mt-4">
        <x-label for="password_confirmation" value="{{ __('Confirm Password') }}" />
        <x-input id="password_confirmation" class="block mt-1 w-full" type="password" name="password_confirmation" required autocomplete="new-password" />
    </div>
```

Δημιουργία ενός νέου χρήστη



Στο κλικ κουμπί "Εγγραφή", θα δημιουργηθεί ο χρήστης στη βάση δεδομένων με τα στοιχεία που δώθηκαν.

Εάν αποσυνδεθούμε από το σύστημα και δημιουργήσουμε έναν ακόμα χρήστη, όπου αυτός θα είναι ο διαχειριστής. Μετά από τη δημιουργία του λογαριασμού, θα χρειαστεί να αλλαχτεί ο τύπος χρήστη σε "1", προκειμένου να τον καθορίσουμε ως διαχειριστή.(Στην βάση δεδομένων επιλογή του χρήστη που θα είναι ο *διαχειριστής και αλλαγή του user_type = 1* έτσι θα διαφέρει ο ρόλος του από τους άλλους χρήστες).Έτσι διασφαλίζεται ότι εάν ο διαχειριστής προσπαθήσει να συνδεθεί, θα μεταφερθεί σε ένα διαφορετικό πίνακα ελέγχου(άλλο view - άλλη σελίδα) , διαφορετικό από αυτόν του κανονικού χρήστη.

5.3 Admin Side (Πλευρά Διαχειριστή)

Για να ενσωματώσουμε το πρότυπο διαχειριστή στον πίνακα ελέγχου διαχειριστή της Laravel, ξεκινάμε με τη λήψη του προτύπου "Corona"(από web) . Μετά τη λήψη, αντιγράφουμε όλα τα αρχεία και τους φακέλους, εξαιρουμένου του package.json, στον δημόσιο φάκελο του έργου μας, δημιουργώντας έναν φάκελο διαχειριστή εάν χρειάζεται. Το αρχείο index.html μέσα στο πρότυπο ενσωματώνεται στη συνέχεια στο αρχείο προβολής διαχειριστή. Εξασφαλίζουμε τη σωστή απόδοση προσαρμόζοντας τις διαδρομές στοιχείων ώστε να αντικατοπτρίζουν τη νέα δομή καταλόγου. Οργανώνοντας τον κώδικα, δημιουργούμε ξεχωριστά αρχεία για κεφαλίδα, πλαϊνή γραμμή, σώμα, CSS και JavaScript. Αυτά τα αρχεία αποθηκεύονται στο φάκελο με τις προβολές διαχειριστή για ευκολία συντήρησης. Μετά την οργάνωση του κώδικα, διασφαλίζουμε ότι όλες οι διαδρομές εικόνας αναφέρονται στο φάκελο διαχειριστή για να αποφύγουμε κατεστραμμένους συνδέσμους .Περαιτέρω βελτιώσεις μπορούν να γίνουν με την προσθήκη λειτουργιών όπως η λειτουργικότητα αποσύνδεσης για τη βελτίωση της εμπειρίας χρήστη. Αυτό περιλαμβάνει τη δημιουργία μιας επιλογής αποσύνδεσης στην κεφαλίδα διαχειριστή και την ενσωμάτωσή της στον πίνακα ελέγχου διαχειριστή. Στην συνέχεια θα προστεθούν πρόσθετες λειτουργίες όπως κατηγορίες προϊόντων και διαχείριση παραγγελιών για περαιτέρω βελτίωση της λειτουργικότητας του πίνακα ελέγχου διαχειριστή. Με την ολοκλήρωση αυτών των βημάτων, ο πίνακας ελέγχου διαχειριστή της Laravel έχει πλέον ενσωματωθεί με επιτυχία με το επιλεγμένο πρότυπο διαχειριστή.

5.3.1 Σχεδιασμός Category Page για τον διαχειριστή

Στο sidebar του πίνακα ελέγχου διαχειριστή, προστέθηκε μια επιλογή category οπου εκεί θα προσθέτουμε την κατηγορία του προιόντος πχ toy

Το αρχείο sidebar.blade.php που βρίσκεται στον φάκελο διαχειριστή της βάσης κωδίκων μας ενημερώθηκε. Το υπάρχον στοιχείο φόρμας αντικαταστάθηκε με μια επιλογή κατηγορίας. Μετά την αποθήκευση των αλλαγών, το πρόγραμμα περιήγησης ανανεώθηκε για να εμφανιστεί η επιλογή κατηγορίας στο sidebar του admin.

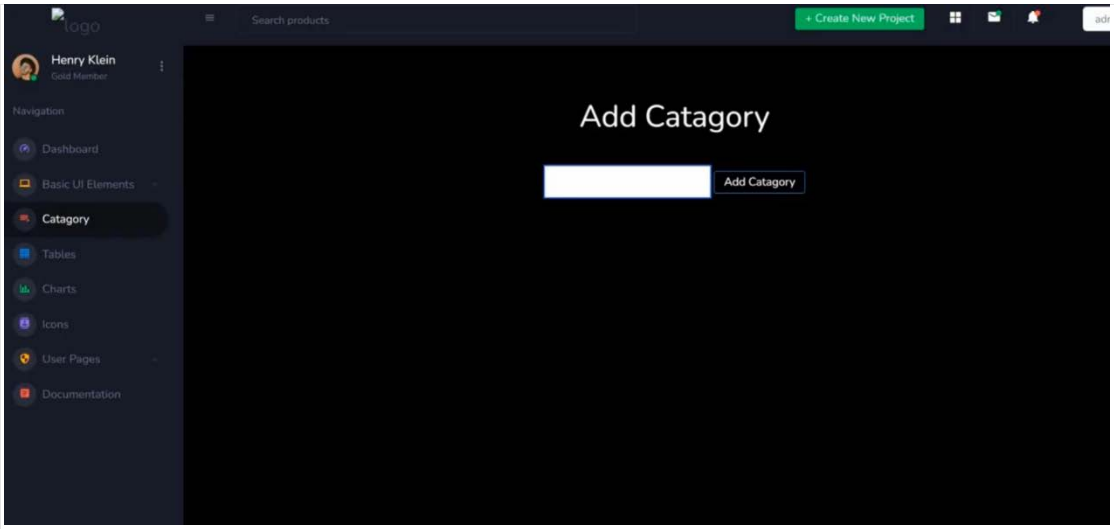
Κάνοντας κλικ στην επιλογή κατηγορία τώρα ανακατευθύνεται σε άλλη σελίδα. Ένα view με το όνομα category.blade.php δημιουργήθηκε για την εμφάνιση κατηγοριών. Η διαδρομή για τη σελίδα της κατηγορίας ορίστηκε στο αρχείο web.php.Δημιουργήθηκε ένας AdminController και ορίστηκε μια μέθοδος με το όνομα queue_category για τη

διαχείριση αιτημάτων που σχετίζονται με κατηγορίες. Ένα αρχείο προβολής category.blade.php δημιουργήθηκε στον φάκελο διαχειριστή.Τα περιεχόμενα της προβολής πίνακα ελέγχου διαχειριστή αντιγράφηκαν στο view category . Τα περιττά μέρη αφαιρέθηκαν από το view category.Η απαραίτητη δομή HTML διατηρήθηκε για σωστή απόδοση.Προστέθηκαν πεδία εισαγωγής για την προσθήκη νέων κατηγοριών.Το πεδίο εισαγωγής ήταν στοίχιση στο κέντρο και το μέγεθος της γραμματοσειράς για την επικεφαλίδα "Add Category".Συμπεριλήφθηκε μια φόρμα με ένα πεδίο εισαγωγής και ένα κουμπί υποβολής για την προσθήκη κατηγοριών.Τα στυλ CSS εφαρμόστηκαν για να αλλάξει το χρώμα του κειμένου του πεδίου εισαγωγής σε μαύρο.

5.3.2 Διαγραφή Κατηγορίας προιόντος

Προσθήκη ενός κουμπιού διαγραφής δίπλα σε κάθε κατηγορία, τη μετάδοση του αναγνωριστικού κατηγορίας στη λειτουργία διαγραφής με το πάτημα του κουμπιού και τη ρύθμιση διαδρομών και λειτουργιών ελεγκτή για τη διαχείριση του αιτήματος διαγραφής. Χρησιμοποιώντας το category model, τα δεδομένα της αντίστοιχης κατηγορίας ανακτήθηκαν και διαγράφηκαν από τη βάση δεδομένων. Επισής προστέθηκε ένα μήνυμα επιβεβαίωσης διαγραφών και μετά την επιτυχή διαγραφή, εμφανίζεται ένα μήνυμα επιτυχίας. Αυτή η συστηματική προσέγγιση διασφαλίζει ότι οι διαχειριστές μπορούν να διαχειρίζονται εύκολα τις κατηγορίες διαγράφοντας τες όπως απαιτείται, με διασφαλίσεις για την αποφυγή σφαλμάτων.

Προσθήκη Κατηγορίας ,Διαγραφη Κατηγοριας, Προβολη Κατηγορίας

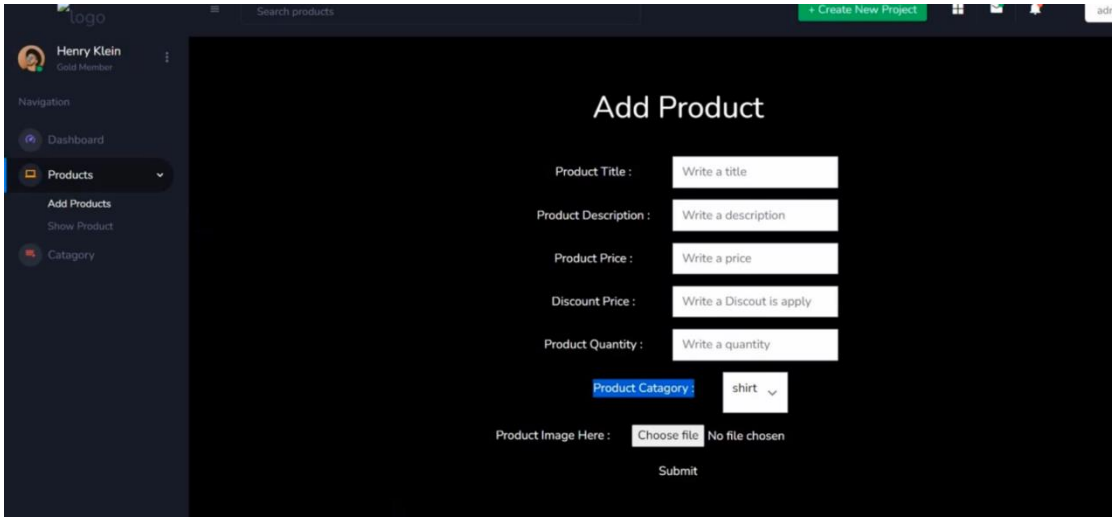


5.3.3 Διαχείριση δεδομένων προϊόντος από τον admin.

Δημιουργία μιας επιλογής στο sidebar του admin page για τα products . Αυτήν επιλογή ονομάστηκε add products . Κάνοντας κλικ στην επιλογή "Προσθήκη προϊόντων" κατευθύνονται ο διαχειριστής σε μια σελίδα όπου μπορούν να εισάγουν λεπτομέρειες προϊόντος, διευκολύνοντας την προσθήκη νέων προϊόντων στη βάση δεδομένων. Στη συνέχεια δημιουργήθηκε ένας πίνακας προϊόντων στη βάση δεδομένων, ο οποίος ορίζει στήλες για τίτλο, περιγραφή, εικόνα, τιμή, ποσότητα, κατηγορία και τιμή έκπτωσης. Τέλος, εκτελέστηκε η εντολή migration για την επιτυχή εφαρμογή του πίνακα προϊόντων στη βάση δεδομένων. Αυτή η απλοποιημένη διαδικασία ενισχύει τη χρηστικότητα του πίνακα διαχείρισης, επιτρέποντας την αποτελεσματική διαχείριση των δεδομένων του προϊόντος.

descriptions, images, categories, quantities, prices, and discount prices συμπεριλήφθηκαν σε αυτήν την ενότητα με κομψούς τίτλους. Κάνοντας κλικ στην επιλογή "Add Product" μεταβαίνετε σε μια σελίδα με πεδία εισαγωγής για τη μεταφόρτωση δεδομένων στον

product table . Δημιουργήθηκε μια φόρμα HTML στον πίνακα διαχείρισης με πεδία εισαγωγής για κάθε χαρακτηριστικό προϊόντος, όπως τίτλος, περιγραφή, τιμή, ποσότητα, τιμή έκπτωσης, κατηγορία και εικόνα. Η φόρμα σχεδιάστηκε με χρήση CSS για να βελτιωθεί η εμφάνιση και η χρηστικότητά της, με στοιχεία ευθυγραμμισμένα και τοποθετημένα κατάλληλα. Η Input validation εφαρμόστηκε με την επισήμανση ορισμένων πεδίων ως required διασφαλίζοντας ότι θα γίνει η εισαγωγή τους . Τέλος, προστέθηκε ένα κουμπί υποβολής στη φόρμα για να μπορέσει ο διαχειριστής να ανεβάσει αποτελεσματικά τις πληροφορίες του προϊόντος που θέλει να προσθέσει στην σελίδα του.



Στη συνέχεια, κατά το κλικ στην επιλογή "Προσθήκη Προϊόντος", θα παρατηρήσετε ότι τα πεδία της φόρμας είναι έτοιμα για εισαγωγή. Η φόρμα έχει διαμορφωθεί με τα κατάλληλα γνωρίσματα δράσης (action), μεθόδου (method) και τύπου (enctype) για να χειριστεί τις μεταφορές αρχείων και την υποβολή δεδομένων. Επιπλέον, ένα token CSRF έχει συμπεριληφθεί για να εξασφαλιστεί η ασφάλεια.

Για να γεμίσει το αναπτυσσόμενο μενού κατηγορίας προϊόντος, τα δεδομένα από τον πίνακα κατηγορίας στη βάση δεδομένων ανακτώνται και εμφανίζονται δυναμικά. Αυτό εξασφαλίζει ότι οι διαχειριστές μπορούν να επιλέξουν την κατάλληλη κατηγορία για το προϊόν που προσθέτουν. Τα ονόματα των κατηγοριών ανακτώνται από τη βάση δεδομένων και εμφανίζονται στο αναπτυσσόμενο μενού για επιλογή.

Επιπλέον, ένας βρόχος επανάληψης προχωράει μέσω των δεδομένων κατηγορίας ανακτημένων από τη βάση δεδομένων, γεμίζοντας το αναπτυσσόμενο μενού με επιλογές που αντιπροσωπεύουν κάθε κατηγορία. Αυτή η δυναμική γέμιση του αναπτυσσόμενου μενού

βελτιώνει την εμπειρία του χρήστη και εξασφαλίζει ότι οι διαχειριστές μπορούν εύκολα να επιλέξουν την επιθυμητή κατηγορία για το προϊόν που προσθέτουν.Επιπλέον, η επιλεγμένη κατηγορία εμφανίζεται στο αναπτυσσόμενο μενού, παρέχοντας οπτική ανατροφοδότηση στον διαχειριστή. Αυτό εξασφαλίζει σαφήνεια και βοηθά στην αποτροπή σφαλμάτων κατά την εισαγωγή δεδομένων. Στην συνέχεια γίνεται η προσθήκη των προϊόντων στην βάση δεδομένων.

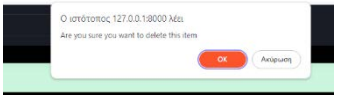
Εμφάνιση όλων των προϊόντων που προστέθηκαν μέσω διαχειριστή στον πίνακα προϊόντων. Κάνοντας κλικ στην επιλογή "show product " στο sidebar ο διαχεριιστής μεταφέρετε σε άλλη σελίδα όπου παρουσιάζονται όλα τα δεδομένα από τον πίνακα προϊόντων της βάσης δέδοεμεων . Η κρίσιμη πτυχή εδώ έγκειται στη ρύθμιση της δρομολόγησης URL για την επιλογή "show product" στην πλαϊνή γραμμή, η οποία πλοηγεί τους χρήστες στην κατάλληλη σελίδα για να προβάλουν τα δεδομένα του προϊόντος. Επιπλέον, η διασφάλιση ότι τα δεδομένα προϊόντος ανακτώνται αποτελεσματικά και εμφανίζονται στον πίνακα ελέγχου διαχειριστή είναι απαραίτητη για τη διευκόλυνση της εύκολης πρόσβασης και διαχείρισης των προϊόντων εντός της εφαρμογής

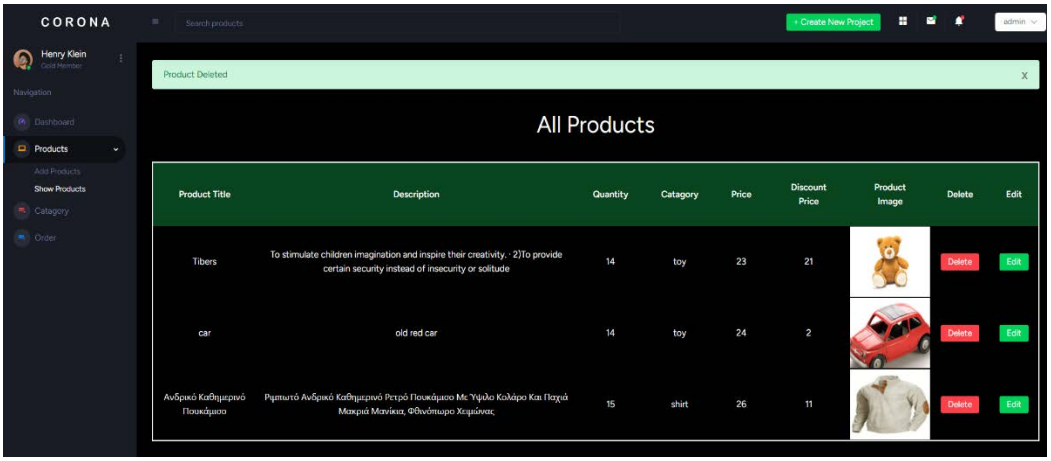
5.3.4 Delete και update product από τον admin.

Κουμπί Διαγραφής Προιόντος(Delete)

Εδώ , δημιουργήθηκαν κουμπιά επεξεργασίας(χωρίς χρήση ακόμη) και διαγραφής για κάθε προϊόν σε μια πίνακα που εμφανίζει όλα τα προϊόντα που προστέθηκαν στη βάση δεδομένων μέσω του πίνακα διαχειριστή. Το κάθε προϊόν έχει κουμπί επεξεργασίας και διαγραφής, τα οποία εμφανίζονται δίπλα από τα δεδομένα του προϊόντος. Όταν κάποιος χρήστης κάνει κλικ στο κουμπί διαγραφής, εμφανίζεται ένα μήνυμα επιβεβαίωσης όπου ζητείται από τον χρήστη να επιβεβαιώσει τη διαγραφή του προϊόντος. Αν ο χρήστης επιλέξει τη διαγραφή, το προϊόν διαγράφεται από τη βάση δεδομένων, και εμφανίζεται ένα

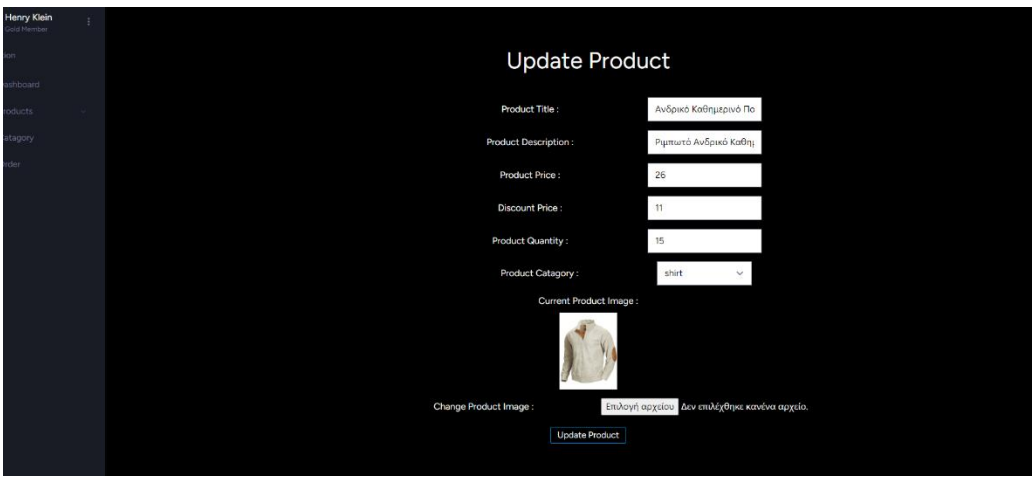
μήνυμα επιβεβαίωσης ότι η διαγραφή πραγματοποιήθηκε επιτυχώς. Αν ο χρήστης ακυρώσει τη διαγραφή, τότε δεν γίνεται τίποτα





Κουμπί επεξεργασίας Προιόντος(edit)

Όταν κάποιος χρήστης κάνει κλικ στο κουμπί επεξεργασίας, το σύστημα ανακατευθύνει το χρήστη σε μια νέα σελίδα όπου μπορεί να ενημερώσει τα δεδομένα του προϊόντος. Το κάθε προϊόν που επιλέγει ο χρήστης εμφανίζεται στη σελίδα ενημέρωσης με όλες τις λεπτομέρειές του, συμπεριλαμβανομένου του τίτλου, της περιγραφής, της τιμής και της κατηγορίας. Επιπλέον, παρέχεται η δυνατότητα αλλαγής της εικόνας του προϊόντος. Αφού γίνουν οι αλλαγές, ο χρήστης λαμβάνει ένα μήνυμα επιβεβαίωσης επιτυχούς ενημέρωσης.



5.4 User Side (Πλευρά Χρήστη)

5.4.1 Εμφάνιση Προιόντων στον Χρήστη.

Αρχικά κάνουμε register έναν νέο άτομο τυπου χρήστη . Εφόσον γίνεται σύνδεση χρήστη θα οδηγηθούμε στο template που προστέθηκε στην αρχή οπου είναι η βάση του eshop για αυτό που θα βλέπει ο χρήστης . Παρατηρούμε το navigation και παρατηρούμε την επιλογή products καθώς σε αυτήν θα γίνει επεξεργασία ώστε να εμφανίζονται τα προιόντα που προστεθηκαν από admin στην βάση οπου από εκεί θα τα ανακτήσουμε .

ADMIN - > ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ - > USER

Γίνεται η επιλογή προιόντων για προβολή .Τα προιόντα φορτώνονται από τη βάση δεδομένων και εμφανίζονται σε μια λίστα στη σελίδα προϊόντων στη σελίδα products .Υπάρχουν εικόνες για κάθε προϊόν που προσκομίζεται από τη βάση δεδομένων. Σε αυτό το σημείο, παρακολουθούμε τις οδηγίες του Amin να κρατήσουμε μόνο την κύρια εικόνα και να αφαιρέσουμε τις υπόλοιπες εικόνες από τη σελίδα προϊόντων.

Μετά την επιλογή των κύριων εικόνων προϊόντος, εμφανίζονται ο τίτλος και η τιμή κάθε προϊόντος κάτω από την κύρια εικόνα. Τα δεδομένα αυτά προέρχονται επίσης από τη βάση δεδομένων. Εάν ένα προϊόν έχει τιμή με έκπτωση, τότε εμφανίζεται η τιμή με έκπτωση δίπλα στην κανονική τιμή. Αυτή η λειτουργία υλοποιείται με ένα if statement που ελέγχει την ύπαρξη της τιμής με έκπτωση επίσης εφαρμόζονται στυλιστικές αλλαγές στις τιμές για να διακρίνονται. Για παράδειγμα, η κανονική τιμή εμφανίζεται σε μπλε χρώμα και η τιμή με έκπτωση σε γκρι.

Επίσης προστίθετε το σύστημα σελιδοποίησης (pagination) που εφαρμόζεται στον κώδικα δίνει τη δυνατότητα στους χρήστες να περιηγούνται σε διαφορετικές σελίδες των δεδομένων όταν αυτά είναι πολλά και δεν χωράνε όλα σε μία σελίδα. Αυτό είναι ιδιαίτερα χρήσιμο όταν έχουμε μεγάλο όγκο δεδομένων που πρέπει να προβληθούν στον χρήστη χωρίζοντάς τα σε σελίδες. Ο κώδικας περιλαμβάνει τη χρήση της συνάρτησης paginate() στο αίτημα για τη λήψη των δεδομένων από τη βάση δεδομένων. Με τη χρήση αυτής της συνάρτησης, τα δεδομένα διαιρούνται σε διάφορες σελίδες με βάση τον αριθμό που καθορίζεται από τον παράγοντα perPage, ο οποίος ορίζει πόσα δεδομένα θα εμφανίζονται

ανά σελίδα. Στον κώδικα παρατηρούμε επίσης τη χρήση της συνάρτησης links() που παρέχει τα στοιχεία πλοήγησης μεταξύ των σελίδων. Αυτή η συνάρτηση δημιουργεί σύνδεσμους για τις προηγούμενες και επόμενες σελίδες, καθώς και για τις σελίδες που περιέχουν τα ενδιάμεσα δεδομένα. Το αποτέλεσμα είναι μια σελίδα με πλοήγηση που επιτρέπει στον χρήστη να περιηγηθεί στις διάφορες σελίδες των δεδομένων όταν αυτά είναι πολλά, προσφέροντας έτσι μια καλύτερη εμπειρία χρήστη.

paginate(10) – εώς 10 προίντα κάθε φορά

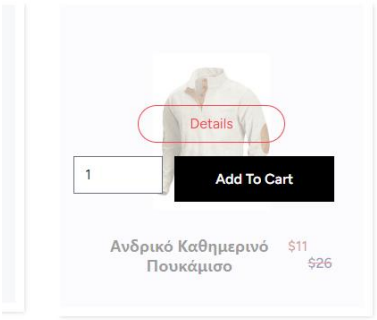
5.4.2 Προβολή λεπτομερειών ανα προιόν(Product Details).

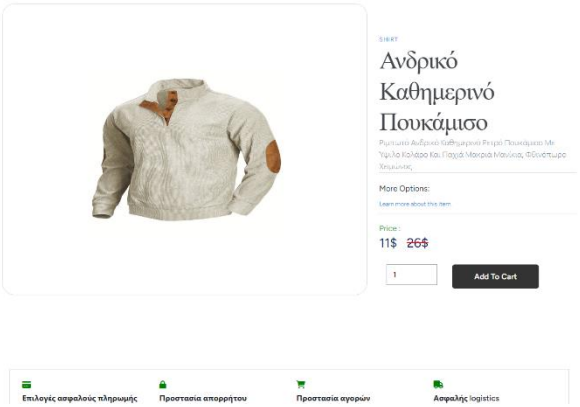
προσθέσουμε μια επιλογή για λεπτομέρειες προϊόντος στη σελίδα, όπου ο χρήστης θα μπορεί να δει όλες τις σχετικές πληροφορίες για ένα συγκεκριμένο προϊόν.

Όταν κάποιος χρήστης κάνει κλικ στο κουμπί "Details", θα μεταφερθεί σε μια άλλη σελίδα που θα εμφανίζει όλες τις λεπτομέρειες του επιλεγμένου προϊόντος. Αυτή η νέα σελίδα θα περιλαμβάνει το όνομα του προϊόντος, την τιμή του, την τιμή με έκπτωση, καθώς και άλλες πληροφορίες που έχουμε στον πίνακα των προϊόντων, όπως η περιγραφή, η κατηγορία και η διαθεσιμότητα. Για να επιτύχουμε αυτό, πρέπει να δημιουργήσουμε μια νέα σελίδα με τον τίτλο "Λεπτομέρειες προϊόντος". Αυτή η σελίδα θα περιέχει το header και το footer του ιστότοπου μας, καθώς και τις λεπτομέρειες του επιλεγμένου προϊόντος. Στον κώδικα της σελίδας θα χρησιμοποιήσουμε το πρότυπο του bootstrap για να δημιουργήσουμε ένα ελκυστικό και λειτουργικό σχήμα εμφάνισης.

Για να επιτύχουμε τη μετάβαση στις λεπτομέρειες προϊόντος, θα χρησιμοποιήσουμε το πρωτόκολλο HTTP και τον controller του Laravel για να χειριστούμε το αίτημα του χρήστη. Όταν ο χρήστης κάνει κλικ στο κουμπί "Details", θα ανακατευθύνεται στη σελίδα με τις λεπτομέρειες του προϊόντος, όπου θα εμφανίζονται όλες οι πληροφορίες που

αντιστοιχούν στο επιλεγμένο προϊόν.Τέλος, θα προσθέσουμε ένα κουμπί "Add To Cart" στη σελίδα λεπτομερειών προϊόντος, ώστε ο χρήστης να μπορεί να προσθέσει το προϊόν στο καλάθι αγορών του. Αυτό θα επιτρέψει στον χρήστη να προχωρήσει στην αγορά του προϊόντος μετά την περιήγηση στις λεπτομέρειες του προϊόντος.

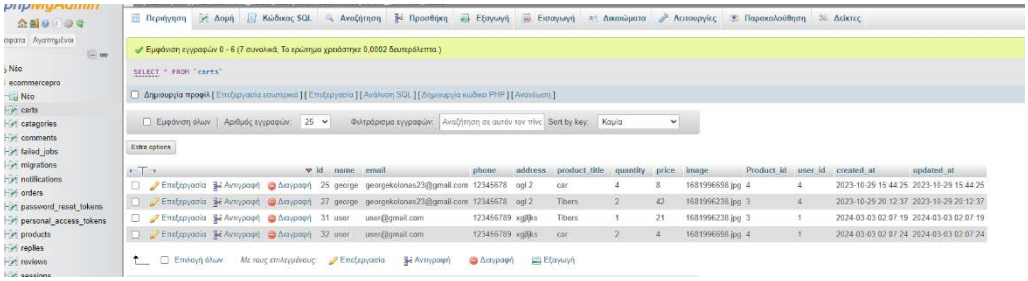




Επίσης προσθέτουμε την λειτουργία να γίνεται έλεγχος για το αν ο χρήστης είναι συνδεδεμένος ή όχι πριν προστεθεί ένα προϊόν στο καλάθι αγορών. Αν ο χρήστης δεν είναι συνδεδεμένος, θα γίνει ανακατεύθυνση στη σελίδα σύνδεσης. Εάν ο χρήστης είναι συνδεδεμένος, το προϊόν θα προστίθεται στο καλάθι αγορών του.

5.4.3 User Cart (Καλάθι αγορών χρήστη).

Προσθήκη ενός νέου πίνακα στη βάση δεδομένων με το όνομα "card" για την αποθήκευση πληροφοριών καλαθιού αγορών. Αυτός ο πίνακας περιλαμβάνει στήλες όπως το όνομα του πελάτη, το *email*, το *τηλέφωνο*, η *διεύθυνση*, ο *τίτλος του προϊόντος*, η *τιμή*, η *ποσότητα*, τα *emails* (εάν υπάρχουν), καθώς και το *ID του προϊόντος* και *το ID του χρήστη*. Επίσης, προστίθενται διάφορες λειτουργίες στον κώδικα για τον έλεγχο του αν ο χρήστης είναι συνδεδεμένος πριν προσθέσει ένα προϊόν στο καλάθι αγορών και για την προσθήκη πληροφοριών στον πίνακα "card" όταν ο χρήστης πατάει το κουμπί "Add To Cart"



Στην συνέχεια αφού δημιουργήθηκε ένας πίνακα στη βάση δεδομένων με το όνομα "card", όπου θα αποθηκεύονται τα δεδομένα για κάθε προϊόν που προστίθεται στο καλάθι αγορών. Ο πίνακας αυτός περιλαμβάνει στήλες για το όνομα, το email, το τηλέφωνο, τη διεύθυνση, τον τίτλο του προϊόντος, την τιμή, την ποσότητα, την εικόνα του προϊόντος, το ID του προϊόντος και το ID του χρήστη έπειτα γίνεται επέκταση του HomeController για να διαχειρίζεται την αποθήκευση δεδομένων προϊόντος και χρήστη στον πίνακα "card". Ανακτάτε τα δεδομένα του χρήστη και του προϊόντος από τη βάση δεδομένων και τα αποθηκεύετε στον πίνακα "card", ενημερώνετε την ιστοσελίδα ώστε να δέχεται τα δεδομένα του προϊόντος και του χρήστη κατά την προσθήκη ενός προϊόντος στο καλάθι.όπου αυτά τα δεδομένα στη συνέχεια αποθηκεύονται στον πίνακα "card".

Με αυτόν τον τρόπο, η ιστοσελίδα αναβαθμίζεται για να υποστηρίζει τη λειτουργικότητα καλαθιού αγορών, προσθέτοντας τη δυνατότητα αποθήκευσης δεδομένων προϊόντος και χρήστη κατά την προσθήκη προϊόντων στο καλάθι.

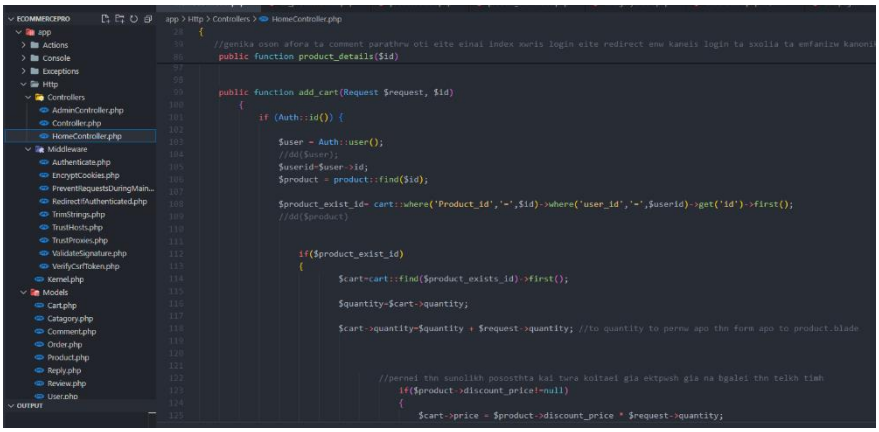
Αφού ο πελάτης πρόσθεσε ένα προϊόν στο καλάθι αγορών του .Τώρα θα δηιουργηθεί η επιλογή να δει όλα τα προϊόντα που πρόσθεσε στο καλάθι. Πρώτα, προστίθεται μια επιλογή "Cart" στο μενού στο πάνω μέρος της σελίδας. Κάνοντας κλικ σε αυτή την επιλογή, ο πελάτης μεταφέρεται σε μια σελίδα όπου μπορεί να δει όλα τα προϊόντα στο καλάθι του.

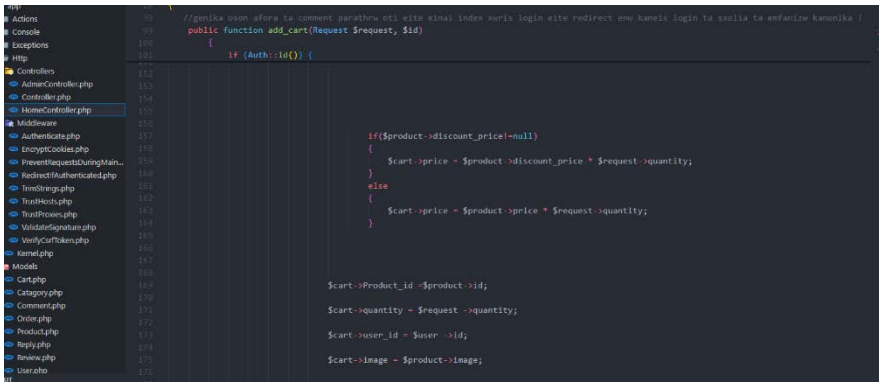
Για την υλοποίηση αυτού, ακολουθήθηκαν τα παρακάτω βήματα:

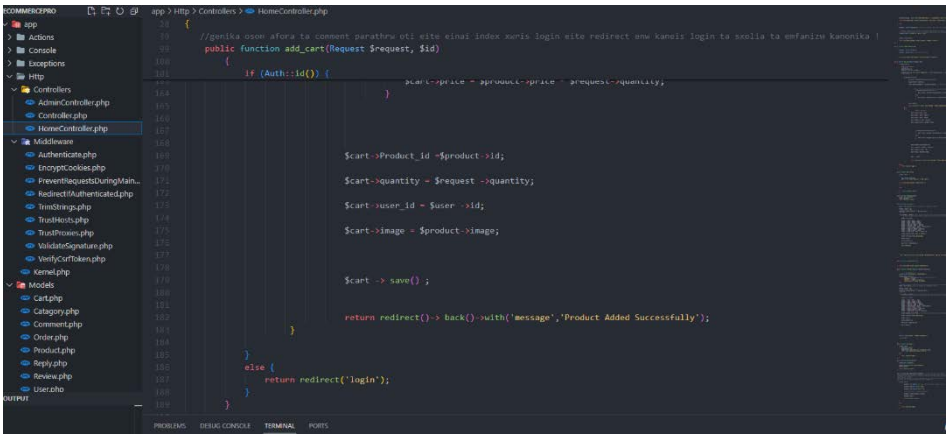
Στην κεφαλίδα της αρχικής σελίδας, προστίθεται η επιλογή "cart" επεξεργαζόμενοι το αρχείο header.blade.php που βρίσκεται στο φάκελο resources/views/home.

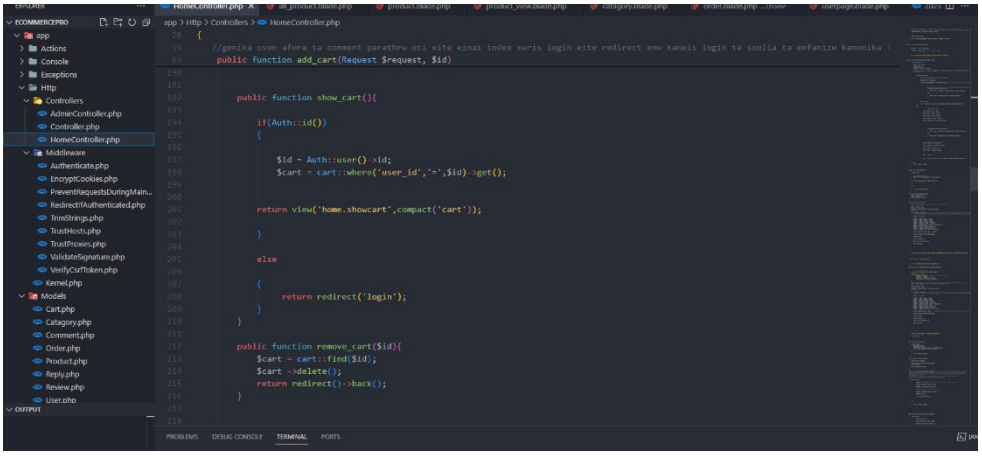
Ορίζεται μια διαδρομή(root) με το όνομα show_cart στο αρχείο web.php για να χειρίζεται αιτήσεις προβολής του καλαθιού. Στον HomeController, δημιουργείται μια μέθοδος με το όνομα show_cart για να χειριστεί αιτήσεις προβολής του καλαθιού. Αυτή η μέθοδος επιστρέφει μια προβολή με το όνομα show_cart.blade.php. Δημιουργείται ένα νέο αρχείο με το όνομα show_cart.blade.php στο φάκελο resources/views/home για να εμφανιστούν τα περιεχόμενα του καλαθιού. Αυτό το αρχείο περιλαμβάνει τα μέρη της κεφαλίδας και του υποσέλιδου της σελίδας, και ανάμεσά τους εμφανίζεται ένας πίνακας που δείχνει τον τίτλο του προϊόντος, την τιμή, την ποσότητα και την εικόνα για κάθε προϊόν στο καλάθι.

Home Controller Συναρτήσεις : **add_cart , show_cart, remove_cart**









Εφαρμόζονται μερικές βασικές στυλιστικές αλλαγές στον πίνακα για να γίνει πιο προσητός οπτικά. Αυτό περιλαμβάνει την κέντραρισμα του πίνακα, την προσθήκη περιγράμματος κτλπ. Αφού δημιουργηθεί και στυλιστικά διαμορφωθεί η σελίδα του καλαθιού, οι πελάτες μπορούν να τον προβούν κάνοντας κλικ στην επιλογή "Cart" στη γραμμή μενού. Έτσι, θα μπορούν να δουν όλα τα προϊόντα που έχουν προσθέσει στο καλάθι τους.

Επιπλέον γίνεται η προσθήκη επιλογής αφαίρεσης ώστε ο χρήστης να μπορεί να αφαιρέσει ένα συγκεκριμένο προϊόν από το καλάθι. Για αυτό, προσθέτουμε ένα κουμπί "Αφαίρεση προϊόντος" σε κάθε εγγραφή του καλαθιού.

Αρχικά, προσθέτουμε ένα κουμπί με την ετικέτα " Remove Product".Για να επιτευχθεί η αφαίρεση, προσθέτουμε έναν σύνδεσμο με ένα URL που καλεί τη μέθοδο remove_card στον HomeController, μαζί με το αναγνωριστικό του προϊόντος που πρέπει να διαγραφεί από το καλάθι. Στον HomeController, όπως φαίνετε παραπάνω ορίστηκε μια μέθοδος remove_card που δέχεται το αναγνωριστικό του προϊόντος ως παράμετρο και διαγράφει το συγκεκριμένο προϊόν από το καλάθι. Στη συνέχεια, ανακατευθύνουμε τον χρήστη πίσω στη σελίδα του καλαθιού. Για να επιβεβαιώσουμε τη διαγραφή προτού εκτελεστεί, προσθέτουμε έναν περιορισμό επιβεβαίωσης στο κουμπί "Remove Product", όπου εμφανίζεται ένα μήνυμα επιβεβαίωσης πριν από τη διαγραφή. Με αυτόν τον τρόπο, ο χρήστης μπορεί να διαγράψει τα προϊόντα από το καλάθι του με ασφάλεια και επιβεβαιώνοντας πρώτα τη δράση.

5.4.4 Product Order (Παραγγελία Προιόντος).

Μετά την προσθήκη προϊόντων στο καλάθι αγορών, δημιουργείται μια διαδικασία όπου ο χρήστης μπορεί να ολοκληρώσει την παραγγελία του. Όταν ο πελάτης κάνει κλικ στην επιλογή παραγγελίας, όλα τα προϊόντα στο καλάθι θα μεταφερθούν στον πίνακα παραγγελιών. Θα δημιουργήσουμε έναν νέο πίνακα που θα ονομάζεται "Order" για αυτόν τον σκοπό. Όταν ο πελάτης επιβεβαιώνει την παραγγελία, θα έχει δύο επιλογές: "Cash on Delivery" και "Pay Using Card.

Μέσα στο αρχείο show_cart.blade.php, μετά την ενότητα με τη συνολική τιμή, θα προσθέσουμε ένα νέο div για την πρόοδο στην παραγγελία. Μέσα σε αυτό το div, θα υπάρχουν δύο κουμπιά: "**Cash on Delivery**" και "**Pay Using Card."**

Αφού προσθεθεί κάποιος σχεδιασμός, θα προχωρήσουμε στη δημιουργία του πίνακα **Order** στη βάση δεδομένων. Θα εκτελέσουμε την εντολή ***php artisan make:model Order -m*** για να δημιουργήσουμε το μοντέλο Order model και το migration file. Στη συνέχεια, θα ορίσουμε τη δομή του πίνακα Order στο migration file , συμπεριλαμβανομένων στηλών για τα στοιχεία του χρήστη, τα στοιχεία του προϊόντος, την κατάσταση πληρωμής και την κατάσταση παράδοσης.

Μόλις ολοκληρωθεί| το migration , θα εκτελέσουμε την εντολή php artisan migrate για να δημιουργήσουμε τον πίνακα Order στη βάση δεδομένων. Το επόμενο βήμα είναι η υλοποίηση της λειτουργικότητας για τη μεταφορά των στοιχείων του καλαθιού στον πίνακα Order όταν ο πελάτης επιβεβαιώνει την παραγγελία.

Συνολικά, η διαδικασία περιλαμβάνει τον σχεδιασμό του τμήματος παραγγελίας, τη δημιουργία του πίνακα Παραγγελίας στη βάση δεδομένων και την υλοποίηση της λογικής για τη μεταφορά των στοιχείων του καλαθιού στον πίνακα Παραγγελίας κατά την επιβεβαίωση της παραγγελίας. Αυτό θα διευκολύνει τη διαδικασία παραγγελίας για τους πελάτες στην πλατφόρμα.

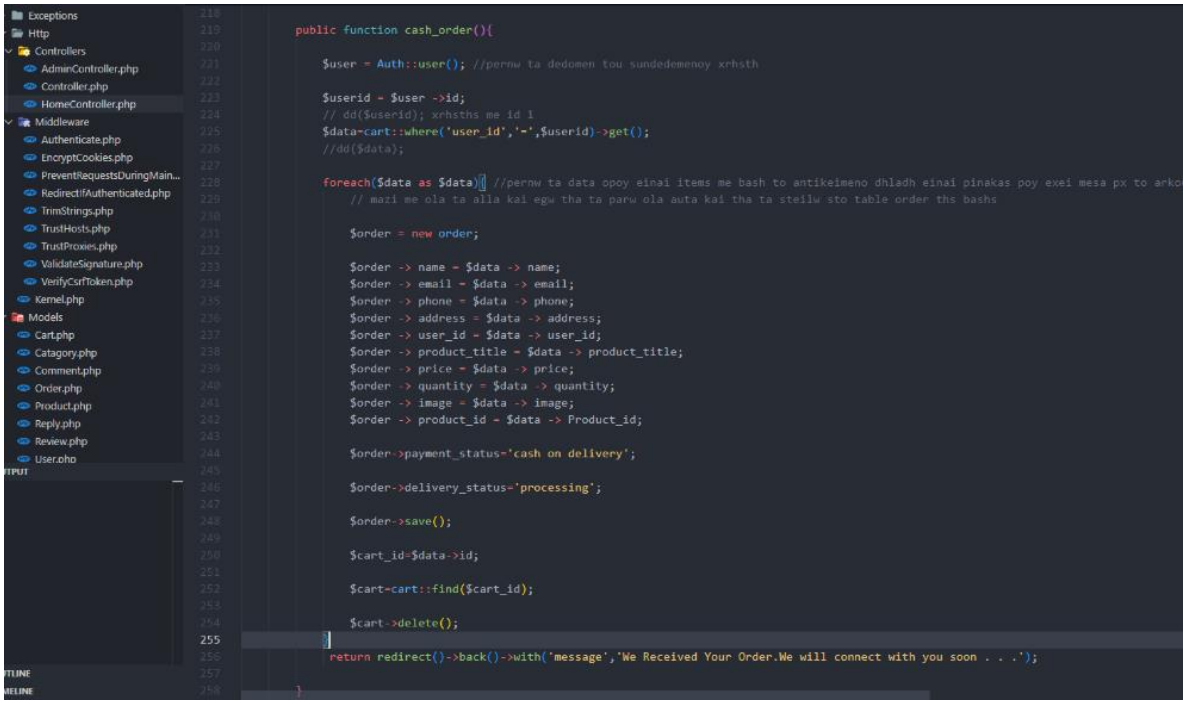
Στην συνέχεια θα πρέπει τα δεδομένα να μεταφερθούν στον πίνακα παραγγελιών όταν κάποιος κάνει κλικ στην επιλογή πληρωμής με αντικαταβολή. Για να το κάνουμε αυτό, θα πρέπει να δημιουργήσουμε ένα route στο αρχείο web.php που θα δρομολογεί στη μέθοδο cash_order του HomeController.

Αφού δημιουργήσουμε το route, θα χρειαστεί να υλοποιήσουμε τη λογική στον HomeController για τη μεταφορά των δεδομένων από τον πίνακα καλαθιού στον πίνακα παραγγελιών. Αυτό περιλαμβάνει την εύρεση του χρήστη που είναι συνδεδεμένος, την ανάκτηση των δεδομένων του καλαθιού για αυτόν το χρήστη και τη μεταφορά τους στον πίνακα παραγγελιών. Αφού ολοκληρώσουμε την υλοποίηση της μεταφοράς των δεδομένων στον πίνακα παραγγελιών, επιστρέφουμε στην ίδια σελίδα καλαθιού

Συνολικά, αυτή η διαδικασία ενσωματώνει τη δημιουργία ενός route, την εξαγωγή των δεδομένων του καλαθιού για τον συνδεδεμένο χρήστη και τη μεταφορά τους στον πίνακα

παραγγελιών, καθώς και την επιστροφή στη σελίδα καλαθιού μετά την ολοκλήρωση της διαδικασίας.

HomeController / cash_order (function)



Επίσης έχει προσθεθεί μια λειτουργία που θα διαγράφει τα προϊόντα από το καλάθι του χρήστη μετά τη μεταφορά τους στον πίνακα παραγγελιών και θα εμφανίζει ένα μήνυμα επιβεβαίωσης ότι η παραγγελία ολοκληρώθηκε με επιτυχία.

Όταν ο χρήστης επιλέξει να πληρώσει με αντικαταβολή, τα προϊόντα που έχει προσθέσει στο καλάθι του θα μεταφερθούν στη λίστα παραγγελιών. Ταυτόχρονα, τα προϊόντα αυτά θα διαγραφούν από το καλάθι του χρήστη. Έπειτα, ο χρήστης θα λάβει ένα μήνυμα επιβεβαίωσης ότι η παραγγελία του ολοκληρώθηκε με επιτυχία.

Αυτό εξασφαλίζει ότι οι πληροφορίες παραμένουν οργανωμένες και ότι ο χρήστης ενημερώνεται για την πρόοδο της παραγγελίας του.

5.4.5 Πληρωμή με Κάρτα (Stripe Payments).

Στην επιλογή "Pay with Card", ουσιαστικά διαμορφώνουμε τη λειτουργικότητα για τους πελάτες να πληρώνουν τις παραγγελίες τους χρησιμοποιώντας πιστωτικές ή χρεωστικές κάρτες.

Αρχικά γίνεται η εγκατάσταση της Βιβλιοθήκης της Stripe: Ξεκινάμε εγκαθιστώντας τη βιβλιοθήκη Stripe PHP χρησιμοποιώντας το Composer. Αυτή η βιβλιοθήκη μας επιτρέπει να αλληλεπιδρούμε με το API της Stripe και να επεξεργαζόμαστε πληρωμές καρτών.

```
composer require stripe/stripe-php
```

Δημιουργία Λογαριασμού στη Stripe

Στη συνέχεια, πρέπει να δημιουργήσουμε ένα λογαριασμό στην πλατφόρμα της Stripe. Αυτό περιλαμβάνει την εγγραφή στον ιστότοπο της Stripe, την παροχή απαραίτητων πληροφοριών όπως email, όνομα και χώρα, και την επαλήθευση του λογαριασμού μέσω email.

Αφού δημιουργήσουμε έναν λογαριασμό στην Stripe, λαμβάνουμε τα κλειδιά API μας από τον Πίνακα Διαχείρισης της Stripe. Αυτά τα κλειδιά αποτελούνται από ένα Δημοσιεύσιμο Κλειδί και ένα Μυστικό Κλειδί, τα οποία θα χρησιμοποιήσουμε για να αυθεντικοποιήσουμε τις αιτήσεις μας στο API της Stripe.

Ρύθμιση των Κλειδιών API στο Αρχείο Περιβάλλοντος

Στη συνέχεια, προσθέτουμε τα κλειδιά API της Stripe στο αρχείο **.env.** Αυτό το αρχείο αποθηκεύει ευαίσθητες πληροφορίες ρύθμισης για την εφαρμογή μας. Προσθέτουμε το public Κλειδί και το secret Κλειδί ως μεταβλητές περιβάλλοντος όπως STRIPE_PUBLISHABLE_KEY και STRIPE_SECRET_KEY.

Ολοκληρώνοντας αυτά τα βήματα, έχουμε προετοιμάσει την εφαρμογή μας για να αλληλεπιδράσει με τις υπηρεσίες επεξεργασίας πληρωμών της Stripe. Με τη βιβλιοθήκη Stripe εγκατεστημένη και τα κλειδιά API ρυθμισμένα, μπορούμε τώρα να υλοποιήσουμε τη λειτουργικότητα για την επεξεργασία πληρωμών καρτών εντός της εφαρμογής μας. Αυτό περιλαμβάνει τη δημιουργία προβολών για τις φόρμες πληρωμής, την πραγματοποίηση των απαραίτητων αλλαγών στον ελεγκτή για την χειρισμό αιτημάτων πληρωμής και την ενσωμάτωση του API της Stripe για την ασφαλή επεξεργασία των πληρωμών με κάρτα.

Δημιουργία Root View για Stripe Payments

Δημιουργία (Route)

Αρχικά, γίνεται αναφορά στη δημιουργία ενός δρομολογίου με το όνομα "Stripe", το ο-ποίο θα συνδέεται με μια συνάρτηση με το όνομα "Stripe" στον HomeController. Αυτό το δρομολόγιο αναμένεται να χειρίζεται τις πληρωμές στην εφαρμογή.

Δημιουργία (View)

Έπειτα, δημιουργείται μια προβολή με το όνομα "payment.blade.php" στο φάκελο των προβολών της εφαρμογής. Αυτή η προβολή περιλαμβάνει ένα form για την αποστολή των στοιχείων πληρωμής.

Ενσωμάτωση Stripe SDK

Στο κομμάτι του κώδικα που ακολουθεί, φαίνεται να ενσωματώνεται το Stripe SDK για τη διαχείριση των πληρωμών. Αυτό περιλαμβάνει τη δημιουργία ενός τοκέν (token) για τα στοιχεία της πληρωμής και την αποστολή αυτού του τοκέν στον διακομιστή.

Έλεγχος Στοιχείων Πληρωμή

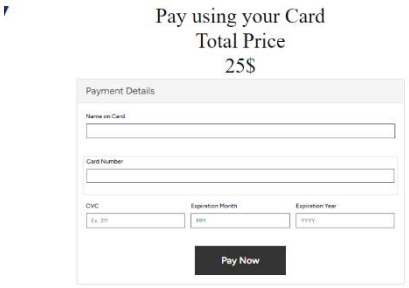
Ο κώδικας περιέχει λογική για τον έλεγχο των στοιχείων της πληρωμής που καταχω-ρούνται από τον χρήστη. Αυτό περιλαμβάνει τον έλεγχο του αριθμού της κάρτας, του CBC και της ημερομηνίας λήξης.

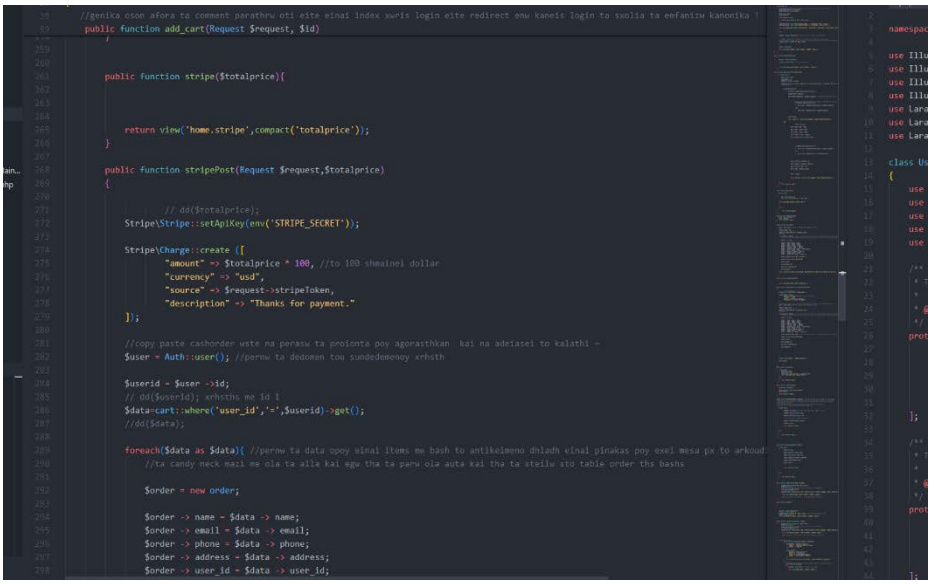
Ανταπόκριση και Αποθήκευση Τοκέν

Ο κώδικας περιλαμβάνει μια ανταπόκριση από το Stripe API, η οποία περιλαμβάνει ένα τοκέν. Αυτό το τοκέν αποθηκεύεται και χρησιμοποιείται για την πραγματοποίηση της πληρωμής.

Εμφάνιση Μηνυμάτων Λάθους και Επιτυχίας

Υπάρχει λογική για την εμφάνιση μηνυμάτων λάθους αν η πληρωμή αποτύχει και μη-νύματα επιτυχίας αν η πληρωμή πραγματοποιηθεί με επιτυχία.





5.5 Πρόσθετες λειτουργίες στην πλευρά του διαχειριστή(Additional features)

Προσθήκη επιλογής "Order" στο sidebar του διαχειριστή

Προστέθηκε μια νέα επιλογή στο μενού του διαχειριστή, ώστε οι διαχειριστές να μπορούν να διαχειριστούν παραγγελίες όπως να ενημερώνουν αν η παραγγελία εχει γίνει delivered και να στέλνουν ένα σχετικό email ως αναφορά την κατάσταση παραγγελιάς και να κρατούν σε pdf την κάθε παραγγελία

Δημιουργία νέου URL και συνάρτησης στον διαχειριστή

Δημιουργήθηκε ένα νέο URL route στο web.php

```
route::get('/order',[AdminController::class,'order']);
```

της εφαρμογής, που αντιστοιχεί σε μια νέα συνάρτηση στον ελεγκτή (controller) του διαχειριστή. Αυτή η συνάρτηση είναι υπεύθυνη για την ανάκτηση και την προβολή των παραγγελιών.

```
    public function order()
    {

      $order=order::all();


      return view('admin.order',compact('order')); //stelnw ola ta
data tou pinaka order ths bashs sto order blade
    }


    public function delivered($id) //kanw catch to id apo to route
poy to esteila apo to order.blade
    {

      $order=order::find($id);
      $order->delivery_status="delivered";
      $order->payment_status="i";
      $order->save();
      //ARA me to poy pathsw to button sto order blade tou deliv-
ered oti exei paradothei enhmerwnei
      //thn bash me bash to id tou kathe product sthn sthlh deliv-
ery status oti exei paradothei
      return redirect()->back();
    }
```

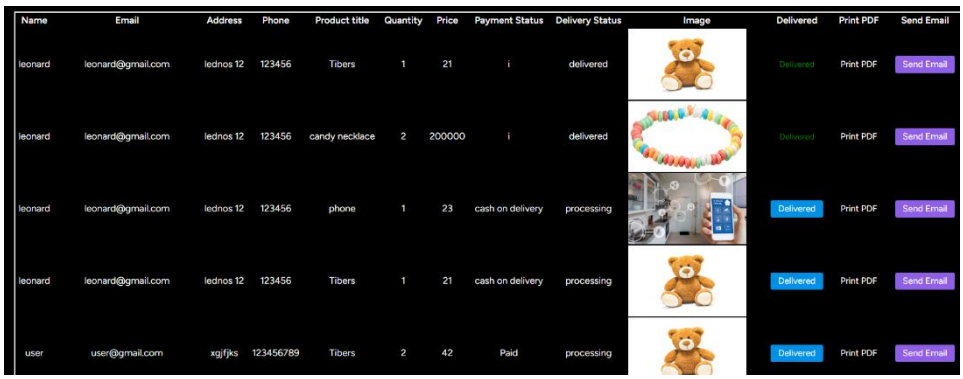
Δημιουργία view για τις παραγγελίες

Δημιουργήθηκε ένα νέο view **order.blade.php** στον φάκελο του διαχειριστή, η οποία περιλαμβάνει τον κώδικα HTML για την εμφάνιση των παραγγελιών σε μορφή πίνακα.

Ακολουθεί το <table>

```
 <table class="table_deg ">

                <tr>

                    <th>Name</th>
                    <th>Email</th>
                    <th>Address</th>
                    <th>Phone</th>
                    <th>Product title</th>
                    <th>Quantity</th>
                    <th>Price</th>
                    <th>Payment Status</th>
                    <th>Delivery Status</th>
                    <th>Image</th>
                    <th>Delivered</th>
                    <th>Print PDF</th>
                    <th>Send Email</th>
                    <!-- to delivered tha einai mia sthlh gian ble-
poym ti exei paradothei  -->


                </tr>
                        @forelse($order as $order)
            <tr>

                <td>{{$order->name}}</td>
                <td>{{$order->email}}</td>
                <td>{{$order->address}}</td>
                <td>{{$order->phone}}</td>
                <td>{{$order->product_title}}</td>
                <td>{{$order->quantity}}</td>
                <td>{{$order->price}}</td>
                <td>{{$order->payment_status}}</td>
                <td>{{$order->delivery_status}}</td>
                <td>
                    <img class="imgs" src="/product/{{$order->im-
age}}" alt="">
                </td>



                <td>

                @if($order->delivery_status=='processing')
```

```
                    <a href="{{url('delivered',$order->id)}}" on-
click="return confirm('Are you sure this product is delivered ?!')"
class="btn btn-primary">Delivered</a>
                    <!-- pernw to sugkekrimeno id stelnei to id
dhladh sto route -->
                    @else

                    <p style="color: green;">Delivered</p>




                    @endif
                </td>
                <td>
                  <a href="{{url('print_pdf',$order->id)}}"
class="btn btn-secendary">Print PDF</a>

                </td>

                <td>

                    <a href="{{url('email',$order->id)}}"
class="btn btn-info">Send Email</a>

                </td>
            </tr>
                            @empty
                            <tr>
                               <td colspan="16">
                                No data Found
                               </td>
                            </tr>


                        @endforelse



            </table>
```



Οι αλλαγές που πραγματοποιήθηκαν:

Προστέθηκε μια επιπλέον στήλη με την ονομασία "delivered" στον πίνακα διαχείρισης των παραγγελιών του διαχειριστή για να εμφανίζεται η κατάσταση παράδοσης κάθε παραγγελίας.

Προστέθηκε ένα κουμπί με την ετικέτα " delivered " για κάθε παραγγελία. Πατώντας αυτό το κουμπί, το Delivery Status όπως φαίνετε στην εικόνα αλλάζει από " processing " σε " delivered ".

Η κατάσταση παράδοσης εμφανίζεται στην στήλη " delivered ". Εάν η κατάσταση είναι " processing ", εμφανίζεται το κουμπί για αλλαγή της κατάστασης. Εάν η κατάσταση είναι " delivered ", αντί για το κουμπί εμφανίζεται ένα κείμενο που υποδηλώνει ότι το προϊόν παραδόθηκε.

Επίσης προστέθηκε ένα παράθυρο επιβεβαίωσης κατά την κλικ στο κουμπί " delivered" για να διασφαλιστεί ότι ο διαχειριστής επιβεβαιώνει ότι η παραγγελία παραδόθηκε.Ο χρωματισμός του κειμένου που υποδεικνύει την κατάσταση παράδοσης άλλαξε σε πράσινο για να γίνει πιο ευδιάκριτο όταν μια παραγγελία παραδίδεται.

Payment Status: Επιπλέον, όταν μια παραγγελία επισημαίνεται ως παραδόθηκε, η κατάσταση πληρωμής ενημερώνεται αυτόματα από " i" σε "paid".

Έγινε εισαγωγή δεδομένων δοκιμής όπου προστέθηκαν μερικά δεδομένα στον πίνακα παραγγελιών, ώστε να μπορέσει να ελεγχθεί η σωστή λειτουργία της προβολής.

Η διαδικασία για την εμφάνιση και τον λήψη PDF αρχείου σε μια εφαρμογή Laravel.

Γίνεται εγκατάσταση του πακέτου dompdf μέσω του composer. Στη συνέχεια,γίνεται πρόσθεση του πακέτου στον παροχέα υπηρεσιών και στο αρχείο config/app.php. Μετά την εγκατάσταση και τη ρύθμιση του πακέτου, δημιουργήθηκε μια στήλη "PDF" στην εφαρμογή μας, καθώς και ένα κουμπί "Print PDF" για κάθε παραγγελία. Όταν κάποιος χρήστης κάνει κλικ στο κουμπί "Print PDF", εκτελείται μια ενέργεια για τη δημιουργία ενός PDF αρχείου που περιλαμβάνει τα δεδομένα της συγκεκριμένης παραγγελίας. Οι πληροφορίες του πελάτη και του προϊόντος εμφανίζονται στο PDF αρχείο, συμπεριλαμβανομένης της εικόνας του προϊόντος. Τέλος, το PDF αρχείο κατεβαίνει στη συσκευή του χρήστη για αποθήκευση ή εκτύπωση.

5.6 Email Setup

Σε αυτήν την ενότητα θα γίνει προσθήκη λειτουργικότητας αποστολής email στη σελίδα του admin.

5.6.1 SMTP

Το **SMTP (Simple Mail Transfer Protocol**) είναι ένα πρωτόκολλο που χρησιμοποιείται για την αποστολή ηλεκτρονικού ταχυδρομείου στο Διαδίκτυο. Αποτελεί το βασικό πρωτόκολλο για τη μετάδοση μηνυμάτων ηλεκτρονικού ταχυδρομείου μεταξύ διακομιστών ηλεκτρονικού ταχυδρομείου και επιτρέπει την αποστολή και λήψη email μεταξύ διαφορετικών χρηστών. Όταν ένας χρήστης αποστέλλει ένα email, το πρωτόκολλο SMTP χρησιμοποιείται για να μεταφέρει το μήνυμα από τον αποστολέα στον διακομιστή SMTP του παραλήπτη. Έπειτα, το email διαβιβάζεται μέσω άλλων διακομιστών μέχρι να φτάσει στον παραλήπτη. Ο παραλήπτης μπορεί στη συνέχεια να ανακτήσει το email χρησιμοποιώντας το πρωτόκολλο ανάκτησης email, όπως το POP3 (Post Office Protocol) ή το IMAP (Internet Message Access Protocol). Το SMTP χρησιμοποιείται επίσης για την αποστολή αυτοματοποιημένων μηνυμάτων, όπως ειδοποιήσεις, επιβεβαιώσεις και αυτόματες απαντήσεις. Είναι ένα απαραίτητο εργαλείο για τη λειτουργία του ηλεκτρονικού ταχυδρομείου στο Διαδίκτυο.

5.6.2 Επιβεβαίωση email μετά από εγγραφή

Ρύθμιση SMTP στο αρχείο .env:

Ορίστε τις ρυθμίσεις SMTP για την αποστολή email. Αυτές περιλαμβάνουν:

MAIL_MAILER=smtp: Ορίζει τον τύπο του mailer ως SMTP.

MAIL_HOST=smtp.gmail.com: Το SMTP server για την αποστολή email μέσω Gmail.

MAIL_PORT=465: Ορίζει τη θύρα SMTP για τη σύνδεση με τον SMTP server.

MAIL_USERNAME=your_email@gmail.com: Το email από το οποίο θα αποστέλλονται τα email επαλήθευσης.

MAIL_PASSWORD=your_app_password: Ο κωδικός πρόσβασης της εφαρμογής που παρέχει πρόσβαση στο email σας.

MAIL_ENCRYPTION=ssl: Η μέθοδος κρυπτογράφησης που θα χρησιμοποιηθεί για τη σύνδεση με τον SMTP server.

MAIL_FROM_ADDRESS=your_email@gmail.com: Η διεύθυνση email από την οποία θα αποστέλλονται τα email.

Ρύθμιση επαλήθευσης email στο αρχείο auth.php:

Ανοίξτε το αρχείο **config/auth.php.**

Βρείτε τη γραμμή που λέει 'verify' => false και αφαιρέστε τα σχόλια (//) προκειμένου να ενεργοποιηθεί η επαλήθευση του email.

Προσθήκη κώδικα στο μοντέλο χρήστη:

Ανοίξτε το μοντέλο χρήστη στον κατάλογο app/Models/User.php.

Ενσωματώστε το MustVerifyEmail trait στην κλάση User.

Υλοποιήστε τη μέθοδο emailVerifiedAt() η οποία επιστρέφει την ημερομηνία επαλήθευσης του email.

Δημιουργία προφίλ χρήστη:

Επεκτείνετε τη φόρμα εγγραφής του Laravel για να περιλαμβάνει πεδία για το email, το όνομα χρήστη και τον κωδικό πρόσβασης, καθώς και πεδίο επαλήθευσης κωδικού.

Αποστολή επιβεβαιωτικού email:

Αφού ο χρήστης εγγραφεί στο σύστημα, αποστείλετε ένα email επαλήθευσης στη διεύθυνση email που καταχώρησε. Το email πρέπει να περιέχει έναν σύνδεσμο που οδηγεί στη διαδικτυακή σελίδα επαλήθευσης του email.

Επαλήθευση του email:

Ο χρήστης πρέπει να κάνει κλικ στον σύνδεσμο επαλήθευσης στο email που λάβει.

Αφού επιβεβαιώσει το email του, η στήλη email_verified_at στον πίνακα χρηστών ενημερώνεται για να επισημανθεί ότι το email έχει επαληθευτεί.

Αυτά τα βήματα συνοψίζουν τη διαδικασία επαλήθευσης του email στο Laravel, που περιγράφεται στον οδηγό που παρατέθηκε.

5.6.3 Δημιουργία View για Αποστολή Email Notification

Προσθήκη κουμπιού "Αποστολή Email":

Αρχικά, δημιουργείτε ένα κουμπί "Send Email" στη λίστα των παραγγελιών. Αυτό το κουμπί επιτρέπει στον χρήστη να αποστείλει ένα email σε μια συγκεκριμένη διεύθυνση email.

Το κουμπί δημιουργείται στο αρχείο admin/orders.blade.php μέσα στον βρόχο foreach που δημιουργεί τις γραμμές του πίνακα για κάθε παραγγελία.

Δημιουργία route και συνάρτησης στον controller:

Roote

```
route::get('/email/{id}',[AdminController::class,'email']);

route::post('/send_user_email/{id}',[AdminControl-
ler::class,'send_user_email']);
```

Στο αρχείο διαδρομών routes/web.php, προσθέτετε μια διαδρομή με το όνομα email που καλεί τη μέθοδο emEmail στον AdminController.

```
public function email($id)
  {
```

```
    $order = order::find($id); // psaxnei me to id poy tou stalthke
sto table order kai krataei oti brei gia to id  to sugkkrimeno
    return view('admin.email',compact('order'));
  }


  public function send_user_email(Request $request ,$id)
  {
      $order=order::find($id);

      $details = [
        'greeting' => $request -> greeting,

        'firstline' => $request -> firstline,

        'body' => $request -> body,

        'button' => $request -> button,

        'url' => $request->url ,

        'lastline' => $request -> lastline,

      ];

      Notification::send($order, new SendEmailNotification($de-
tails));

      //gia na mporesq na to xrhesimpoihsq den xexnw na to balw
sthn arxh me to use


      return redirect()->back();


  }
```

Η διαδρομή αυτή θα πρέπει να έχει μια παράμετρο για το ID της παραγγελίας.

Δημιουργία φόρμας για την αποστολή email:

Στο αρχείο admin/email_info.blade.php, δημιουργήθηκε μια φόρμα για την αποστολή email όπου η φόρμα αυτή περιλαμβάνει πεδία για το περιεχόμενο του email, όπως τον

χαιρετισμό, το κυρίως μήνυμα και το κείμενο του κουμπιού που θα πρέπει να εμφανίζεται στο email.

```
 <form action="{{url('send_user_email',$order->id)}}"
method="POST">

          @csrf


          <div class="text-center display-5 p-5">
            <label>Email Greeting : </label>
            <input type="text" name="greeting" >
          </div>

          <div class="text-center display-5 p-5">
            <label>Email FirstLine : </label>
            <input type="text" name="firstline" >
          </div>

          <div class="text-center display-5 p-5">
            <label>Email Body : </label>
            <input type="text" name="body" >
          </div>

          <!-- tha uparxei ena button poy tha exei ena link to
opoio  -->


          <div class="text-center display-5 p-5">
            <label>Email Url : </label>
            <input type="text" name="url" >
          </div>


          <div class="text-center display-5 p-5">
            <label>Email Last Line : </label>
            <input type="text" name="lastline" >
          </div>

          <div class="text-center  p-5">

            <input class="bg-light" type="submit" value="Send
Email" class="btn btn-primary btn-lg" >
          </div>
    </form>
```

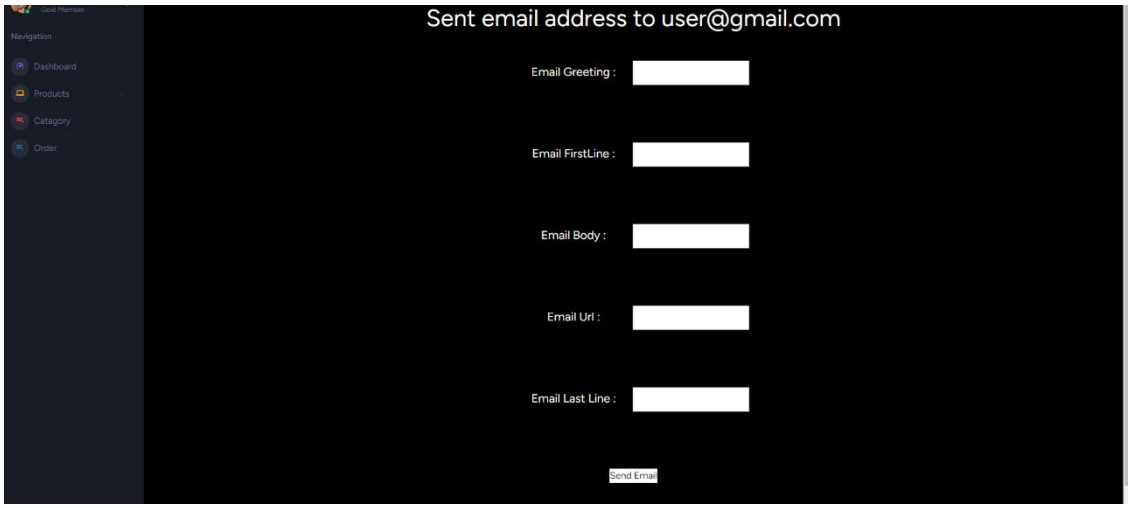
Υλοποίηση συνάρτησης στον controller για την αποστολή email:

Στον AdminController, όπως φαίνεται στο παραπάνω κομμάτι του κώδικα προστέθηκε μια μέθοδο senduseremail, η οποία θα λαμβάνει το ID της παραγγελίας ως παράμετρο.

Σε αυτή τη μέθοδο, ανακτάτε τα δεδομένα της συγκεκριμένης παραγγελίας και τα προωθείτε στην προβολή για την αποστολή email.

Προβολή φόρμας αποστολής email:

Στον AdminController, όταν ο χρήστης πατάει το κουμπί "Send Email", τον κατευθύνετε στο view email.blade.php .Τα δεδομένα της συγκεκριμένης παραγγελίας φορτώνονται στη φόρμα για να τα χρησιμοποιήσει ο χρήστης κατά την αποστολή του email.



5.6.4 Αποστολή Email Notification(Λειτουργικότητα)

Δημιουργία Migration Πίνακα για τις Ειδοποιήσεις (Notification Table Migration):

: php artisan notifications:table.

Δημιουργεί ενα (migration) για τον πίνακα ειδοποιήσεων. Αυτός ο πίνακας θα αποθηκεύει πληροφορίες σχετικά με τις ειδοποιήσεις που θα στέλνονται μέσω της εφαρμογής.

Προσθήκη Πεδίων στο Πίνακα Ειδοποιήσεων (Notification Table Fields):

: php artisan make:notifications_table.

Προστίθενται τα απαραίτητα πεδία για την αποθήκευση των πληροφοριών των ειδοποιήσεων, όπως η ταυτότητα του χρήστη, ο τύπος της ειδοποίησης, ο τίτλος και το περιεχόμενο της ειδοποίησης, καθώς και η ημερομηνία δημιουργίας.

Migration Execution:

: php artisan migrate.

Ετσι θα γίνει εφαρμογή των αλλαγών στη βάση δεδομένων μέσω της εντολής migrate, προκειμένου να δημιουργηθεί ο πίνακας ειδοποιήσεων.

Δημιουργία Κλάσης Ειδοποίησης (Notification Class Creation):

: php artisan make:notification SendEmailNotification.

Αλλαγές: Δημιουργία μιας νέας κλάσης ειδοποίησης με το όνομα **SendEmailNotification.** Αυτή η κλάση θα χρησιμοποιηθεί για τη δημιουργία και αποστολή email ειδοποιήσεων.

Υλοποίηση Λειτουργικότητας Αποστολής Email (Email Sending Functionality):

Υλοποίηση Συνάρτησης sendEmailNotification: Προσθήκη λειτουργικότητας για τη σύνθεση και αποστολή email ειδοποιήσεων χρησιμοποιώντας την κλάση ειδοποίησης που δημιουργήθηκε στο προηγούμενο βήμα.

Ενσωμάτωση Λειτουργικότητας Ειδοποιήσεων στον Κώδικα Εφαρμογής:

Αλλαγές στους Ελεγκτές (Controllers): Προσθήκη λειτουργικότητας στους ελεγκτές της εφαρμογής για τη διαχείριση και αποστολή των ειδοποιήσεων.

Αλλαγές στις Διαδρομές (Routes): Προσθήκη διαδρομών στο αρχείο διαδρομών της εφαρμογής για τη ρύθμιση των αιτημάτων που σχετίζονται με τις ειδοποιήσεις.

(Στις προηγούμενες εικόνες φαίνεται το route και οι αλλαγές στον controller **)(send_user_emai)**

Notifications < SendEmailNotification.php αρχείο

```php
class SendEmailNotification extends Notification
{
    use Queueable;

    /**
     * Create a new notification instance.
     */

    private $details;

    public function __construct($details)
    {
        $this->details=$details;
    }

    /**
     * Get the notification's delivery channels.
     *
     * @return array<int, string>
     */
    public function via(object $notifiable): array
    {
        return ['mail'];
    }

    /**
     * Get the mail representation of the notification.
```

```
     */
    public function toMail(object $notifiable): MailMessage
    {
        return (new MailMessage)
                    ->greeting($this->details['greeting'])
                    ->line($this->details['firstline'])
                    ->line($this->details['body'])
                    ->action($this->details['button'],$this->de-
tails['url'])
                    ->line($this->details['lastline']);
    }

    /**
     * Get the array representation of the notification.
     *
     * @return array<string, mixed>
     */
    public function toArray(object $notifiable): array
    {
        return [
            //
        ];
    }
}
```

5.7 Σελίδα Χρήστη Show και Cancel Order

Δημιουργία Επιλογής Oder στο menu της σελίδας του πελάτη όπου θα δείχνει την κατάσταση της παραγγελίας (Χρήστη):

Προστέθηκε μια νέα επιλογή , δίπλα από την επιλογή cart , που ονομάζεται "Order".

Αυτή η επιλογή θα εμφανίζεται μόνο όταν ο χρήστης είναι συνδεδεμένος.

Δρομολόγηση Επιλογής Παραγγελίας:

Αν ο χρήστης είναι συνδεδεμένος και κάνει κλικ στην επιλογή "Order", δρομολογείται στην προβολή της παραγγελίας.

Αν ο χρήστης δεν είναι συνδεδεμένος, τον ανακατευθύνει στη σελίδα σύνδεσης.

Order.blade.php

```
 <table class="text-center border">
```

```
            <tr>
                <th class="border">Product Title</th>

                <th class="border">quantity</th>

                <th class="border">Price</th>

                <th class="border">Payment Status</th>

                <th class="border">Delivery Status</th>

                <th class="border">Image</th>

                <th class="border">Cancel Order</th>
            </tr>


            @foreach($order as $order)
            <tr class="border">
                <td>{{$order->product_title}}</td>
                <td>{{$order->quantity}}</td>
                <td >{{$order->price}}</td>
                <td>{{$order->payment_status}}</td>
                <td>{{$order->delivery_status}}</td>
                <td class="d-flex justify-content-center">
                    <img  width="150px" height="80px" src="prod-
uct/{{$order->image}}" alt="">
                </td>
                <td>
                    @if($order->delivery_status=='processing')
                    <a onclick="return confirm('Are you sure to
cancel this Order ?! ')"class="btn btn-danger" href="{{url('can-
cel_order',$order->id)}}">Cancel Order</a>
                    @elseif($order->delivery_status=='Order Can-
celed')
                    <p>Order Canceled</p>
                    @else
                    <p>Order Delivered</p>



                    @endif

                </td>
```

```
            </tr>

            @endforeach
        </table>
```

Δημιουργήθηκε το view order.blade.php στον φάκελο του πελάτη.

Ελέγχος Σύνδεσης Χρήστη:

Στον ελεγκτή της αρχικής σελίδας (HomeController), προστέθηκε η μέθοδος **show_order** και **cancel order** όπου εδώ ο χρήστης μπορεί να ακυρώσει την παραγγελία

```
    public function show_order()
        {
            if(Auth::id())
            {   $user=Auth::user();
                $userid=$user->id;
                $order=order::where('user_id','=',$userid)->get();
                return view('home.order',compact('order'));
            }
            else
            {
                return redirect('login');
            }
        }

        public function cancel_order($id)
        {
            $order=order::find($id);

            $order->delivery_status='Order Canceled';
            $order->save();

            return redirect()->back();
        }
```

Routes :

```
route::get('/show_order',[HomeController::class,'show_order']);

route::get('/cancel_order/{id}',[HomeController::class,'cancel_or-
der']);
```

Αυτή η μέθοδος ελέγχει εάν ο χρήστης είναι συνδεδεμένος.

Αν είναι, εμφανίζεται η προβολή της παραγγελίας.

Αν δεν είναι, ο χρήστης ανακατευθύνεται στη σελίδα σύνδεσης.

Order Details

Product Title	quantity	Price	Payment Status	Delivery Status	Image	Cancel Order
Tibers	1	21	Paid	Order Canceled		Order Canceled
car	1	2	Paid	Order Canceled		Order Canceled
Tibers	1	21	Paid	Order Canceled		Order Canceled
car	1	2	Paid	Order Canceled		Order Canceled
Tibers	2	42	i	delivered		Order Delivered

5.8 Σύστημα Προσθήκης Σχόλιων (Comments and Reply)

Η δημιουργία ενός συστήματος σχολίων αποτελεί ένα σημαντικό στοιχείο σε πολλές εφαρμογές και ιστοσελίδες, καθώς παρέχει έναν τρόπο για τους χρήστες να αλληλεπιδρούν, να ανταλλάσσουν απόψεις και να μοιράζονται τις εμπειρίες τους.

Σε αυτήν την ενότητα θα εξετάσουμε τις βασικές αρχές και τα βήματα για τη δημιουργία ενός λειτουργικού συστήματος σχολίων με την χρήση της Laravel.

Η δημιουργία ενός συστήματος σχολίων σε μια εφαρμογή Laravel ξεκινά με την προσθήκη μιας φόρμας στη σελίδα, όπου ο χρήστης μπορεί να εισάγει τα σχόλιά του. Μετά την εισαγωγή του σχολίου, ο χρήστης μπορεί να πατήσει το κουμπί "Comment" για να υποβάλει το σχόλιο του.

Μόλις το σχόλιο υποβληθεί, εμφανίζεται κάτω από τη φόρμα σχολίων. Αυτό δημιουργεί ένα ροή σχολίων που αντανακλά τις απόψεις και τις αντιδράσεις των χρηστών.

Πέραν αυτού, κάθε σχόλιο συνοδεύεται από ένα κουμπί "Reply". Όταν ένας χρήστης κάνει κλικ στο "Reply", εμφανίζεται ένα πεδίο κειμένου υποσχομένο για απάντηση στο συγκεκριμένο σχόλιο. Αυτό επιτρέπει στους χρήστες να ανταλλάσσουν απόψεις, να συζητούν και να αναπτύσσουν τις ιδέες τους πάνω στα ήδη υπάρχοντα σχόλια.

Δημιουργία Πίνακα Σχολίων (Comment Table) και Πίνακα Απαντήσεων (Reply Table)

Δημιουργώντας δύο πίνακες στη βάση δεδομένων, έναν για τα σχόλια και έναν για τις απαντήσεις. Αυτοί οι πίνακες περιλαμβάνουν τα πεδία που απαιτούνται για να αποθηκεύσουν τα σχόλια και τις απαντήσεις, όπως το όνομα του χρήστη, το κείμενο του σχολίου και της απάντησης, καθώς και το ID του χρήστη που τα πρόσθεσε.

create_comments_table.php για τη δημιουργία του πίνακα σχολίων (comments table) με τα απαραίτητα πεδία **id, name , comment.**

create_replies_table.php για τη δημιουργία του πίνακα απαντήσεων (replies table) με τα απαραίτητα πεδία **id , name , comment_id(που θα είναι το comment στο οποιό έγινε reply) ,reply ,user_id**

Μοντέλα (Models):

Comment.php: Το μοντέλο Comment.php δημιουργήθηκε για να αντιστοιχεί στον πίνακα σχολίων και να διαχειρίζεται τις σχετικές ενέργειες.

Reply.php: Αντίστοιχα, δημιουργήθηκε το μοντέλο Reply.php για τις απαντήσεις.

Κώδικας για την προσθήκη της λειτουργία προσθήκη σχόλιων και reply σε αυτών στην αρχική σελίδα της ιστοσελίδας :

```
<div class="text-center pb-4">
            <h1 style="font-size: 30px; text-align: center; pad-
ding-top: 20px; padding-bottom: 20px">Comments</h1>


         <form action="{{url('add_comment')}}" method="POST">
               @csrf
            <textarea style="height:150px; width: 600px;" place-
holder="Comment something here" name="comment"></textarea>
```

```
            <br>

            <input class="btn btn-primary" type="submit"
value="Comment">
            <!-- <a href="" class="btn btn-primary">Comment</a> -->
         </form>

         </div>

         <div style="padding-left:20%;">

            <h1 style="font-size: 30px;"  class="pb-4">All Com-
ments</h1>
               @foreach($comment as $comment)
            <div>

               <b>User : {{$comment->name}}</b>
               <p>Comment : {{$comment->comment}}</p>

               <a class="text-info" href="javascript::void(0); "on-
click="reply(this)" data-Commentid="{{$comment->id}}">Reply</a>
               <br><br>
               <!-- apothkuw sto data-commentid to id tou  comment
-->



            </div>

               @foreach($reply as $rep)

               @if($rep->comment_id==$comment->id)
               <div style="padding-left: 3%; padding-bottom:10px;
padding-bottom: 10px;">
                  <b>User : {{$rep->name}}</b>
                  <p>Comment : {{$rep->reply}}</p>
                  <a class="text-info" href="javascript::void(0);
"onclick="reply(this)" data-Commentid="{{$comment->id}}">Reply</a>

                     <br><br>
               </div>
               @endif
               @endforeach

            @endforeach
```

```
            <div style="display: none; " class="replyDiv">

                <form action="{{url('add_reply')}}" method="POST">

                 @csrf
                   <input type="text" id="commentId" name="commen-
tId" hidden="" >

                   <textarea style="height: 100px; width: 500px;"
name="reply" placeholder="write something here "></textarea>

                   <br>

                   <button type="submit" class="btn btn-warning">Re-
ply</button>

                   <a href="javascript::void(0); " class="btn btn-
primary" onclick="reply_close(this)">Close</a>
                   <!-- an pathsw edw kleinw to div  -->


                </form>
            </div>

        </div>
```

Κλικ στο Κουμπί "Reply": Όταν ο χρήστης κάνει κλικ σε ένα κουμπί "Reply", ένα JavaScript σενάριο ενεργοποιείται.

Εμφάνιση Πεδίου Απάντησης:

Το JavaScript σενάριο επιλέγει το αντίστοιχο πεδίο απάντησης που αντιστοιχεί στο κουμπί "Reply" που κλικάρισε ο χρήστης και το εμφανίζει .

Απόκρυψη Πεδίων Απάντησης:

Το JavaScript σενάριο αποκρύπτει όλα τα πεδία απάντησης κατά την αρχικοποίηση της σελίδας, ώστε να μην είναι ορατά από προεπιλογή.

Κώδικας javascript

```
 function reply(caller)
                {

                   document.getElementById('commen-
tId').value=$(caller).attr('data-Commentid');
                   $('.replyDiv').insertAfter($(caller));
                   $('.replyDiv').show();

                }



                function reply_close(caller)
                {

                   $('.replyDiv').hide();

                }
```

Οι Συναρτήσεις add_comment και add_reply στον controller HomeController.php

```
        public function add_comment(Request $request) //περνς απο
την form μεσω του ακτιον στο add_comment
        //stelnw auta apo thn form mesw tou request kai kanw neo
antikeimeno sthn bash me tble comment kai
        //apothkeuw sthn sthlh comment to sxolio kanw savve kai me
to klik gurnaw sthn selida poy hmoyn
        //alliws gurnaw sto login
        //auto omws apo thn bash to sxolio poy apothekuw pws to em-
fanizw sthn userpage ?
        //ta stelnw ola sto index kai feontai eite eisai logged h
oxi
        {
            if(Auth::id())
            {
```

```
                    $comment= new comment;//new table name //table
name, comment , user_id

                    $comment->name=Auth::user()->name;

                    $comment->user_id=Auth::user()->id;

                    //tha kanw request to comment name=comment

                    $comment->comment=$request->comment;

                    $comment->save();

                    return redirect()->back();

            }

            else

            {
                return redirect('login');
            }
        }




        public function add_reply(Request $request)
        {
            if(Auth::id())
            {
                    $reply=new reply;

                    $reply->name=Auth::user()->name;

                    $reply->user_id=Auth::user()->id;

                    $reply->comment_id=$request->commentId;

                    $reply->reply=$request->reply;

                    $reply->save();
```

```
                return redirect()->back();

        }

        else

        {
            return redirect('login');
        }
    }
```

Routes :

route::post('/add_comment',[HomeController::class,'add_comment']);

route::post('/add_reply',[HomeController::class,'add_reply']);

5.9 Αναζήτηση προιόντων στο homepage

Η προσθήκη λειτουργίας αναζήτησης προϊόντων στην αρχική σελίδα ενός ηλεκτρονικού καταστήματος είναι ζωτικής σημασίας για τη βελτίωση της εμπειρίας του χρήστη και τη διευκόλυνση της πλοήγησής του. Με την προσθήκη ενός απλού αλλά αποτελεσματικού πεδίου αναζήτησης στην αρχική σελίδα, οι χρήστες μπορούν εύκολα και γρήγορα να εντοπίσουν τα προϊόντα που τους ενδιαφέρουν.

Προσθήκη Φόρμας Αναζήτησης: Προσθέσατε ένα στοιχείο <form> στη σελίδα, πιθανότατα μέσα στο πρότυπο HTML της εφαρμογής Laravel σας. Αυτή η φόρμα περιλαμβάνει ένα πεδίο <input> για το ερώτημα αναζήτησης και ένα κουμπί υποβολής.

```
 <form action="{{url('product_search')}}" method="get">

            @csrf

            <input style="width:500px; "type="text" name="search"
placeholder="Search for something">

            <input type="submit" value="search">

            </form>
```

Χειρισμός Φόρμας: Συνάρτηση χειρισμού με όνομα "search" για την υποβολή της φόρμας. Αυτή η συνάρτηση θα οριστεί στον ελεγκτή μέσω του route της θα λάβει το ερώτημα αναζήτησης που υποβάλλεται από τον χρήστη και θα εκτελέσει τις αναγκαίες λειτουργίες αναζήτησης, πιθανώς ερωτώντας τη βάση δεδομένων για σχετικά προϊόντα βάσει του ερωτήματος αναζήτησης.

Το route :

```
route::get('/product_search',[HomeController::class,'prod-
uct_search']);
```

homecontroller function product_search

```
   public function product_search(Request $request)
        {
                $comment=comment::orderby('id','desc')->get();
                $reply=reply::all();
                //pernw to search apo thn form kai 4axnw sthn bash
me auto name=search
                $search_text=$request->search;
                    //table name product
                $product=product::where('ti-
tle','LIKE',"%$search_text%")->orWhere('cata-
gory','LIKE',"%$search_text%")->paginate(10);
```

```
                return view('home.userpage',com-
pact('product','comment','reply'));

               //4axnw me bash ton titlo o titlos na einai like to
$search_text poy phra

        }
```

Η μέθοδος **product_search** στον HomeController διαχειρίζεται τη λειτουργία αναζήτησης προϊόντων. Αρχικά, ανακτούνται όλα τα σχόλια και οι απαντήσεις από τη βάση δεδομένων, ταξινομημένα κατά φθίνουσα σειρά βάσει του πεδίου 'id'. Έπειτα, λαμβάνεται το κείμενο αναζήτησης από την είσοδο του χρήστη. Στη συνέχεια, γίνεται εκτέλεση ενός ερώτηματος στη βάση δεδομένων χρησιμοποιώντας το μοντέλο Product για να βρεθούν προϊόντα με τίτλο ή κατηγορία που ταιριάζουν με το κείμενο αναζήτησης. Τα αποτελέσματα παρουσιάζονται σε σελίδες, κάθε σελίδα με μέγεθος 10 προϊόντων. Τέλος, θα επιστραφεί η προβολή 'home.userpage' μαζί με τα αποτελέσματα της αναζήτησης, τα σχόλια και τις απαντήσεις στη φόρμα αναζήτησης.

6 Συμπεράσματα

Συμπερασματικά κατά τη διάρκεια της δημιουργίας του ηλεκτρονικού καταστήματος χρησιμοποιώντας το Laravel, ως φοιτητής είχα την ευκαιρία να εξερευνήσω τις λειτουργίες και τις δυνατότητες αυτής της mvc αρχιτεκτονικής. Ανακάλυψα ότι το Laravel προσφέρει ένα πλούσιο οικοσύστημα εργαλείων και λειτουργιών που διευκολύνουν τη δημιουργία πλήρως λειτουργικών και ασφαλών ηλεκτρονικών καταστημάτων. Ανέπτυξα καλές γνώσεις σχετικά με τη χρήση της PHP για τη διαχείριση των δεδομένων και των λειτουργιών του διακομιστή, καθώς και τη σημασία της HTML, CSS και JavaScript για τον σχεδιασμό και την αλληλεπίδραση με τον χρήστη. Από την άλλη πλευρά, η δημιουργία ενός ηλεκτρονικού καταστήματος με το Laravel μου έδωσε μια εμπειρία στην ανάπτυξη σύνθετων ιστοσελιδών , ενώ ταυτόχρονα βελτίωσα τις δεξιότητές μου στον προγραμματισμό. Αναγνώρισα τη σημασία της αντιμετώπισης προκλήσεων καθώς κατά τη διαδικασία ανάπτυξης μου έδωσε αυτοπεποίθηση και έμπνευση για περαιτέρω εξέλιξη στον τομέα της αναπτυξης ιστοσελιδών. Συνολικά, η εμπειρία μου με το php framework της Laravel στη δημιουργία ενός ηλεκτρονικού καταστήματος ήταν εξαιρετικά εποικοδομητική και με έβαλε σε ένα μονοπάτι προς την επαγγελματική μου ανάπτυξη στο κομμάτι της ανάπτυξης ιστοσελιδών.

Βιβλιογραφία

[1] https://www.freecodecamp.org/news/html-css-and-javascript-explained-for-beginners/

[2] https://blog.hubspot.com/marketing/web-design-html-css-javascript.

[3] https://blog.miva.com/the-history-of-ecommerce-how-did-it-all-begin

[4] https://en.wikipedia.org/wiki/Consumer_electronics_store

[5] https://iservices.gr/kataskevi-eshop/woocommerce/

[6] https://techplace.gr/woocommerce-ti-einai-kai-giati-na-to-epilexo/

[7] https://kinsta.com/knowledgebase/what-is-wordpress/

[8] https://www.magestore.com/blog/what-is-magento/

[9] https://www.talend.com/resources/what-is-mysql/#:~:text=MySQL%20is%20a%20relational%20database,information%20in%20a%20corporate%20network.

[10] https://laravel.com/

[11] https://www.tutorialspoint.com/laravel/laravel_overview.htm

[12] https://medium.com/@nnadichime04/laravel-mvc-architecture-explained-21e783dbbd14#:~:text=Laravel%20is%20a%20popular%20PHP,the%20view%2C%20and%20the%20controller.

[13] https://www.w3schools.com /

[14] https://www.udemy.com/course/create-e-commerce-website-using-laravel-beginner-to-advance/?couponCode=ST15MT31224

[15] https://www.apachefriends.org /

[16] https://www.javatpoint.com/phpmyadmin

[17] https://getbootstrap.com/

[18] https://code.visualstudio.com/

Ο Φάκελος με τον κώδικα την βάση και όλα τα απαραίτητα αρχεία βρίσκεται στο παρακάτω link :

https://drive.google.com/drive/folders/1LG0wpa38UcDpDPFLht3XSzWqwob6ZG7x?usp=sharin