Ελληνικό Μεσογειακό Πανεπιστήμιο Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

«Υλοποίηση Ηλεκτρονικού καταστήματος για χρυσοχοείο με σύστημα διαχείρισης περιεχομένου & Mobile Application με απομακρυσμένη διαχείριση βάσης δεδομένων»

Τσαχρέλιας Γεώργιος ΑΜ:3951

Επιβλέπων καθηγητής : Παπαδάκης Νικόλαος

ΗΡΑΚΛΕΙΟ
2019

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κύριο Παπαδάκη Νικόλαο που μου εμπιστεύτηκε αυτή την πτυχιακή και πίστεψε στο θέμα που ανέπτυξα και ασχολήθηκα και πάνω από όλα για όλη την βοήθεια και την κατεύθυνση του σε ότι ζήτημα προέκυψε από το ξεκίνημα μέχρι και το τελείωμα τις πτυχιακής μου και κυρίως την οικογένεια μου που μου στάθηκε τόσο χρόνια ώστε να ολοκληρώσω τα νεύρα μου. Τέλος θα ήθελα να ευχαριστήσω το παλιό μου τμήμα που μπήκα σε αυτήν την σχολή με όνομα Εφαρμοσμένης Πληροφορικής και Πολυμέσων και τέλος τους υπολοίπους καθηγητές που με διδάξανε τις γνώσεις τους και εξελίχθηκα.

Περιεχόμενα

Table of Contents

Πίνακας Εικόνων

Πίνακας Πινάκων

Σύνοψη

Σκοπός της παρούσας πτυχιακής εργασίας είναι η δημιουργία ενός custom CMS e-commerce παραγγελιών. Το σύστημα αυτό θα επιτρέπει στον επισκέπτη να δημιουργεί λογαριασμό η θα επιλέγει σαν επισκέπτης για να συνεχίσει με το να διαλέγει από κάποιο μενού τα προϊόντα που επιθυμεί. Ο διαχειριστής θα μπορεί να εμπλουτίζει το μενού των προϊόντων καταχωρώντας τις τιμές το όνομα ακόμα και την φωτογραφία και να μπορεί να ορίζει το Φόρος Προστιθέμενης Αξίας. Στη συνέχεια θα εγκατασταθεί το διαχειριστικό με την χρήση custom commands τα οποία εγκαθιστούν την βάση δεδομένων και προσθέτουν τα κατάλληλα δεδομένα ώστε να έχουμε όλη την εμφάνιση των δεδομένων για το διαχειριστικό το οποίο είναι βασισμένο σε ανοιχτού κώδικα λογισμικού με την χρήση αντικειμενοστραφής δομής Laravel. Η παρούσα πλατφόρμα θα ανέβει σε απευθείας σύνδεση στο διαδίκτυο σε webserver εικονική μηχανή του Okeanos. Και τέλος την mobile εφαρμογή η οποία είναι υλοποιημένη με την χρήση του Framework React Native και παρουσιάζω κάποια βασικά προνόμια που θα έχει ένας εγγεγραμμένος χρηστής του ηλεκτρονικού καταστήματος.

Abstract

<< Implementation of an online store for jewels with a content management system (CMS) & Mobile Application with remote database management >>

The aim of the present thesis statement is the creation of an e-commerce shop. Its system shall allow the customer to sign up or choose to continue as a guest, by clicking from a specific product menu, the object desired. The administrator shall be enabled to enrich the menu of products by listing their names, prices, pictures and even add their VAT. After that, the admin panel will be installed by using custom commands, and then add the appropriate data in database so to have the complete presentation of the data panel, which is based on an open source project, with the use of the object-oriented programming framework, Laravel. The platform will be uploaded in a stable internet connection to the Virtual Machine server, Okeanos. Finishing the mobile application which has already been created with the use of Framework React Native will also been presented, along with some of the basic privileges of being a signed-up user of the e-commerce shop.

Κεφάλαιο 1: Εισαγωγή

1.1 Περίληψη

Ο σκοπός της πτυχιακής εργασία μου ήταν να αναπτύξω ένα ηλεκτρονικό κατάστημα για χρυσοχοείου το οποίο θα βασιζόταν σε ένα ανοιχτού κώδικα λογισμικού το οποίο θα ήταν βασισμένο σε ένα Συστήματα Διαχείρισης Περιεχομένου με προσαρμοσμένο κώδικα με την χρήση αντικειμενοστραφής προγραμματισμός και το framework Laravel και το πακέτο voyager για το διαχειριστικό και όχι κάποιο έτοιμο Joomla/WordPress/Drupal CMS και μια εφαρμογή για android κινητά το οποίο θα ήταν μόνο για τους πελάτες που έχουν εγγραφεί στο ηλεκτρονικό κατάστημα.

Το εν μέρη ηλεκτρονικό κατάστημα πληροί μια γκάμα από δυνατότητες όπως ένα κλασικό WordPress σύστημα διαχείριση περιεχομένου με πολλαπλούς χρήστες και δικαιώματα για τον κάθε χρήστη όπως πελάτες , αρθρογράφους , συντάκτες διαχειριστές και μια κατηγορία πλήρης διαχείρισης. Ακόμα μπορεί να προσθέσει / αφαιρέσει/ επεξεργαστεί / δει προϊόντα , κατηγορίες προϊόντων, blog posts, blog κατηγορίες, blog tags να προσθέσει μεθόδους πληρωμής ακόμα και μεθόδους μεταφοράς.

Έχει την δυνατότητα να δει ηλεκτρονικές παραγγελίες να ενημέρωση την κατάσταση παραγγελίας και να δει πια πληρωμή με πιστωτική κάρτα δεν πέρασε η δεν πληρώθηκε.

Υπάρχει ακόμα και η δυνατότητα να διαχειριστεί πολυμέσα που θέλει να προσθέσει η να αφαιρέσει ακόμα και να κάνει επεξεργασία φωτογραφιών και τέλος ακόμα ένα προνόμιο μόνο για τους σούπερ διαχειριστές όπου μπορούν να προσθέσουν νέους πίνακες στην βάση δεδομένων να φτιάξουν μια νέα λογική να ενώσουν σχέσεις πινάκων και να προσθέσουν δυναμικά περιεχόμενο σε αυτά που δημιούργησαν.

Το εν λόγω mobile application έχει τα βασικά πράγματα ώστε να γίνει μια παραγγελία κατευθείαν από το κινητό android του πελάτη χωρίς να πάει στον περιηγητή να κάνει την παραγγελία του. Μπορεί να δει το ιστορικό παραγγελιών του τα προϊόντα που έχει βάλει στα αγαπημένα του ακόμα και το profile του και να αλλάξει τα προσωπικά του στοιχεία.

1.2 Κίνητρο για τη διεξαγωγή της εργασίας

Το κίνητρο για την πτυχιακή που παρουσιάζω ήταν η εκμάθηση κατασκευής ιστοσελίδων και mobile application με νέες τεχνολογίες οι οποίες είναι open source και έχουν μεγάλο εύρος

commit στο GitHub και πολλούς συνεργάτες που εργάζονται καθημερινά και κάνουν διορθώσεις .

Επίσης στην σημερινή εποχή η χρήση του Διαδικτύου είναι μέρος της καθημερινότητάς μας.

Ήθελα να κατασκευάσω μια ιστοσελίδα και ένα application για κινητά που θα χρησιμοποιηθεί για προγραμματιστικούς σκοπούς και όχι για κάποια ιδιαίτερη χρήση η κάποια πώληση σε κάποιο κατάστημα, ωστόσο και για διευκόλυνση των μελλοντικών πελατών στο να κάνουν ηλεκτρονικές αγορές σε ένα ολοκληρωμένο σύστημα e-commerce.

1.3 Σκοπός και στόχος εργασίας

Ο σκοπός της εργασίας ήταν να δημιουργηθεί ένας διαδικτυακός ιστότοπος και μια εφαρμογή για κινητά τηλέφωνα που θα είναι εύκολα προσβάσιμα στον επισκέπτη που θα ήθελε να κάνει μια ηλεκτρονική παραγγελία με 3 απλά βήματα.

Θα μπορεί με απλές κινήσεις να κάνει τις αγορές προϊόντων που επιθυμεί μονός του από οποιοδήποτε σημείο του πλανήτη αυτός επιθυμεί είτε σαν εγγεγραμμένος χρήστης είτε σαν επισκέπτης.

Στόχος της εργασίας είναι να παρέχει όλες τις λειτουργίες με ευκολία στον πελάτη για την ολοκλήρωση πληρωμής παραγγελιών (e-commerce) με την χρήση είτε πιστωτικής κάρτας είτε αντικαταβολής και επιλογή με την μέθοδο διανομής θα φτάσει στο σπίτι του ή στο κοντινότερο κατάστημα της περιοχής του.

1.4 Δομή της εργασίας

Η συνέχεια της εργασίας έχει οργανωθεί σε 9 κεφάλαια:

Κεφάλαιο 2: Αναφέρονται χρήσιμοι ορισμοί και κάποιες βασικές έννοιες για την χρήση του διαδικτύου ακόμα και ορισμοί για το τη είναι ηλεκτρονικό εμπόριο καθώς και τους τρόπους πληρωμής

Κεφάλαιο 3: παρουσίαση εργαλείων που χρησιμοποίησα για την κατασκευή της ιστοσελίδας και για την υλοποίηση της εφαρμογής για κινητά

Κεφάλαιο 4: Παρουσιάζονται απαραίτητες τεχνολογίες για την κατασκευή της ιστοσελίδας και της εφαρμογής με νέες τεχνολογίες και συγκρίσεις με άλλες παρόμοιες τεχνολογίες και κάποια πλεονεκτήματα / μειονεκτήματα για όλες αυτές της τεχνολογίες

Κεφάλαιο 5: Δίνω μια αναλυτική περιγραφή για το τη είναι ένα σύστημα διαχείρισης περιεχομένου (CMS), και συγκρίνω τα διαφορετικά ήδη των CMS που υπάρχουν στην αγορά και αναλύω τα πλεονεκτήματα και τα μειονεκτήματα.

Κεφάλαιο 6: Αναλύω το πακέτο Voyager και την αρχιτεκτονική του η οποία είναι βασισμένη στο framework της Laravel που με βάση αυτό έφτιαξα ένα custom σύστημα διαχείρισης περιεχομένου.

Κεφάλαιο 7: Εξήγηση τις εγκατάστασης του ηλεκτρονικού καταστήματος σε Microsoft Windows Server 2012 και αναφορά όσων third party βιβλιοθηκών τις Google, Stripe, Algolia, Mailgun, MailChimp χρειάστηκα.
Κεφάλαιο 8: Παρουσίαση κύριων σημείων του κώδικα και προσομοίωση τις ιστοσελίδας.

Κεφάλαιο 9: Παρουσίαση κύριων σημείων του κώδικα και προσομοίωση τις android εφαρμογής.

Κεφάλαιο 2: Μεθοδολογία Υλοποίησης

2.1 Μέθοδος ανάλυσης & ανάπτυξης πτυχιακής

ΜΕΘΟΔΟΛΟΓΙΑ	ΑΝΑΦΟΡΑ ΣΤΗΝ ΒΙΒΛΙΟΓΡΑΦΙΑ
Laragon Server	https://laragon.org/
Laravel	https://laravel.com/
Bootstrap 4	https://getbootstrap.com/
Voyager	https://laravelvoyager.com/
Stripe API	https://stripe.com/docs/api
Algolia	https://www.algolia.com/
React Native	https://facebook.github.io/react-native/
React Native Expo	https://docs.expo.io/versions/latest/

Πίνακας 1(technologies)

Third Party Libraries που χρειάστηκα για την web πλατφόρμα.

- https://github.com/LaravelCollective/laravelcollective.com
- https://github.com/Crinsane/LaravelShoppingcart
- https://github.com/cartalyst/stripe-laravel
- https://github.com/laravel/scout
- https://github.com/algolia/algoliasearch-client-php
- https://github.com/bradcornford/Googlmapper
- https://github.com/stevenschobert/instafeed.js/
- https://github.com/jordanmiguel/laravel-popular
- https://github.com/barryvdh/laravel-debugbar
- https://github.com/dingo/api
- https://github.com/tymondesigns/jwt-auth
- https://github.com/thephpleague/fractal
- https://github.com/RyanDaDeng/laravel-google-recaptcha-v3

Third Party Libraries που χρειάστηκα για την android react native εφαρμογή.

- https://github.com/yasaricli/react-native-counters
- https://github.com/sbycrosz/react-native-credit-card-input
- https://github.com/react-native-training/react-native-elements
- https://github.com/GeekyAnts/NativeBase
- https://github.com/archriss/react-native-snap-carousel
- https://github.com/leecade/react-native-swiper
- https://github.com/react-navigation/react-navigation
- https://github.com/reduxjs/react-redux

2.2 Θεωρίες

2.2.1 Ηλεκτρονικό εμπόριο

2.2.1.1 Τι είναι το ηλεκτρονικό εμπόριο

Ηλεκτρονικό εμπόριο θεωρητικά είναι η αγοροπωλησία αγαθών και υπηρεσιών ή η μετάδοση κάποιων δεδομένων η κάποιον κεφαλαίων μέσω του διαδικτύου. Σε ένα απλό ηλεκτρονικό εμπόριο είναι η αγορά αγαθών η υπηρεσιών μεταξύ η εμπόρων , προμηθευτών η καταναλωτών. Το ηλεκτρονικό εμπόριο χωρίζετε σε 8 κατηγορίες και θα αναφέρω λίγα πράγματα για την κάθε μια.

Business to Customer (B2C) – Επιχείρηση προς Πελάτη

Είναι το εμπόριο μεταξύ επιχειρήσεων και καταναλωτών. Περιλαμβάνει τους καταναλωτές που αγοράζουν προϊόντα ή υπηρεσίες από τα ηλεκτρονικά καταστήματα μέσω των ηλεκτρονικών μέσων και ονομάζεται δημοφιλές ηλεκτρονικό εμπόριο, το οποίο είναι έναν από τους γρηγορότερους και αναπτυσσόμενους τομείς της οικονομίας.

Business to Business (B2B) – Επιχείρηση προς Επιχείρηση

Σε αυτό, οι συναλλαγές πραγματοποιούνται μεταξύ εταιρείας σε εταιρεία ή μεταξύ επιχειρήσεων και χονδρεμπόρου. Σύμφωνα με την έρευνα, περίπου το 80% του ηλεκτρονικού εμπορίου στον κόσμο ασχολείται με το ηλεκτρονικό εμπόριο μεταξύ επιχειρήσεων, με αποτέλεσμα πολλές εταιρείες να είναι πολύ καλύτερες και πιο κερδοφόρες, όπου μπορούν να προωθήσουν γρήγορα τα προϊόντα τους.

Customer to Business (C2B) – Πελάτη προς Επιχείρηση

Ο καταναλωτής θα προσφέρει υπηρεσίες ή προϊόντα στις εταιρείες και οι εταιρείες θα πληρώσουν για τις προσφορές του καταναλωτή, κάτι που είναι εντελώς διαφορετικό από το παραδοσιακό μέχρι τώρα Internet Marketing. Μπορούμε να βρούμε αυτού του είδους **C2B συναλλαγές** σε blogs, forums κλπ. όπου ο διαχειριστής κατευθύνει τον αναγνώστη στην ιστοσελίδα του για αγορές.

Customer to Customer (C2C) – Πελάτη προς Πελάτη

Στις συναλλαγές αυτές, πραγματοποιείται πώληση μεταξύ πελάτη με πελάτη. Συνήθως αυτό το είδος πωλήσεων διευκολύνεται από μια ιστοσελίδα τρίτων που παράγει τα στοιχεία συναλλαγής. Αυτού του είδους οι συναλλαγές ωφελούν τόσο τον πωλητή όσο και τον αγοραστή, επειδή δεν υπάρχει διαμεσολαβητής. Έτσι δεν θα υπάρξουν και επιπλέον χρεώσεις. Πρόκειται για μια **ιδιωτική συναλλαγή** μεταξύ δύο ατόμων.

Business to Government (B2G) – Επιχείρηση προς Κυβέρνηση

Γενικά, έτσι ορίζεται το εμπόριο μεταξύ μιας επιχειρήσεις με τον δημόσιο τομέα. Η εταιρεία προσφέρει τις υπηρεσίες ή τα προϊόντα της σε διάφορους κυβερνητικούς τομείς, συμπεριλαμβανομένων των ομοσπονδιακών, κρατικών και τοπικών κυβερνήσεων, ειδικά για δημόσιες συμβάσεις, διαδικασίες δανειοδότησης, αιτήσεις για προσφορές ή πληροφορίες, προσφορές και άλλες κυβερνητικές πράξεις.

Government to Business (G2B) – Κυβέρνηση προς Επιχείρηση

Σε αυτό το μοντέλο, η Κυβέρνηση παρέχει πληροφορίες ή υπηρεσίες στον επιχειρηματικό οργανισμό, όπως είναι οι δημοπρασίες, οι προσκλήσεις υποβολής προσφορών, οι λειτουργίες υποβολής αιτήσεων κ.ά..

Government to Customer (G2C) – Κυβέρνηση προς Πελάτη

Κύριος στόχος αυτού του εμπορίου, είναι να παρέχει τις καλύτερες και αποτελεσματικότερες υπηρεσίες μέσω αντίστοιχων ιστοσελίδων προς κάθε πολίτη- πελάτη, όπως διάφορα συστήματα κοινωνικής πρόνοιας, πληροφορίες σχετικά με κυβερνητικά τμήματα ή απαραίτητες αιτήσεις που θα χρησιμοποιηθούν από τους πολίτες.

Mobile Commerce (M- Commerce) – Κινητό Εμπόριο

Το **Κινητό Εμπόριο** είναι η αγορά και η πώληση αγαθών ή υπηρεσιών, κυρίως μέσω των κινητών τηλεφώνων. Μια έρευνα σχετικά με το κινητό εμπόριο δείχνει ότι πάνω από το **70**% των ηλεκτρονικών συναλλαγών γίνεται μέσω κινητών τηλεφώνων. Τα καλύτερα παραδείγματα είναι το mobile banking, η πώληση εισιτηρίων και οι πληρωμές μέσω κινητού τηλεφώνου, vouchers, διάφορα κουπόνια κ.ά..

2.2.1.2 Δυνατότητες, Πλεονεκτήματα και Μειονεκτήματα του Ηλεκτρονικού εμπορίου

Μερικές από τις βασικές δυνατότητες του ηλεκτρονικού εμπορίου θα δούμε παρακάτω:

- Αναζήτηση προϊόντων ανά κατηγορία / είδος.
- Διαφορετικά είδη συναλλάγματος.
- Ο πελάτης έχει τη δυνατότητα να εγγραφεί με τα στοιχεία του και με τη χρήση email & password να βλέπει το αρχείο των παραγγελιών
- Υπολογισμός φόρων.
- Υπολογισμός εξόδων αποστολής.

- Shopping Cart.

Τα πλεονεκτήματα του ηλεκτρονικού εμπορίου:

- Παγκόσμια Εμβέλεια
- Αγορές οποιαδήποτε ώρα και ημέρα – 24/7
- Ψώνια από το σπίτι με άνεση
- Καλύτερες προσφορές ή εκπτώσεις για τους πελάτες
- Δυνατότητα να λαμβάνετε προσφορές μετρητών
- Αγορές χωρίς ταλαιπωρία, άμεσα και εύκολα
- Χαμηλότερο λειτουργικό κόστος για την επιχείρηση
- Πληρωμή χωρίς μετρητά
- Υποστήριξη Πελατών
- Διαχείριση αποθεμάτων
- Διαφήμιση & Marketing
- Secure e-paid

Τα μειονεκτήματα του ηλεκτρονικού εμπορίου:

- Αμφιβολία ως προς την αυθεντικότητα και την ασφάλεια
- Δεν υπάρχει η δυνατότητα χειρισμού πριν από την αγορά
- Δυσκολία στην επιστροφή προϊόντων
- Βλάβη των προϊόντων κατά τη μεταφορά
- Μπορεί να χαθούν τα προϊόντα\

2.2.1.3 Πληρωμές στο ηλεκτρονικό εμπόριο

Εισαγωγή

Όπως το ηλεκτρονικό εμπόριο εξελίχτηκε και αναπτύχθηκε ραγδαίες έτσι έπρεπε να εξελίξουμε και τους τρόπους πληρωμής σε κάτι πιο αυτοματοποιημένο καθώς ο πωλητής με τον αγοραστή μπορεί να βρίσκετε σε κάποια απόσταση οπότε έχουμε κάποιος νέους τρόπους που θα κάνουμε το ηλεκτρονικό εμπόριο πιο γρήγορο από το να πληρώσουμε σε μετρητά η σε κάποια επιταγή. Θα αναφέρω ονομαστικά μερικούς από τους πιο συνήθη τρόπους ηλεκτρονικής πληρωμής και παρακάτω θα αναφερθώ στον καθένα ξεχωριστά. Πλέον κάνουμε αγορές μέσω αντικαταβολής, πιστωτικής κάρτας, χρεωστικής κάρτας, PayPal, google pay, apple pay, εμβάσματος μέσω τραπέζης και ακόμα και σε πληρωμή με κρυπτό νομίσματα.

Πληρωμή με αντικαταβολή

Μια λύση αρκετά ασφαλής για τους δύσπιστους πελάτες περί ασφαλείας είναι ο τρόπος με αντικαταβολή ο όποιος είναι ο κλασικό τρόπος που μπορείς να κανείς μια ηλεκτρονική αγορά και να πληρώσεις με μετρητά όταν το προϊόν / αγαθό φτάσει στην πόρτα σου έτσι είσαι σίγουρος ότι ήρθε και πληρώνεις με μετρητά την αγορά σου. Βέβαια σε αυτών τον τρόπο υπάρχουν τα πλεονεκτήματα και τα μειονεκτήματα ένα πλεονέκτημα αναφέρθηκε πιο πριν και είναι το θέμα

ασφάλειας το οποίο ο κάθε καταναλωτής φοβάται για την αυθεντικότητα και την ασφάλεια του κάθε προϊόντος. Παρόλα αυτά ο πωλητής έχει το μειονέκτημα ότι μπορεί να καθυστερήσει η πληρωμή λόγω την καθυστέρησης από την εταιρία που θα παραδώσει το αγαθό στον πελάτη γιατί στην προκειμένη περίπτωση ο καταναλωτής είναι ο τρίτος έτσι πληρώνει την μεταφορική και η μεταφορική πληρώνει τον πωλητή.

Κατάθεση εμβάσματος σε τραπεζικό λογαριασμό

Σε όλες τις τράπεζες όταν κάνουμε μια τραπεζική συναλλαγή μέσο κάρτας τότε η τράπεζα κρατάει ένα μικρό ποσό το οποίο το ονομάζει έξοδα συναλλάγματος και είναι η προμήθεια τις κάθε τράπεζας. Κάποιες τράπεζες εμφανίζουν μηνιαία ένα ενημερωτικό σημείωμα με την κίνηση της κάρτας και τα επιπλέον έξοδα προμηθειών ενώ άλλες το προσθέτουν στην μετατροπής του σε ευρώ.

Κάτι τελευταίο είναι ότι η κάθε τράπεζα έχει διαφορετικές πολιτικές περί του θέματος αυτού και διαφορετικό κόστος προμηθειών το οποίο αναγράφετε στην ιστοσελίδα της κάθε τράπεζας και βρίσκετε στους όρους.

Πληρωμή με κάρτα

Προπληρωμένες κάρτες

Ένα μεγάλο μέσω όρο πληθυσμού πιστεύει και θεωρεί ότι οι προπληρωμένες κάρτες είναι η πιο ικανοποιητική μέθοδος για τον πελάτη γιατί προσθέτει κάθε φορά το πόσο που θέλει να σπαταλήσει και τίποτα παραπάνω. Ο τρόπος αυτός δεν σε υποχρεώνει να κανείς σύνδεση με κάποιο λογαριασμό απλά μόνο μια προμήθεια που τραβάει η κάθε τράπεζα η οποία διαφέρει από τράπεζα σε τράπεζα.

Πιστωτικές κάρτες

Οι κάρτες αυτές από το 2015 έχουν καθιερώσει έναν υποχρεωτικό τρόπο συναλλαγών και ταυτόχρονα σύγχρονο για την εποχή που είμαστε και μπορούμε να της εκδώσουμε σε οποιαδήποτε τράπεζα θελήσουμε. Αυτές οι κάρτες βγήκαν στην αγορά για να μας λύσουν τα χέρια και να κάνουμε αγορές με εικονικό χρήμα το οποίο χρήμα έχει όριο μηνιαίο ώστε να σπαταλάμε όλοι μας μέχρι ένα πόσο.

Χρεωστικές κάρτες

Αυτού του είδους οι κάρτες είναι πανομοιότυπες με της πιστωτικές με μια κυρία διάφορα ότι το πόσο της αγοράς μεταφέρετε απευθείας από τον λογαριασμό του κατόχου στο λογαριασμό του πωλητή και τέλος αυτές οι κάρτες είναι συνδεδεμένες με έναν λογαριασμό κατάθεσης στην τράπεζα. Στις εν λόγω κάρτες δεν γίνετε η πληρωμή εάν το πόσο του λογαριασμού μας δεν είναι διαθέσιμο. Ανοίγοντας ένα τραπεζικό λογαριασμό σε μια τράπεζα, αυτή μας παρέχει τη χρεωστική κάρτα είτε για ανάληψη χρημάτων, μέσω του ATM, είτε για αγορές ειδών και υπηρεσιών.

Apple Pay

Το Apple Pay είναι μια υπηρεσία ηλεκτρονικής πληρωμής που πραγματοποιείτε από ένα iOS κινητό τηλέφωνο και χαρακτηρίζετε ως ψηφιακό πορτοφόλι της Apple Inc. που επιτρέπει στους χρήστες να πραγματοποιούν πληρωμές προσωπικά, σε εφαρμογές iOS και στον ιστό. Υποστηρίζεται από το iPhone, το Apple Watch, το iPad και το Mac. Ψηφιοποιεί και μπορεί να αντικαταστήσει μια συναλλαγή chip και PIN με πιστωτική ή χρεωστική κάρτα σε ένα τερματικό σημείο εξυπηρέτησης χωρίς επαφή. Η Apple Pay δεν απαιτεί τερματικούς σταθμούς επαφής χωρίς πληρωμή της Apple Pay. Λειτουργεί με οποιονδήποτε έμπορο που δέχεται πληρωμές χωρίς επαφή. Είναι πολύ παρόμοια με τις επαφές χωρίς επαφή που έχουν ήδη χρησιμοποιηθεί σε πολλές χώρες, με την προσθήκη επαλήθευσης δύο παραγόντων μέσω του Touch ID, του Face ID, του PIN ή του κωδικού πρόσβασης. Η υπηρεσία επιτρέπει στις συσκευές Apple να επικοινωνούν ασύρματα με συστήματα σημείων πώλησης χρησιμοποιώντας μια κεραία επικοινωνίας κοντά στον τόπο (NFC), ένα αποκλειστικό τσιπ που αποθηκεύει κρυπτογραφημένες πληροφορίες πληρωμής (γνωστό ως Secure Element) και το Touch ID και το Πορτοφόλι της Apple.

Google Pay

Το Google Pay είναι μια πλατφόρμα ψηφιακού πορτοφολιού και ένα ηλεκτρονικό σύστημα πληρωμών που αναπτύχθηκε από την Google για την τροφοδοσία αγορών μέσω εφαρμογών και κλήσεων από κινητές συσκευές, με δυνατότητα πληρωμών μέσω χρηστών Android τηλέφωνα, ταμπλέτες ή ρολόγια.

Από τις 8 Ιανουαρίου 2018, το παλιό Android Pay και το Πορτοφόλι Google έχουν ενοποιηθεί σε ένα ενιαίο σύστημα αμοιβών που ονομάζεται Google Pay. Το Android Pay μετονομάστηκε και μετονομάστηκε ως Google Pay. Επίσης, ανέλαβε την επωνυμία της δυνατότητας αυτόματης συμπλήρωσης του Google Chrome. Το Google Pay υιοθετεί τις λειτουργίες τόσο του Android Pay όσο και του Πορτοφολιού Google μέσω των υπηρεσιών πληρωμών μεταξύ των καταστημάτων, του ομότιμου και των online πληρωμών.

2.2.2 Στατικές Ιστοσελίδες

2.2.2.1 Γενικά

Πρόκειται για τις δύο κατηγορίες ιστοσελίδων στο διαδίκτυο και αυτές είναι οι στατικές και οι δυναμικές.

Στην εμφάνιση τους είναι σχεδόν παρόμοιες και δεν διακρίνονται μεγάλες αλλαγές αλλά η μεγάλη διάφορα τους είναι η λειτουργικότητα των δύο αυτών σελίδων γιατί στις δυναμικές σελίδες στέλνουμε ένα αίτημα στον διακομιστή ώστε να μας απαντήσει και να μας φέρει δυναμικά τα δεδομένα από την βάση δεδομένων ενώ στην στατική σελίδα απλά τα δεδομένα μας βρίσκονται στην HTML και για να γίνουν αλλαγές πρέπει να προσθέσουμε έξτρα σημεία κώδικα.. Τώρα οι δυναμικές σελίδες χωρίζονται σε διάφορα είδη όπως απλά δυναμική σελίδα που φέρνει δεδομένα από την βάση αλλά υπάρχει και η άλλη κατηγορία που είναι συστήματα διαχείρισης περιεχομένου τα λεγόμενα CMS που θα αναλύσουμε και θα πούμε πιο κάτω τη ακριβώς είναι.

2.2.2.2 Στατικές ιστοσελίδες

Οι στατικές ιστοσελίδες, είναι ιστοσελίδες που περιέχουν μόνο τις βασικές πληροφορίες που πρέπει να υπάρχει σε μια σελίδα. Είναι δηλαδή σελίδες που μπορούν να περιέχουν φωτογραφίες, βίντεο, ήχο και κείμενο το οποίο είναι ίδιο για κάθε χρήστη του site χωρίς να έχει την δυνατότητα για κάποια αλλαγή, τις αλλαγές θα μπορεί να της κάνει μόνο ο διαχειριστής της σελίδας ή κάποιο άτομο που θα έχει τις απαραίτητες γνώσεις για να παρεμβεί στον κώδικα της σελίδας. Αυτό δεν είναι καθόλου πρακτικό επειδή θα πρέπει τα περιεχόμενα να ανανεώνονται συνεχώς. Πλέον υπάρχον δυναμικές ιστοσελίδες με πολύ ωραία σχεδίαση και με αρκετές δυνατότητες.

Πλεονεκτήματα-Μειονεκτήματα στατικών ιστοσελίδων

Μερικά πλεονεκτήματα που αφορούν τις στατικές ιστοσελίδες είναι:

- Χαμηλό κόστος κατασκευής
- Φορτώνουν πιο γρηγορά στον περιηγητή
- Μια στατική ιστοσελίδα σχεδιάζεται πιο γρηγορά
- Είναι φιλικές προς τις μηχανές αναζήτησης (SEO Friendly) έχοντας υψηλότερη κατάταξη σε σχέση με τις δυναμικές
- Δυνατότητα αναβάθμισης σε δυναμικές

Μερικά μειονεκτήματα που αφορούν τις στατικές ιστοσελίδες είναι:

- Θα πρέπει να υπάρχουν γνώσεις προγραμματισμού για την ανανέωση του περιεχόμενου
- Υψηλό κόστος για ανανέωση του περιεχομένου
- Δεν υπάρχει αλληλεπίδραση με τους χρήστες
- Δεν υπάρχει αυτόματη ανανέωση
- Μη ελκυστικές προς τους χρήστες

2.2.3 Δυναμικές Ιστοσελίδες

Οι δυναμικές ιστοσελίδες ,στην εμφάνιση μπορεί να μην έχουν μεγάλες διάφορες με τις στατικές, αλλά οι δυναμικές έχουν περισσότερες δυνατότητες γιατί ουσιαστικά πρόκειται για μια πλατφόρμα και όχι για μια απλή ηλεκτρονική σελίδα.

Ο σκοπός μιας δυναμικής σελίδας είναι να μπορεί ο χρήστης να αλληλοεπιδρά με την ιστοσελίδα ώστε ο ίδιος να μπορεί να επιλέξει τις ενέργειες που θα εφαρμόσει και το τι περιεχόμενο θα θέλει να προβάλει η ιστοσελίδα του. Την ίδια διαδικασία θα μπορούν να κάνουν πολλαπλοί χρήστες και ο καθένας τους θα έχει διαφορετικά δικαιώματα και θα μπορεί να προσθέσει διαφορετικό περιεχόμενο από κάποιον άλλων χρήστη.

Οι δυναμικές ιστοσελίδες χρησιμοποιούν μια βάση δεδομένων οπού αποθηκεύουν και επεξεργάζονται πληροφορίες.

Πλεονεκτήματα-Μειονεκτήματα δυναμικών ιστοσελίδων

Μερικά πλεονεκτήματα που αφορούν τις δυναμικές ιστοσελίδες είναι:

- Ο χρήστης μπορεί να αλληλοεπιδρά με τη σελίδα
- Δυνατότητα καταχώρησης απεριορίστου περιεχομένου
- Είναι απλό στη διαχείριση
- Δυνατότητα άμεσης και εύκολης τροποποίησης των περιεχομένων της σελίδας
- Είναι φιλικές στις μηχανές αναζήτησης (SEO)
- Δυνατότητα η δυναμική σελίδα να έχει μηδενικό κόστος ανανέωσης
- Χρήση του συστήματος διαχείριση περιεχομένου

Μερικά πλεονεκτήματα που αφορούν τις δυναμικές ιστοσελίδες είναι:

- Πολύ υψηλό κόστος κατασκευής
- Θέλει αρκετά μεγάλο χώρο για φιλοξενία
- Θέλει χρόνο για να κατασκευαστή
- Αργή η φόρτωση στον περιηγητή

2.2.4 Domain Names

2.2.4.1 Γενικά

Το όνομα διακομιστή είναι μια πιο απλή γραφή μιας IP διεύθυνσης, ενός αριθμού σαν αυτόν "127.0.0.1" η οποία είναι καταχωρημένη σε έναν πίνακα domain name service (DNS) και είναι μοναδική και αντιστοιχεί σε μια ιστοσελίδα στο διαδίκτυο. Κάθε μια ιστοσελίδα στο διαδίκτυο μπορεί να είναι προσπελάσιμη είτε με το domain name της είτε με την IP της.

2.2.4.2 Πως χρησιμοποιώ τα domain name;

Για να χρησιμοποιήσω ένα domain name πρέπει να ανοίξω έναν browser ο όποιος είναι εγκατεστημένος σε όλους τους υπολογιστές και σε όλες της συσκευές κινητά και ταμπλέτες καθώς μπορώ να το χρησιμοποιήσω και σε άλλες υπηρεσίες όπως (ftp, telnet, mail κ.ά.). Κάποιοί γνωστοί και συνήθεις browser είναι ο Firefox, ο Chrome, ο Opera, ο Safari για τα iPhone, iPads, Macs και ο κλασικός Microsoft Edge που υπάρχει όταν ανοίγεται ένα νέο υπολογιστή ακόμα και σε κινητά και ταμπλέτες υπάρχουν ακόμα κάποιοι π.χ. Samsung Browser για όλες της Samsung συσκευές κ.α.

2.2.4.3 Καταλήξεις στα domain name

Σε όλα τα domain names έχουμε δύο διαφορετικά μέρη, την κατάληξη και το πρόθεμα. Στην διεύθυνση google.com, το google είναι το πρόθεμα και το com είναι κατάληξη.

Στο πρόθεμα ενός domain name μπορούμε να έχουμε αριθμούς, γράμματα και παύλες. Όλα τα υπόλοιπα σύμβολα δεν μπορούν να εισαχθούν στο πρόθεμα. Και τέλος σε ένα domain name μπορούμε να εισάγουμε και ελληνικούς χαρακτήρες.

Τώρα στις μέρες μας υπάρχουν πολλές καταλήξεις γνωστές και κάποιες όχι τόσο συνήθεις. Μια από τις πιο γνωστές καταλήξεις στην Ελλάδα είναι η .gr, καθώς και ανάλογα την χώρα έχουμε ίδιες καταλήξεις παράδειγμα για την Γαλλία υπάρχει το .fr, ενώ κάποιες παλιές

καταλήξεις είναι οι (.com, .info, .net, .org, .eu). Στα παρακάτω site μπορείτε να δείτε τις διαθέσιμες καταλήξεις που υπάρχουν αυτήν την στιγμή στην αγορά : GoDaddy.com, Papaki.com, domains.google κ.α.

2.2.5 Λογισμικό και εφαρμογές Android

2.2.5.1 Τι είναι android;

Το Android είναι ένα OS το οποίο χρησιμοποιείται από συσκευές κινητής τηλεφωνίας και τρέχει τον πυρήνα του λειτουργικού Linux. Η Google ήταν αυτή που πρώτα ασχολήθηκε με το Android, ενώ μετέπειτα ανέλαβε η Open Handset Alliance. Με το σύστημα αυτό οι κατασκευαστές λογισμικού έχουν τη δυνατότητα να συνθέτουν κώδικα χρησιμοποιώντας τη Java ως γλώσσα προγραμματισμού, έχοντας τον έλεγχο της συσκευής μέσω, των ανεπτυγμένων από την Google, βιβλιοθηκών λογισμικού.

Το Android κατασκευάστηκε κυρίως για συσκευές με οθόνη αφής, τέτοιες όπως τα smartphones και τα tablets. Ωστόσο έχει ακόμα ανεπτυγμένα διαφορετικά περιβάλλοντα για τις τηλεοράσεις (Android TV), τα αυτοκίνητα (Android Auto) καθώς και τα ρολόγια χειρός (Android Wear) αντίστοιχα. Παρόλο που δημιουργήθηκε για συσκευές με οθόνη αφής, το Android χρησιμοποιήθηκε και χρησιμοποιείται από πολλές ηλεκτρονικές συσκευές όπως, οι κονσόλες παιχνιδιών, συνηθισμένους Η/Υ (πχ. το HP Slate 21), ψηφιακές φωτογραφικές μηχανές και άλλες.

Το Android θεωρείται το πιο διαδεδομένο λειτουργικό σύστημα στον κόσμο, ενώ οι πωλήσεις των συσκευών που το χρησιμοποιούν ξεπερνούν κατά πολύ αυτές των άλλων λογισμικών (πχ. Windows, IOS, Mac OS X).

Η παρουσίαση της πλατφόρμας Android έγινε για πρώτη φορά στις 5 Νοεμβρίου 2007, ενώ την ίδια μέρα ανακοινώθηκε και η ίδρυση του οργανισμού Open Handset Alliance. Στόχος του οργανισμού αυτού είναι η συνεργασία 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού και κατασκευής hardware με σκοπό να αναπτύξουν και να εξελίξουν τα ανοιχτά πρότυπα στις συσκευές κινητής τηλεφωνίας. Η Google η οποία, όπως και προαναφέρθηκε, ασχολήθηκε πρώτη με το Android, κοινοποίησε το μεγαλύτερο μέρος του κώδικα του, ακολουθώντας τους όρους της Apache License, ελεύθερης άδειας λογισμικού.

Τέλος το λογότυπο του λειτουργικού συστήματος το ανέλαβε η γραφίστρια Ιρίνα Μπλόκ και απεικονίζει ένα ρομπότ σε χρώμα πράσινου μήλου.

Η ιστορία εκδόσεων του Android του λειτουργικού συστήματος των κινητών ξεκίνησε με την κυκλοφορία του Android beta το Νοέμβριο του 2007. Η πρώτη εμπορική έκδοση ήταν το Android 1.0 που κυκλοφόρησε το Σεπτέμβριο του 2008. Το Android είναι υπό συνεχή ανάπτυξη από την Google και την Open Handset Alliance (OHA), και έχουν γίνει μια σειρά από ενημερώσεις στην λειτουργία του συστήματος από την αρχική κυκλοφορία του.

Από τον Απρίλιο του 2009, οι εκδόσεις του Android έχουν θέμα από την ζαχαροπλαστική στην κωδική ονομασία τους, και κυκλοφόρησαν σε αλφαβητική σειρά, εξαιρουμένων των εκδόσεων 1.0 και 1.1, που δεν τέθηκαν υπό συγκεκριμένα κώδικά ονόματα:

Κωδικό όνομα	**Νούμερο έκδοσης**	**Ημερομηνία αρχικής κυκλοφορίας**	**Επίπεδο API**
N/A	1.0	23 Σεπτεμβρίου 2008	1
	1.1	9 Φεβρουαρίου 2009	2
Cupcake	1.5	27 Απριλίου 2009	3
Donut	1.6	15 Σεπτεμβρίου 2009	4
Eclair	2.0 – 2.1	26 Οκτωβρίου 2009	5 – 7
Froyo	2.2 – 2.2.3	20 Μαΐου 2010	8
Gingerbread	2.3 – 2.3.7	6 Δεκεμβρίου 2010	9 – 10
Honeycomb	3.0 – 3.2.6	22 Φεβρουαρίου 2011	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	18 Οκτωβρίου 2011	14 – 15
Jelly Bean	4.1 – 4.3.1	9 Ιουλίου 2012	16 – 18
KitKat	4.4 – 4.4.4	31 Οκτωβρίου 2013	19 – 20
Lollipop	5.0 – 5.1.1	12 Νοεμβρίου 2014	21 – 22
Marshmallow	6.0 – 6.0.1	5 Οκτωβρίου 2015	23
Nougat	7.0 – 7.1.2	22 Αυγούστου 2016	24 – 25
Oreo	8.0 – 8.1	21 Αυγούστου 2017	26 – 27
Pie	9.0	6 Αυγούστου 2018	28

Πίνακας 2 (Android Versions)

2.2.5.2 Εφαρμογές android

Ένα μεγάλο εύρος στις κινητές συσκευές και ταμπλέτες που κυκλοφορούν στην αγορά είναι διαθέσιμες με ένα από τα πιο δημοφιλές OS της Google το οποίο ονομάζεται και ως Android σύστημα και εκτός από το ότι είναι ένα δημοφιλές λογισμικό που είναι έχει αναπτύξει μια μεγάλη γκάμα εφαρμογών είτε από ίδιους προγραμματιστές της Google είτε από free developers που δημιουργούν κάθε μέρα νέες υπηρεσίες και νέες εφαρμογές οι οποίες είναι διαθέσιμες να κατεβάσει κάθε χρήστης από το Google Store.

2.2.5.3 Εμπειρία χρήστη για εφαρμογές Android

Στην γραφική εμπειρία του χρήστη η προσθήκη ενός background, ενός animation η ενός ήχου την σωστή χρονική στιγμή αποτελούν πλεονέκτημα μιας εφαρμογής να γίνει πετυχημένη και φέρνει μια ικανοποίηση στον χρήστη που την χρησιμοποιεί.

Πρέπει να δοθεί μεγάλο βάρος στην διαδικασία του σχεδιασμού μια εφαρμογής γιατί είναι ο κυρίως σκοπός μιας εφαρμογής να αποκτήσει δημοτικότητα και να την κατεβάσουν αρκετοί

χρήστες ακόμα και πελάτες, μετά πρέπει να δοθεί σημασία στην λειτουργικότητα της εφαρμογής και να κάνει αυτοματοποιημένα πράγματα ώστε ο χρήστης να μείνει ικανοποιημένος με την χρήση της.

2.2.5.4 Γιατί να κατεβάσετε μία εφαρμογή Android;

- Για να αποκτήσετε όφελος πάνω στην μεγάλη ανάπτυξη που έχει αυτήν την στιγμή η πλατφόρμα της Google Android η οποία έχει αμέτρητες συσκευές καθημερινά και έχει άπειρους ανεξάρτητους προγραμματιστές που ανεβάζουν της ιδέες τους καθημερνά και έχει σαν αποτέλεσμα εκατομμύρια downloads εφαρμογών από την πλατφόρμα της Google .
- Για να έχετε τους πελάτες από κοντά σας ακόμα και όταν κινούνται και να μπορείτε να έχετε την γεωγραφική τους θέση ώστε να τους φέρετε κοντά σας στο φυσικό σας κατάστημα και όλα αυτά με την υπηρεσία της Google Maps που σας παρέχει η ίδια η Google.
- Για να εκμεταλλευτείτε τους διάφορους τρόπους ώστε να αναπτύξετέ και να δικτυωθείτε γρήγορα μέσω των νέων τεχνολογιών από όλα τα social media και από όλες της μεγάλες πλατφόρμες.

2.2.6 Λογισμικό Ανοικτού Κώδικα (Open Source Software)

2.2.6.1 Γενικά

Το λογισμικό ανοιχτού κώδικα είναι λογισμικό με πηγαίο κώδικα που μπορεί κανείς να επιθεωρήσει, να τροποποιήσει και να βελτιώσει.

Ο πηγαίος κώδικας είναι ένα κομμάτι του κώδικα που πολλοί διαφορετικοί χρήστες δεν μπορούν να δουν ποτέ. Αλλά μόνο κάποιος από τους προγραμματιστές αυτό του κώδικα μπορεί να παρέμβει και να αλλάξει κάποιο κομμάτι είτε αυτό είναι μια εφαρμογή είτε ένα πρόγραμμα. Καθώς οι προγραμματιστές είναι αυτοί που θα έχουν την πρόσβαση ώστε να αλλάξουν κάποια σημεία του πηγαίου κώδικα που ίσως να μην λειτουργούν σωστά η έστω τμήματα αυτού του κώδικα.

2.2.6.2 Ποια είναι η διαφορά του open source σε σχέση με άλλων τύπων λογισμικού;

Κάποιο από το τμήμα του πηγαίου κώδικα είναι ιδιόκτητο το οποίο και μπορεί μόνο το άτομο που το έχει δημιουργήσει είτε η ομάδα η ο οργανισμός να παρέμβει πάνω σε αυτό και να έχει τον απόλυτο έλεγχο του. Οι άνθρωποι ονομάζουν αυτό το είδος λογισμικού "ιδιόκτητο" ή "κλειστή πηγή" λογισμικό.

Η τροποποίηση και η αντιγραφή ενός ιδιόκτητου λογισμικού μπορεί νόμιμα μόνο από τους αρχικούς συγγραφείς να γίνει και όχι από κάποιον άλλο. Σε περίπτωση που κάποιος θέλει να χρησιμοποιήσει αυτόν τον ιδιόκτητο κώδικα τότε πρέπει να υπογραφτεί μια άδεια την πρώτη φορά οι οποία θα λέει ότι δεν μπορούν να κάνουν τίποτα με αυτό το ιδιόκτητο λογισμικό. Ένα παράδειγμα ιδιόκτητου λογισμικού είναι της Adobe το Photoshop και της Microsoft το Office.

Στην περίπτωση που το λογισμικό είναι ανοιχτού κώδικα τότε είναι διαφορετικό. Οι δημιουργοί του πηγαίου κώδικα τον κάνουν ανοιχτό σε όλο το κοινό που θα θελήσει να δει η να αντιγράψει αυτόν τον κώδικα η και να αλλάξει η να μοιραστεί κομμάτια του. Ένα παράδειγμα

ανοιχτού λογισμικού είναι το Gimp παρόμοιο σαν το Photoshop και το Libre Office παρόμοιο με της Microsoft το Office.

Θα δούμε ότι όπως γίνετε και με τον ιδιόκτητο κώδικα έτσι γίνετε και στον ανοιχτό κώδικα. Δηλαδή οι χρήστες του ανοιχτού κώδικα πρέπει να αποδέχονται τους όρους χρήσης όταν θα χρειαστούν τον ανοιχτό κώδικα όπως κάνουμε και στον ιδιόκτητο κώδικα μόνο που οι άδειες του ανοιχτού λογισμικού και του ιδιόκτητο διαφέρουν αρκετά μεταξύ τους.

Οι άδειες ανοιχτού λογισμικού αλλάζουν τον τρόπο που οι προγραμματιστές θα χρησιμοποιήσουν η θα ανταλλάξουν το λογισμικό. Γενικά, οι άδειες ανοιχτού κώδικα επιτρέπουν στους χρήστες ηλεκτρονικών υπολογιστών την άδεια χρήσης λογισμικού ανοιχτού κώδικα για οποιονδήποτε σκοπό επιθυμούν. Ορισμένες άδειες ανοιχτού κώδικα αυτό που ορισμένοι αποκαλούν άδειες "copyleft" ορίζουν ότι όποιος κυκλοφορεί ένα τροποποιημένο πρόγραμμα ανοικτού κώδικα πρέπει επίσης να απελευθερώσει τον πηγαίο κώδικα για το πρόγραμμα μαζί με αυτόν. Επίσης κάποιες open source άδειες λένε ότι οποιοσδήποτε προγραμματιστής αλλάξει η μοιραστεί έναν πηγαίο κώδικα του εκάστοτε λογισμικού δεν πρέπει να χρεώσει τέλη αδείας.

Το βασικό νόημα είναι ότι με την χρήση ανοιχτού κώδικα μπορεί να γίνει η ανταλλαγή δεδομένων οπότε μεταξύ προγραμματιστών υπάρχει μια συνεργασία γιατί δίνετε ίδιος κώδικας ώστε να τροποποιηθούν κομμάτια η να ενσωματωθούν καλύτερες λογικές λύσεις πάνω στο εκάστοτε project.

2.2.6.3 Γιατί να χρησιμοποιήσω λογισμικό ανοιχτού κώδικα;

Οι προγραμματιστές προτιμούν open source project σε ιδιόκτητο λογισμικό για διάφορους λόγους, όπως:

Έλεγχος. Αρκετοί προγραμματιστές προτιμούν τα open source project επειδή έχουν περισσότερο έλεγχο στο εκάστοτε προγεστερόνη. Επιβλέπουν τον κώδικα τον τροποποιούν αν κάτι δεν τους αρέσει και ενσωματώνουν μέρη από αλλά σημεία άλλων project. Επίσης τον συγκεκριμένο κώδικα ανοιχτού λογισμικού μπορούν να το χρησιμοποιήσουν και απλοί χρήστες που δεν είναι απαραίτητα προγραμματιστές.

Εκπαίδευση. Άλλοι άνθρωποι, όπως το λογισμικό ανοιχτού κώδικα, επειδή τους βοηθά να γίνουν καλύτεροι προγραμματιστές. Επειδή ο κώδικας ανοιχτού κώδικα είναι προσβάσιμος στο κοινό, οι σπουδαστές μπορούν εύκολα να το μελετήσουν, καθώς μαθαίνουν να κάνουν καλύτερο λογισμικό. Οι μαθητές μπορούν να κάνουν share τη δουλειά (Project) τους με πολλούς άλλους είτε προγραμματιστές είτε μαθητές, γράφοντας σχόλια και κριτικές, καθώς αναπτύσσουν το ταλέντο τους. Όταν οι άνθρωποι ανακαλύπτουν λάθη στον πηγαίο κώδικα των προγραμμάτων, μπορούν να μοιραστούν αυτά τα λάθη με άλλους για να τους βοηθήσουν να αποφύγουν να κάνουν αυτά τα ίδια λάθη.

Ασφάλεια. Ένα μεγάλο εύρος ανθρώπων δεν προτιμάει να χρησιμοποιεί ιδιόκτητο κώδικα γιατί δεν θεωρείτε ασφαλές και σταθερός σε σχέση με τον ανοιχτού λογισμικού κώδικα. Παράλληλα ένας προγραμματιστής μπορεί εύκολα να κάνει αλλαγές και διορθώσεις σε ένα open source

κώδικα που έχει ξεχάσει ο ιδιοκτήτης που τον έγγραψε. Τέλος σε ένα open source project μπορούν παράλληλα να δουλέψουν πολλοί προγραμματιστές μαζί να διορθώσουν και να κάνουν αλλαγές η έστω να δημιουργήσουν κάτι από την αρχή πολύ πιο γρήγορα από έναν ιδιόκτητο κώδικα λογισμικού.

Σταθερότητα. Ένα μεγάλο ποσοστό χρηστών τους βολεύει η χρήση open source σε σχέση με τον ιδιόκτητο κώδικα λόγο μεγάλων project. Επειδή οι προγραμματιστές διανέμουν δημοσίως τον πηγαίο κώδικα για το λογισμικό ανοιχτού κώδικα, οι χρήστες που βασίζονται σε αυτό το λογισμικό για κρίσιμες εργασίες μπορούν να είναι σίγουροι ότι τα εργαλεία τους δεν θα εξαφανιστούν ούτε θα υποστούν βλάβη εάν οι αρχικοί δημιουργοί τους σταματούν να δουλεύουν πάνω τους. Επιπλέον, το λογισμικό ανοικτής πηγής τείνει να ενσωματώνει και να λειτουργεί σύμφωνα με τα ανοικτά πρότυπα.

Κεφάλαιο 3: Εργαλεία Υλοποίησης και Εγκατάσταση

3.1 Ανάλυση και Εγκατάσταση IntelliJ IDEA

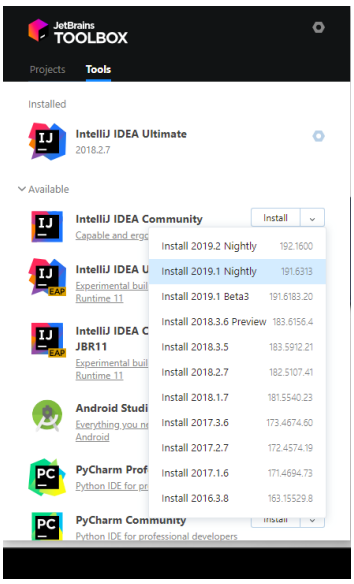
3.1.1 Ανάλυση IntelliJ IDEA

Το IntelliJ IDEA αναπτύχθηκε από το JetBrains, παλαιότερα γνωστό ως IntelliJ. Πρώτη κυκλοφορία έγινε το 2001 και χαρακτήρισε χαρακτηριστικά όπως προηγμένη πλοήγηση κώδικα και δυνατότητα κωδικών refactor, γεγονός που το έκανε πολύ δημοφιλές. Έλαβε ακόμη και τη διάκριση να ψηφιστεί το καλύτερο εργαλείο προγραμματισμού που βασίζεται στην Java το 2010, παραγκωνίζοντας καθιερωμένα εργαλεία όπως το NetBeans, το Eclipse και το JDeveloper. Το περιβάλλον ανάπτυξης ανοιχτού κώδικα για το Android που κυκλοφόρησε η Google το 2014 βασίζεται επίσης σε IntelliJ IDEA. Το IDE υποστηρίζει πολλές άλλες γλώσσες προγραμματισμού, όπως η Python, η Java, η JavaScript, η PHP, η Lua και η Scala.

Ο μεγαλύτερος λόγος που θεωρείται ως ένα από τα καλύτερα εργαλεία προγραμματισμού που βασίζονται στην Java είναι τα χαρακτηριστικά βοήθειας που το καθιστούν εύκολο στη χρήση και κάνει τα προγράμματα που δημιουργούνται από αυτό πολύ καλά σχεδιασμένα. Διαθέτει επίσης προηγμένες λειτουργίες ελέγχου σφαλμάτων, που επιτρέπει τον ταχύτερο και ευκολότερο έλεγχο σφαλμάτων.

3.1.2 Εγκατάσταση IntelliJ IDEA

Για να κάνουμε εγκατάσταση τον IDE θέλουμε πρώτα το installer.exe για το Toolbox app και μετά τρέχουμε τον wizard installer και κάνουμε εγκατάσταση ότι IDE θέλουμε όπως βλέπουμε στην παρακάτω εικόνα.



Εικόνα 1(Toolbox Install)

3.2 Ανάλυση και Εγκατάσταση Laragon

3.2.1 Ανάλυση Laragon

3.2.1.1 Τι είναι το Laragon;

Το Laragon είναι ένα φορητό, απομονωμένο, γρήγορο και ισχυρό περιβάλλον γενικής ανάπτυξης για PHP, Node.js, Python, Java, Go, Ruby. Είναι γρήγορο, ελαφρύ, εύκολο στη χρήση και εύκολο στην επέκταση.

Το Laragon είναι ιδανικό για την κατασκευή και διαχείριση σύγχρονων εφαρμογών ιστού. Εστιάζεται στην απόδοση - σχεδιασμένη γύρω από τη σταθερότητα, την απλότητα, την ευελιξία και την ελευθερία.

Το Laragon είναι πολύ ελαφρύ και θα παραμείνει όσο το δυνατόν πιο φτωχό. Το ίδιο το δυαδικό πυρήνα είναι μικρότερο από 2MB και χρησιμοποιεί λιγότερη από 4MB μνήμη RAM όταν εκτελείται.

Το Laragon δεν χρησιμοποιεί υπηρεσίες Windows. Διαθέτει τη δική του ενορχήστρωση υπηρεσιών, η οποία διαχειρίζεται τις υπηρεσίες ασύγχρονα και χωρίς αποκλεισμούς, ώστε να βρείτε τα πράγματα να τρέχουν γρήγορα και ομαλά με το Laragon.

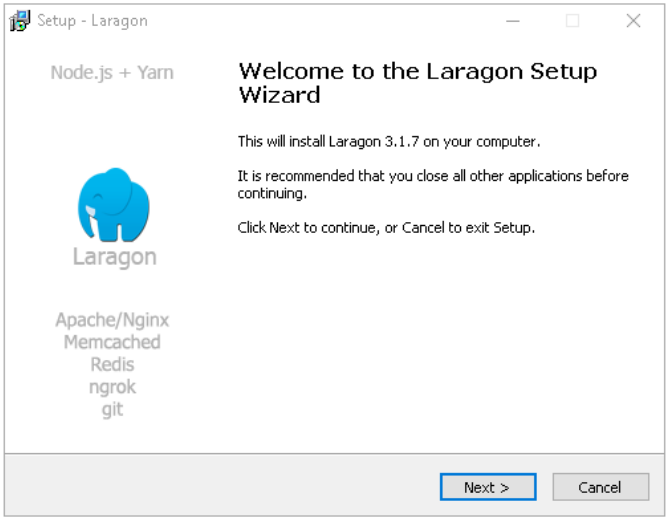
Χαρακτηριστικά

- **Εύκολα URLs**
  Use app.test instead of localhost/app
- **Φορητό**
  Μπορείτε να μεταφέρετε το φάκελο του Laragon (σε άλλους δίσκους, σε άλλους φορητούς υπολογιστές, συγχρονισμό στο Cloud) χωρίς καμία ανησυχία.

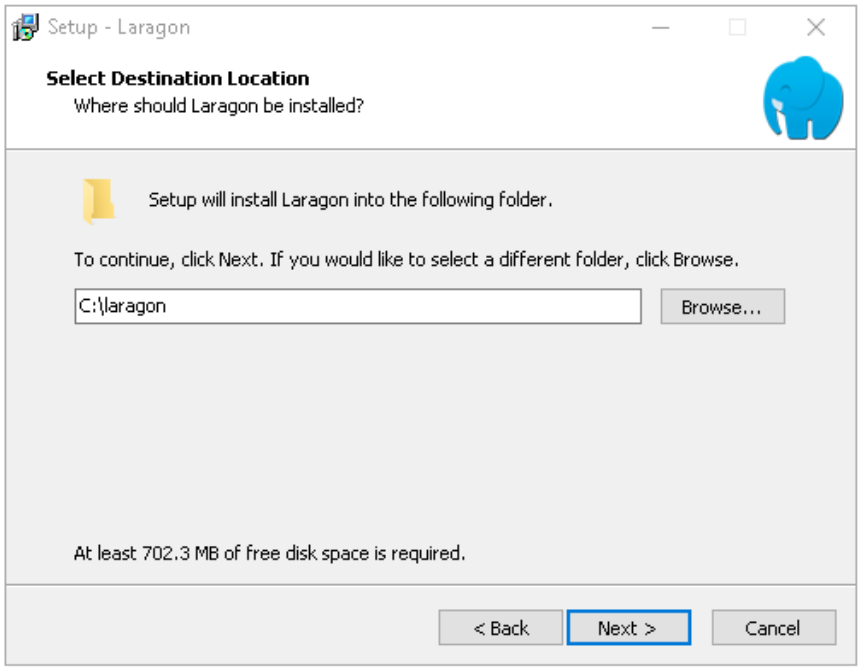
- **Απομονωμένο**
  Το Laragon διαθέτει ένα απομονωμένο περιβάλλον με το λειτουργικό σας σύστημα, θα κρατήσει το σύστημά σας καθαρό.
- **Εύκολη λειτουργία**
  Σε αντίθεση με άλλα το Laragon έχει εύκολα auto configurations που το καθιστά εύκολο στην χρήση του. Αυτός είναι ο λόγος για τον οποίο μπορείτε να προσθέσετε άλλες εκδόσεις των PHP, Java, Python, Ruby, Go, Apache, Nginx, PostgreSQL , MySQL και MongoDB χωρίς κόπο με απλά κλικ.
- **Σύγχρονο και Δυνατό**
  Ο Laragon έρχεται με μοντέρνα αρχιτεκτονική, η οποία είναι κατάλληλη για την κατασκευή σύγχρονων εφαρμογών ιστού. Μπορείτε να εργαστείτε και με τους δύο Apache & Nginx καθώς είναι πλήρως ελεγχόμενοι.
  Επίσης, το Laragon κάνει τα πράγματα πολύ πιο εύκολα:
  - Θέλετε να έχετε ένα WordPress CMS; Μόνο 1 κλικ.
  - Θέλετε να δείξετε το τοπικό σας έργο στους πελάτες; Μόνα 1 κλικ.
  - Θέλετε να ενεργοποιήσετε / απενεργοποιήσετε μια επέκταση PHP; Μόνο 1 κλικ

3.2.2 Εγκατάσταση Laragon

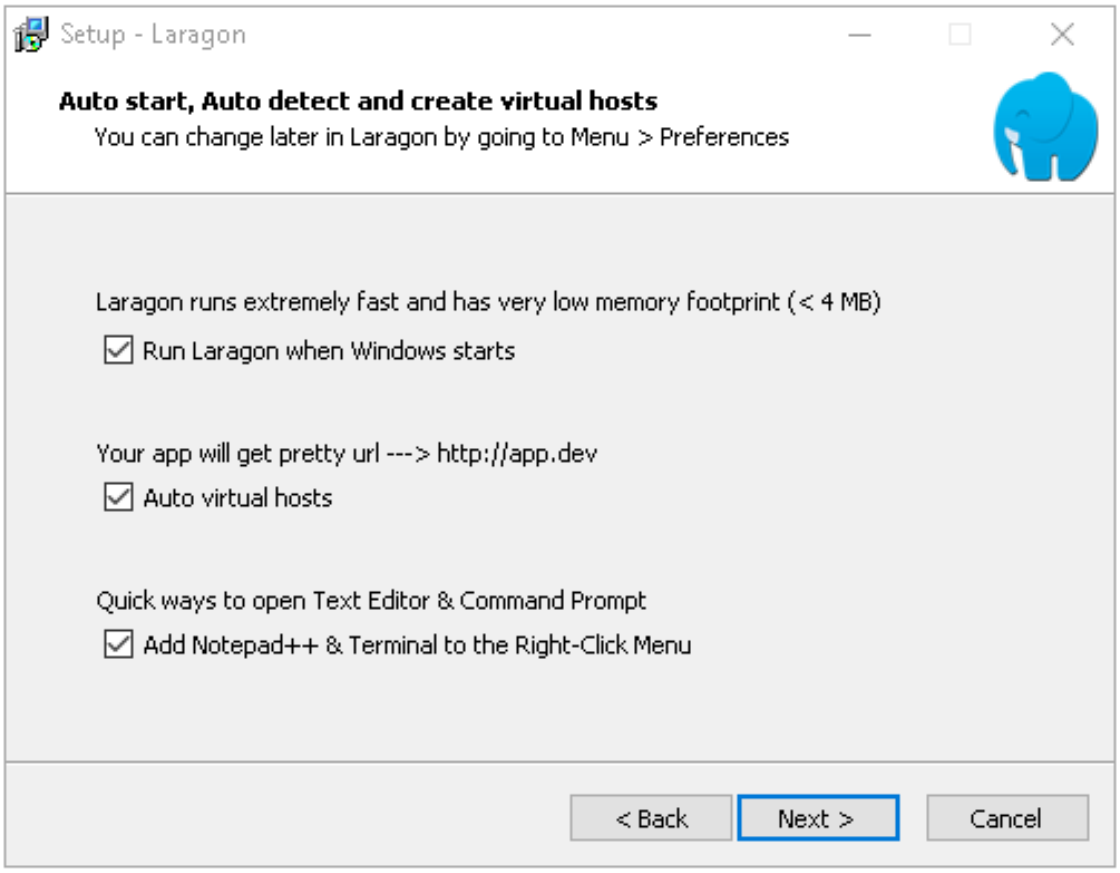
Για να ξεκινήσουμε με το Laragon κατεβάζουμε την τελευταία version από το site και πατάμε next, next, next όπως βλέπουμε στις παρακάτω εικόνες.



Εικόνα 2 (Laragon Setup - Welcome page)



Εικόνα 3 (Laragon Setup - Select location)



Εικόνα 4 (Laragon Setup - Options)

3.3 Ανάλυση και Εγκατάσταση Postman

3.3.1 Ανάλυση Postman

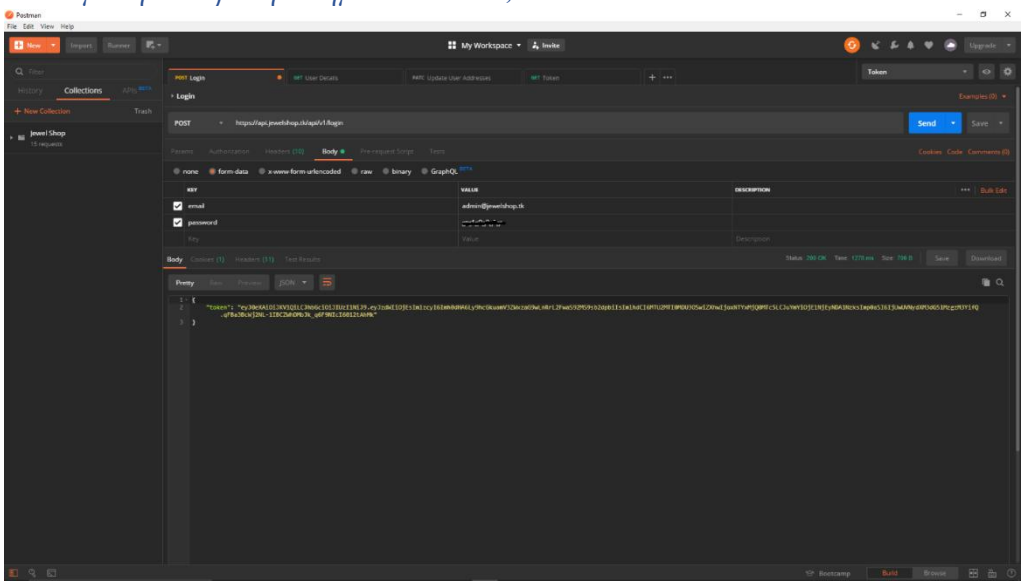
3.3.1.1 Τι είναι το Postman;

Το Postman είναι ένα εξαιρετικό εργαλείο όταν προσπαθείτε να αναλύσετε τα Restful API που γίνονται από άλλους ή τα δοκιμαστικά αυτά που έχετε κάνει εσείς. Προσφέρει μια κομψή διεπαφή χρήστη με την οποία μπορείτε να κάνετε αιτήματα HTML, χωρίς την ταλαιπωρία να γράφετε μια δέσμη κώδικα μόνο για να δοκιμάσετε τη λειτουργικότητα του API.

Ας υποθέσουμε ότι ήθελα να κάνω μια αίτηση GET εναντίον ενός fan-made API για το video game Hearthstone για να ψάξω κάρτες με "τοξότη" στο όνομά τους. Αν ήθελα να δοκιμάσω ένα αίτημα GET κατά αυτής της διαδρομής χωρίς να χρησιμοποιήσω τον Postman, αντί να γράψω κώδικα σε κάτι σαν το Flask. Σε αυτήν την περίπτωση θα έπρεπε να γράψω μια ολόκληρη νέα διαδρομή και λειτουργία για να εκτελέσω το αίτημα, τότε θα έπρεπε να καθορίσω με περισσότερη κωδικοποίηση αυτό που θέλω να μοιάσει η απάντηση και τελικά θα έπρεπε να εκτυπώσω την απάντηση στην κονσόλα ή να δώσω κάποιο άλλο τρόπο να βλέπω την απάντηση. Χρειάζεται να γράψω όλα αυτά ούτως ή άλλως για να φτιάξω μια λειτουργούσα εφαρμογή που χρησιμοποιεί αυτό το API, αλλά όλα αυτά για να δοκιμάσω απλώς τη λειτουργικότητα του API είναι άσκοπα κουραστική και χρονοβόρα όταν υπάρχει κάτι σαν το Postman.

Με τον Postman, μια τέτοια δοκιμή είναι πολύ πιο εύκολη. Το μόνο που έχω να κάνω είναι να συνδέσω τη διαδρομή στη γραμμή διευθύνσεων, να επιλέξω τη μέθοδο απόκρισης GET στο αναπτυσσόμενο πλαίσιο στα αριστερά της, να κλέψω το κλειδί API μου στην ενότητα "Κεφαλίδες", να διευκρινίσω ότι θέλω την απάντηση στην "μορφή" JSON, και να στείλω την αποστολή. Στη συνέχεια, λαμβάνω τα δεδομένα απόκρισης σε ευανάγνωστο JSON με κωδικό κατάστασης 200, επιβεβαιώνοντας ότι η αίτηση GET ήταν επιτυχής.

3.3.1.2 Τι γίνεται με την υποβολή αιτημάτων POST;



Εικόνα 5 (Postman API login)

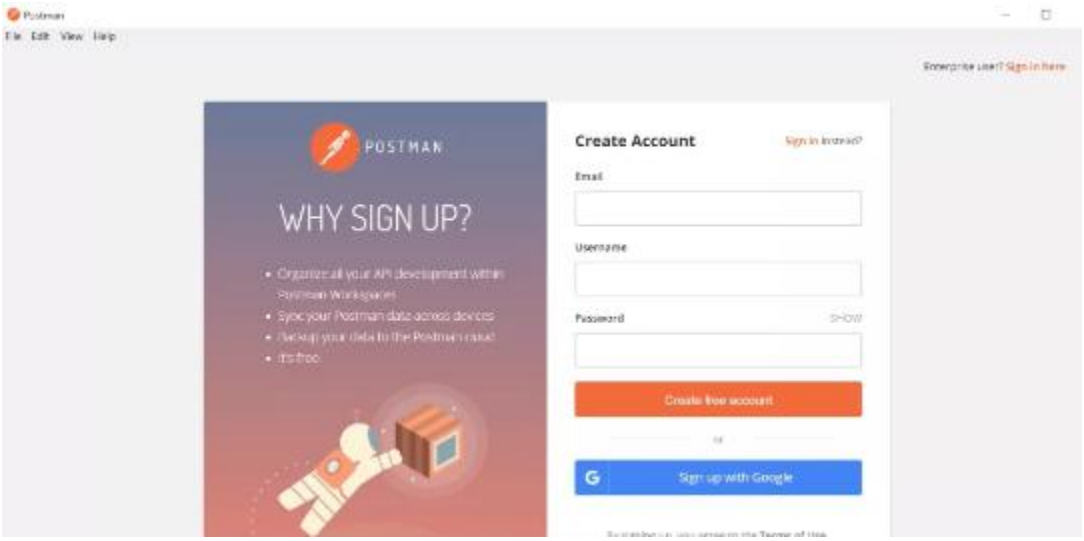
Στο παραπάνω παράδειγμα, έκανα ένα αίτημα POST στο https://api.jewelshop.tk/api/v1/login, ένα " API REST για την δοκιμή και το πρωτότυπο". Όπως και όταν κάνατε ένα αίτημα GET με τον Postman, προσθέστε τη διαδρομή στη γραμμή διευθύνσεων, αλλά αντ' αυτού της επιλογής GET στο αναπτυσσόμενο πλαίσιο, επέλεξα αντ' αυτού POST. Στο σώμα αιτήσεων, που έχει οριστεί ως "ακατέργαστο", εισήγαγα ένα ψεύτικο email και ένα password για να δούμε τι μας επιστρέφει το API.

Μπορείτε να δείτε κάτω από το σώμα αιτήσεων το σώμα απάντησης, το οποίο σε αυτήν την περίπτωση μου επιστρέφει το token ώστε να μπορώ να κάνω όλα τα αιτήματα στον διακομιστή με την μέθοδο του JSON Web Token την οποία θα αναφέρω στο κεφάλαιο API. Επίσης μου επέστρεψε και έναν κωδικό κατάστασης 200 που επιβεβαιώνει ότι έκανα ένα επιτυχημένο αίτημα POST.

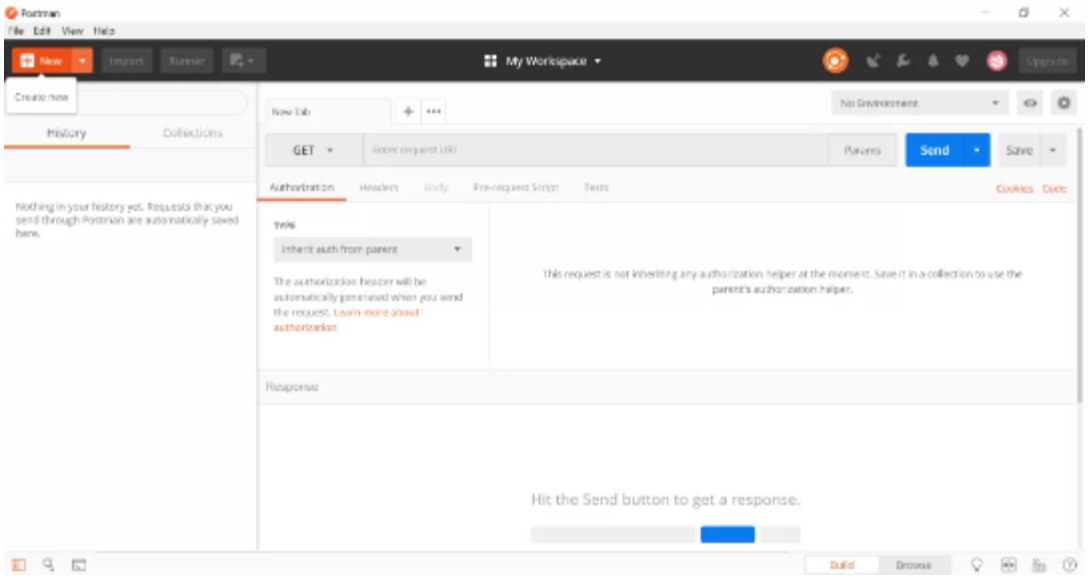
Ο Postman μπορεί να εκτελέσει PUT, PATCH, DELETE και διάφορες άλλες μεθόδους αίτησης καθώς επίσης και βοηθητικά προγράμματα για να βοηθήσει στην ανάπτυξη APIs. Οι δωρεάν και πληρωμένες εκδόσεις είναι διαθέσιμες για Mac, Windows, Linux, αλλά και ως εφαρμογή Chrome.

3.3.2 Εγκατάσταση Postman

Κατεβάζουμε από την ιστοσελίδα του Postman το installer και πατάμε next, next, next στην εγκατάσταση. Καθώς ανοίξουμε το πρόγραμμα θα μας ζητήσει να κάνουμε σύνδεση με λογαριασμό η να το δουλέψουμε σαν επισκέπτες, προχωράμε ανάλογα με το τι θέλουμε και παρακάτω βλέπουμε το γραφικό περιβάλλον της εφαρμογής.



Εικόνα 6 (Postman SignIn)



Εικόνα 7 (Postman Graphical Environment)

3.4 Ανάλυση και Εγκατάσταση Android SDK

3.4.1 Ανάλυση Android SDK

3.4.1.1 Τι είναι το Android Software Development Kit;

Η δημιουργία Android λογισμικού είναι όταν δημιουργούμε καινούριες εφαρμογές για κινητά τηλέφωνα η ταμπλέτες που εκτελούν το Android λειτουργικό σύστημα. Η Google αναφέρει ότι οι "εφαρμογές Android μπορούν να γραφτούν χρησιμοποιώντας γλώσσες Kotlin, Java και C ++" χρησιμοποιώντας το KIT ανάπτυξης λογισμικού Android (SDK), ενώ είναι δυνατή και η χρήση άλλων γλωσσών. Όλες οι γλώσσες εκτός JVM, όπως Go, JavaScript, C, C ++ ή συναρμολόγηση, χρειάζονται τη βοήθεια του κώδικα γλώσσας JVM, που μπορεί να παρέχεται από εργαλεία, πιθανόν με περιορισμένη υποστήριξη API. Ορισμένες γλώσσες / εργαλεία προγραμματισμού επιτρέπουν την υποστήριξη εφαρμογών μεταξύ πλατφορμών, δηλ. Και για το Android και για το iOS. Τα εργαλεία, τα αναπτυξιακά περιβάλλοντα και η υποστήριξη γλωσσών από τρίτους συνέχισαν να εξελίσσονται και να επεκτείνονται από τότε που κυκλοφόρησε το αρχικό SDK το 2008. Επιπλέον, με μεγάλες επιχειρηματικές οντότητες όπως η Walmart, η Amazon και η Bank of America, που ενδιαφέρονται να ασχοληθούν και να πουλήσουν μέσω κινητών τηλεφώνων, η ανάπτυξη εφαρμογών κινητής τηλεφωνίας γίνεται μάρτυρας μετασχηματισμού.

3.4.1.2 Android Software Development Kit

Μέσο του KIT ανάπτυξης λογισμικού Android SDK εμπεριέχετε ένα ολοκληρωμένο σύνολο εργαλείων αναβάθμισης μέσο αυτού υπάρχει ένα πρόγραμμα εντοπισμού σφαλμάτων, βιβλιοθηκών, ένας εξομοιωτής ακουστικών ρυθμιζόμενο στο QEMU, τεκμηρίωση σεμινάρια και κώδικα δείγματος. Οι σύγχρονες πλατφόρμες ανάπτυξης εμπεριέχουν υπολογιστές με Linux , Mac OS η κάποια νέα έκδοση τον windows 7 και μετά. Το SDK δεν διαδίδετε στο ίδιο το Android, από τον Μάρτιο του 2015 όμως η ανάπτυξη λογισμικού γίνετε εφικτή κάνοντας χρήση εξειδικευμένων εφαρμογών Android.

Έως τα τέλη του 2014 το πλήρες περιβάλλον ανάπτυξης (IDE) ήταν το Eclipse και αυτό που υποστήριξε επισήμως, κάνοντας χρήση το εργαλείο Android Development Tools (ADT) παρόλο που και το IntelliJ IDEA IDE υποστήριζε την ανάπτυξη Android εφαρμογών από την αρχή. Ενώ αντιθέτως το NetBeans IDE έκανε την ανάπτυξη του Android μέσω ενός Plugin. Ενώ από της αρχές του 2015 σαν επίσημο IDE ήταν το Android Studio το οποίο φτιάχτηκε από την Google με την χρήση του IntelliJ IDEA IDE. Παρ' όλα αυτά οι developers είναι ελεύθεροι να χρησιμοποιήσουν ότι IDE θελήσουν αλλά η Google έβγαλε ανακοίνωση σαν επίσημο εργαλείο το Android Studio.

Επιπρόσθετα ο κάθε προγραμματιστής έχει την δυνατότητα να κάνει χρήση οποιοδήποτε επεξεργαστή κειμένου για να επεξεργαστεί αρχεία κειμένου Java και XML και μετέπειτα να χρησιμοποιήσει εργαλεία γραμμής εντολών Java Development Kit και Apache Ant για την δημιουργία και την αποφυγή σφαλμάτων των εφαρμογών Android, όπως και την εξέταση και εξακρίβωση συνημμένων συσκευών Android. Οι αναβαθμίσεις από SDK του Android συμπορεύονται με την συνολική εξέλιξη της πλατφόρμας Android. Μέσο του SDK υποστηρίζονται ακόμα παλαιότερες εκδόσεις της πλατφόρμας Android , σε περίπτωση που κάποιος προγραμματιστής επιθυμεί να εντάξει τις εφαρμογές του σε κάποιες παλαιότερες. Υπάρχει δυνατότητα να μεταφορτωθούν τα εργαλεία ανάπτυξης και αφού έχετε υπόψιν την νεότερη έκδοση και πλατφόρμα, είναι δυνατόν οι πλατφόρμες και τα εργαλεία του παρελθόντος να μεταφορτωθούν για δοκιμές συμβατότητας.

Οι εφαρμογές Android φτιάχνονται σε μορφή .apk και η αποθήκευση τους γίνετε κάτω από τον φάκελο data/app του Android λειτουργικού (Ο φάκελος αυτός μπορεί να πάρει πρόσβαση μόνο ο εξουσιοδοτημένος χρήστης root για λόγους ασφαλείας). Στο πακέτο APK περιέχονται αρχεία dex (αρχεία κώδικα byte που βρίσκονται ως εκτελέσιμα αρχεία Dalvik), αρχεία πόρων κ.λπ.

3.4.1.3 Android SDK Platform Tools

Τα εργαλεία πλατφόρμας SDK Android είναι ένα υποσύνολο του πλήρους SDK που μπορεί να μεταφορτωθεί χωριστά, αποτελούμενο από εργαλεία γραμμής εντολών όπως adb και fastboot.

3.4.1.4 Android Debug Bridge (adb)

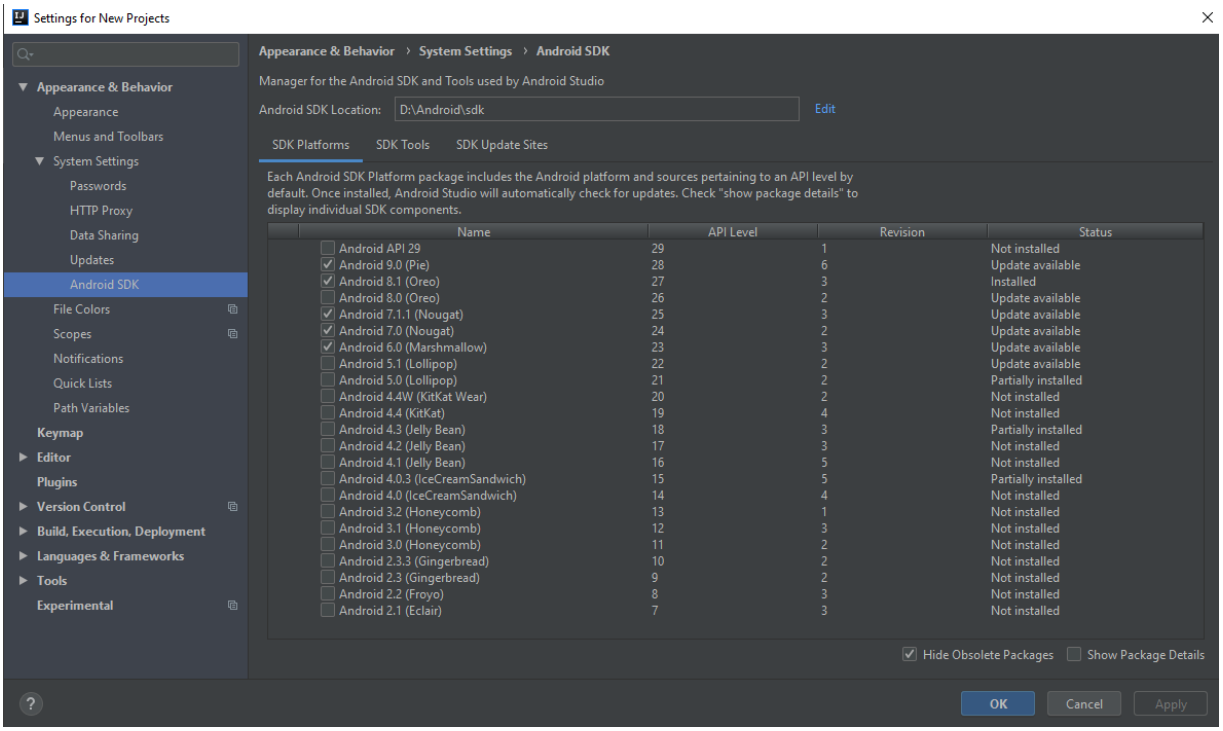
Το Bridge Debug Android (adb) είναι ένα εργαλείο για την εκτέλεση εντολών σε μια συνδεδεμένη συσκευή Android. Ο δαίμονας adbd εκτελείται στη συσκευή και ο πελάτης adb ξεκινά έναν διακομιστή παρασκηνίου να εκτελεί πολλαπλές εντολές που αποστέλλονται σε συσκευές. Εκτός από τη διασύνδεση της γραμμής εντολών, υπάρχουν πολυάριθμες γραφικές διεπαφές χρήστη για τον έλεγχο της adb.

3.4.2 Εγκατάσταση Android SDK

Αρχικά ανοίγουμε τον IntelliJ IDEA και κάτω δεξιά βλέπουμε ένα γαντζάκι να γράφει configure το πατάμε και πάμε στα settings όπως βλέπουμε και στην παρακάτω εικόνα.

Εικόνα 8 (IntelliJ Start Up)

Στην συνέχεια πηγαίνουμε Appearance & Behavior / System Settings / Android SDK και εκεί βάζουμε το path από το Android SDK που έχουμε κατεβάσει από εδώ https://developer.android.com/studio/ και ένας δεύτερος τρόπος είναι μέσα από τον ίδιο τον IDE να κατεβάσω τα πακέτα που θέλω χωρίς να ψάξω τα κατάλληλα πακέτα μονός μου από τον browser. Όπως βλέπω στην φωτογραφία πιο κάτω μπορώ να κατεβάσω και διαφορετικές version android ώστε να δοκιμάσω την εφαρμογή μου και σε πιο παλιά κινητά.



Εικόνα 9 (IntelliJ Android SDK Setup)

Κεφάλαιο 4: Απαραίτητες Γνώσεις Υλοποίησης

4.1 Frontend

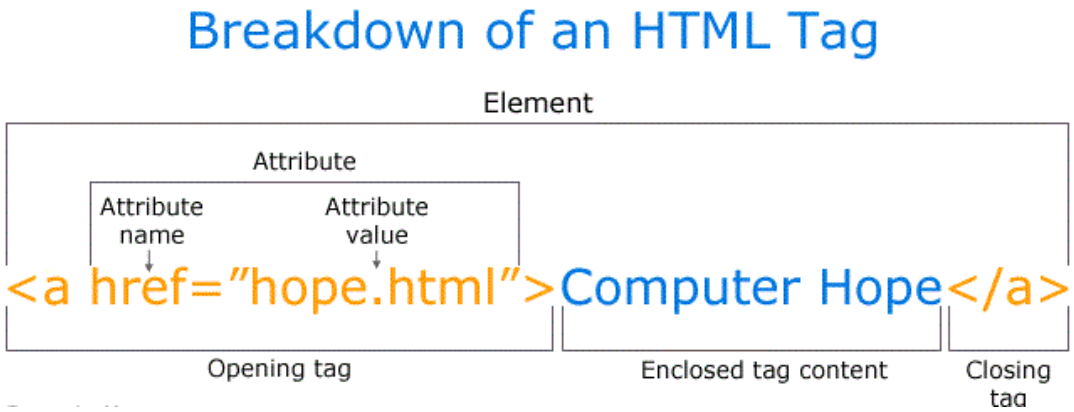
4.1.1 Γλώσσα Προγραμματισμού HTML

4.1.1.1 Γενικά

Αρχικά αναπτύχθηκε από τον Tim Berners-Lee το 1990, η HTML είναι σύντομη για το Hypertext Markup Language. Η HTML χρησιμοποιείται για τη δημιουργία ηλεκτρονικών εγγράφων (που ονομάζονται σελίδες) που εμφανίζονται στο World Wide Web. Κάθε σελίδα περιέχει μια σειρά συνδέσεων με άλλες σελίδες που ονομάζονται υπερσύνδεσμοι . Κάθε ιστοσελίδα που βλέπετε στο Διαδίκτυο γράφεται χρησιμοποιώντας μία έκδοση του κώδικα HTML ή άλλου.

Ο HTML κώδικας μας εξασφαλίζει την σωστή σύνταξη των εικόνων και του κείμενο ώστε ένας browser να μας τα εμφανίσει σωστά όπως πρέπει να τα δούμε. Χωρίς HTML, ένα πρόγραμμα περιήγησης δεν θα ξέρει πώς να εμφανίζει κείμενο ως στοιχεία ή να φορτώνει εικόνες ή άλλα στοιχεία. Η HTML παρέχει επίσης μια βασική δομή της σελίδας, πάνω στην οποία επικαλύπτονται φύλλα στυλ Cascading για να αλλάξει την εμφάνισή της. Κάποιος θα μπορούσε να σκεφτεί ότι η HTML είναι τα οστά (δομή) μιας ιστοσελίδας, και το CSS είναι το δέρμα του (εμφάνιση).

4.1.1.2 Πως φαίνεται ένα tag στην HTML;



Εικόνα 10 (Html Tags)

Όπως φαίνεται στο παράδειγμα ετικέτας HTML παραπάνω, δεν υπάρχουν πολλά στοιχεία. Σχεδόν όλες οι ετικέτες HTML έχουν μια ετικέτα ανοίγματος που περιέχει το όνομα με οποιαδήποτε χαρακτηριστικά, μια στενή ετικέτα που περιέχει μια πλάγια κάθετο και το όνομα της ετικέτας που κλείνει. Για ετικέτες που δεν έχουν ετικέτα κλεισίματος όπως <img>, είναι καλύτερο να αποσυνδέσετε την ετικέτα από μια εμπρός κάθετο.

Κάθε ετικέτα περιέχεται μέσα σε βραχίονες μικρότερες από και μεγαλύτερες από τις γωνίες και όλα μεταξύ της ετικέτας ανοίγματος και κλεισίματος εμφανίζονται ή επηρεάζονται από την ετικέτα. Στο παραπάνω παράδειγμα, η <a> ετικέτα δημιουργεί μια σύνδεση που ονομάζεται "Computer Hope" που δείχνει το αρχείο hope.html.

4.1.1.3 Πως φαίνεται η HTML;

Το παρακάτω είναι ένα παράδειγμα μιας βασικής ιστοσελίδας γραμμένης σε HTML, καθώς και μια περιγραφή κάθε τμήματος και της λειτουργίας της.

```
1  <!DOCType HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "https://www.w3.org/TR/html4/loose.dtd">
2  <html>
3      <head>
4          <title>Example page</title>
5          <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
6      </head>
7      <body>
8          <h1>This is a heading</h1>
9          <p>This is an example of a basic HTML page.</p>
10     </body>
11 </html>
```

Εικόνα 11 (Html Example)

Το παραπάνω κομμάτι περιέχει τα βασικά συστατικά σε μια βασική ιστοσελίδα. Η πρώτη γραμμή (DOCType) περιγράφει σε ποια έκδοση της HTML γράφτηκε η σελίδα έτσι ώστε ένα πρόγραμμα περιήγησης στο Internet να μπορεί να ερμηνεύσει το κείμενο που ακολουθεί. Στη συνέχεια, η ετικέτα ανοίγματος HTML επιτρέπει στο πρόγραμμα περιήγησης να γνωρίζει ότι διαβάζει κώδικα HTML. Η ετικέτα HTML ακολουθείται από την ενότητα κεφαλαίων που περιέχει πληροφορίες σχετικά με τη σελίδα, όπως τον τίτλο, τις ετικέτες meta και το σημείο εντοπισμού του αρχείου CSS. Το τμήμα σώματος είναι όλο το περιεχόμενο που είναι ορατό στο πρόγραμμα περιήγησης. Για παράδειγμα, όλο το κείμενο που βλέπετε εδώ περιέχεται μέσα στις

ετικέτες του σώματος. Τέλος, με το να κλείσω σωστά ετικέτες για κάθε στοιχείο έχω την σωστή σύνταξη.

4.1.1.4 Τι είναι HTML5;

Η HTML5 είναι η ενημέρωση που έγινε σε HTML από HTML4 (το XHTML ακολουθεί ένα διαφορετικό σχήμα αρίθμησης εκδόσεων). Χρησιμοποιεί τους ίδιους βασικούς κανόνες με το HTML4, αλλά προσθέτει κάποιες νέες ετικέτες και χαρακτηριστικά που επιτρέπουν την καλύτερη σημασιολογία και τα δυναμικά στοιχεία που ενεργοποιούνται χρησιμοποιώντας το JavaScript. Τα νέα στοιχεία περιλαμβάνουν ενότητα, <article>, <aide>, <bdi>, <details>, <dialog>, < figcaption>, <figure>, <footer>, <header>, <main> <mark>, <meter>, <nav>, <progress>, <rp>, <rt> , <ruby>, <section>, <summary>, <time> και <wbr>. Υπάρχουν επίσης νέοι τύποι εισόδου για φόρμες, οι οποίες περιλαμβάνουν tel, αναζήτηση, url, email, ημερομηνία, ημερομηνία, μήνα, εβδομάδα, ώρα, ημερομηνία, τοπικό, αριθμό, εμβέλεια και χρώμα.

Με την αυξανόμενη κίνηση για να διατηρηθεί χωριστή η δομή και το στυλ, έχουν αφαιρεθεί πολλά στοιχεία σχεδίασης μαζί με εκείνα που είχαν προβλήματα προσβασιμότητας ή είδαν πολύ λίγη χρήση. Αυτά τα ακόλουθα στοιχεία δεν θα πρέπει πλέον να χρησιμοποιούνται σε κώδικα HTML: , <frame>, <applet>, <font>, <acronym> και <tt>. Το HTML5 επίσης απλοποιεί τη δήλωση του τύπου doctype στην ετικέτα όπως θα δούμε παρακάτω.

```
<!doctype html>
```

4.1.1.5 Πως φαίνεται η HTML5;

Όπως φαίνεται παρακάτω, ο κώδικας HTML5 είναι πολύ παρόμοιος με το παλαιότερο παράδειγμα HTML4, αλλά είναι καθαρότερος και έχει μια αναθεωρημένη ετικέτα doctype.

```
1  <!doctype html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title>Example page</title>
6      </head>
7      <body>
8          <h1>This is a heading</h1>
9          <p>This is an example of a basic HTML page.</p>
10     </body>
11 </html>
```

Εικόνα 12 (Html 5 Example)

4.1.1.6 Πώς να δημιουργήσετε και να προβάλετε HTML;

Επειδή η HTML είναι μια γλώσσα σήμανσης, μπορεί να δημιουργηθεί και να προβληθεί σε οποιοδήποτε πρόγραμμα επεξεργασίας κειμένου, εφόσον αποθηκεύεται με επέκταση αρχείου .htm ή .html. Ωστόσο, οι περισσότεροι βρίσκουν ευκολότερο να σχεδιάζουν και να δημιουργούν ιστοσελίδες σε HTML χρησιμοποιώντας έναν επεξεργαστή HTML.

Μόλις δημιουργηθεί το αρχείο HTML, μπορεί να προβληθεί τοπικά ή να μεταφορτωθεί σε ένα διακομιστή ιστού που θα προβληθεί στο διαδίκτυο χρησιμοποιώντας ένα πρόγραμμα περιήγησης.

4.1.2 Γλώσσα Προγραμματισμού CSS

4.1.2.1 Γενικά

Το CSS σημαίνει Cascading Style Sheets με έμφαση στο "Style". Ενώ η HTML χρησιμοποιείται για τη δομή ενός εγγράφου ιστού (ορίζοντας θέματα όπως τίτλους και παραγράφους και επιτρέποντάς σας να ενσωματώσετε εικόνες, βίντεο και άλλα μέσα), το CSS έρχεται και καθορίζει ότι οι μορφές σελίδας στυλ σελίδας του εγγράφου, τα χρώματα και οι γραμματοσειρές καθορίζονται με CSS. Σκεφτείτε το HTML ως το θεμέλιο (κάθε σπίτι έχει ένα) και το CSS ως τις αισθητικές επιλογές (υπάρχει μεγάλη διαφορά ανάμεσα σε ένα βικτοριανό αρχοντικό και ένα σύγχρονο σπίτι μέσα του αιώνα).

4.1.2.2 Πώς λειτουργεί το CSS;

Το CSS φέρνει στυλ στις ιστοσελίδες σας, αλληλοεπιδρώντας με στοιχεία HTML. Τα στοιχεία είναι τα επιμέρους συστατικά HTML μιας ιστοσελίδας, για παράδειγμα μια παράγραφος η οποία στην HTML μπορεί να μοιάζει με αυτή:

```
<p>This is my paragraph!</p>
```

Εάν θέλετε να κάνετε αυτήν την παράγραφο να εμφανίζεται ροζ και τολμηρή στα άτομα που βλέπουν την ιστοσελίδα σας μέσω ενός προγράμματος περιήγησης ιστού, θα χρησιμοποιούσατε τον κώδικα CSS που μοιάζει με αυτόν τον τρόπο:

```
p {color:pink;  font-weight:bold;}
```

Σε αυτή την περίπτωση, ο όρος "p" (η παράγραφος) ονομάζεται "selector" είναι το μέρος του κώδικα CSS που καθορίζει ποιο στοιχείο HTML θα επηρεάσει το στυλ CSS. Στο CSS, ο selector γράφεται στα αριστερά του πρώτου σγουρού bracket. Η πληροφορία μεταξύ των bracket ονομάζεται δήλωση και περιέχει ιδιότητες και τιμές που εφαρμόζονται στον selector. Οι ιδιότητες είναι όπως το μέγεθος γραμματοσειράς, το χρώμα και τα περιθώρια, ενώ οι τιμές είναι οι ρυθμίσεις για αυτές τις ιδιότητες. Στο παραπάνω παράδειγμα, το "χρώμα" και η "γραμματοσειρά" είναι και οι δύο ιδιότητες, και το "ροζ" και "έντονο" είναι τιμές. Το πλήρες σετ παρενθέσεων

```
{color:pink;  font-weight:bold;}
```

είναι η δήλωση, και πάλι, ο "p" (που σημαίνει η παράγραφος HTML) είναι ο selector. Αυτές οι ίδιες βασικές αρχές μπορούν να εφαρμοστούν για να αλλάξουν μεγέθη γραμματοσειρών, χρώματα φόντου, εσοχές περιθωρίων και πολλά άλλα. Για παράδειγμα :

```
body {background-color:lightblue;}
```

4.1.2.3 Εξωτερικό, Εσωτερικό ή Inline CSS;

Μπορεί να αναρωτιέστε πώς αυτός ο κώδικας CSS εφαρμόζεται στην πραγματικότητα σε περιεχόμενο HTML. Όπως και με το HTML, το CSS γράφεται σε απλό, απλό κείμενο μέσω επεξεργαστή κειμένου ή επεξεργαστή κειμένου στον υπολογιστή σας και υπάρχουν τρεις βασικοί τρόποι για να προσθέσετε αυτόν τον κώδικα CSS στις σελίδες HTML. Ο κώδικας CSS (ή τα φύλλα στυλ) μπορεί να είναι εξωτερικός, εσωτερικός ή ενσωματωμένος. Τα εξωτερικά φύλλα στυλ αποθηκεύονται ως αρχεία .css και είναι υπεύθυνα για την εμφάνιση ολόκληρου του ιστότοπου μέσω ενός αρχείου (αντί να προσθέσουν μεμονωμένες εμφανίσεις κώδικα CSS σε κάθε στοιχείο HTML που θέλετε να προσαρμόσετε). Για να χρησιμοποιήσετε ένα εξωτερικό φύλλο στυλ, τα αρχεία .html πρέπει να περιλαμβάνουν μια ενότητα κεφαλίδας που συνδέει με το εξωτερικό φύλλο στυλ και μοιάζει με κάτι τέτοιο:

```
<head>
    <link rel="stylesheet"  type="text/css"  href="style.css">
</head>
```

Εικόνα 13 (Import CSS)

Αυτό θα συνδέσει το αρχείο .html με το εξωτερικό φύλλο στυλ (σε αυτήν την περίπτωση, το style.css) και όλες οι οδηγίες του CSS σε αυτό το αρχείο θα εφαρμοστούν στη συνδεδεμένη σελίδα σαν .html.

Τα εσωτερικά φύλλα στυλ είναι οδηγίες CSS γραμμένες απευθείας στην κεφαλίδα μιας συγκεκριμένης σελίδας .html. (Αυτό είναι ιδιαίτερα χρήσιμο αν έχετε μια μόνο σελίδα σε μια τοποθεσία που έχει μια μοναδική εμφάνιση.) Ένα εσωτερικό φύλλο στυλ μοιάζει με αυτό:

```
<style>
    body {
        background-color:red;
    }
    p {
        font-size:20px;
        color:blue;
    }
</style>
```

Εικόνα 14 (Import CSS in top)

ένα χρώμα φόντου red και οι παράγραφοι με 20 font, μπλε γραμματοσειρά θα εφαρμοστούν τώρα σε αυτήν την ενιαία σελίδα .html.

Τέλος, τα ενσωματωμένα στυλ είναι αποσπάσματα CSS γραμμένα απευθείας σε κώδικα HTML και ισχύουν μόνο για μια ενιαία παράμετρο κωδικοποίησης. Για παράδειγμα:

```
<body>
    <h1 style="font-size:40px; color:violet;">Check out this headline!</h1>
</body>
```

Εικόνα 15 (Add tags CSS Inline Code)

Σε γενικές γραμμές, τα εξωτερικά φύλλα στυλ είναι η πιο αποτελεσματική μέθοδος για την εφαρμογή του CSS σε έναν ιστότοπο (είναι ευκολότερο να παρακολουθείτε και να εφαρμόσετε το στυλ ενός ιστότοπου από ένα αποκλειστικό αρχείο CSS), ενώ τα εσωτερικά φύλλα στυλ και το inline στυλ μπορούν να χρησιμοποιηθούν σε μια περίπτωση σε περίπτωση που πρέπει να γίνουν μεμονωμένες αλλαγές στυλ.

Έτσι, αν το HTML είναι το θεμέλιο, τα πλαίσια, οι τοίχοι και οι δοκοί που υποστηρίζουν τον ιστότοπό σας, σκεφτείτε το CSS το χρώμα βαφής, τα στυλ παραθύρων και τον εξωραϊσμό που έρχεται αργότερα. Δεν μπορείτε να πάρετε οπουδήποτε χωρίς να βάλετε πρώτα αυτά τα θεμέλια, αλλά μόλις το κάνετε θα θέλετε να ακολουθήσετε με κάποιο στυλ και το CSS είναι το εισιτήριο για την απελευθέρωση του εσωτερικού διακοσμητή σας.

4.1.3 Γλώσσα Προγραμματισμού JavaScript/jQuery

4.1.3.1 Γενικά

Η JavaScript είναι μια γλώσσα προγραμματισμού σεναρίων που την χρησιμοποιούμε για τη δημιουργία και τον έλεγχο δυναμικού περιεχομένου ιστότοπου αλλά αυτό μπορεί να μην έχει πολύ νόημα αν είστε νέοι στην τεχνολογία. Επομένως, ας αντικαταστήσουμε το "δυναμικό περιεχόμενο ιστότοπου" με "πράγματα που μετακινούνται, ανανεώνονται ή αλλιώς αλλάζουν στην οθόνη σας χωρίς να χρειάζεται να επαναλάβετε τη φόρτωση μιας ιστοσελίδας με μη αυτόματο τρόπο".

Σκεφτείτε χαρακτηριστικά όπως κινούμενα γραφικά, slideshows φωτογραφιών, προτάσεις αυτόματης συμπλήρωσης κειμένου και δια δραστικές φόρμες. Ή καλύτερα, σκεφτείτε τις λειτουργίες ιστού που θεωρείτε εντελώς δεδομένες, όπως όταν ενημερώνετε το χρονικό σας γράμμα Facebook στην οθόνη σας ή η Google προτείνει όρους αναζήτησης με βάση λίγα γράμματα που έχετε πληκτρολογήσει στη γραμμή αναζήτησής σας. Και στις δύο περιπτώσεις, αυτό είναι JavaScript στη πραγματικότητα.

4.1.3.2 Ποια είναι τα βασικά στοιχεία του JavaScript;

Αν ελπίζετε να ξεπεράσετε μια καριέρα στην τεχνολογία, η ερώτησή σας μπορεί να ακούγεται περισσότερο: "τι είναι το JavaScript και το όντως χρειάζομαι;"

Ανέφερα παραπάνω ότι η JavaScript είναι μια "γλώσσα δέσμης ενεργειών". Οι γλώσσες συγγραφής είναι γλώσσες κωδικοποίησης που χρησιμοποιούνται για την αυτοματοποίηση διαδικασιών που διαφορετικά οι χρήστες θα έπρεπε να εκτελούν μόνοι τους, βήμα προς βήμα. Χωρίς scripting, οποιεσδήποτε αλλαγές στις ιστοσελίδες που επισκέπτεστε θα απαιτούσαν είτε τη μη αυτόματη επαναφόρτωση της σελίδας είτε την πλοήγηση μιας σειράς στατικών μενού για να φτάσετε στο περιεχόμενο που βρίσκεστε.

Μια γλώσσα scripting όπως το JavaScript (JS, για όσους γνωρίζουν) κάνει την βαριά ανύψωση λέγοντας προγράμματα ηλεκτρονικών υπολογιστών, όπως ιστοσελίδες ή εφαρμογές ιστού, να κάνουν κάτι. Στην περίπτωση του JavaScript, αυτό σημαίνει όλα αυτά τα δυναμικά χαρακτηριστικά που περιγράφονται νωρίτερα όπως εικόνες για να ζωντανέψουν, φωτογραφίες για να κυκλώσουν σε μια παρουσίαση διαφανειών ή για αυτόματες συμπλήρωση προτάσεων για

να απαντήσουν στις προτροπές. Είναι το "script" στη JavaScript που κάνει αυτά τα πράγματα να συμβαίνουν φαινομενικά μόνα τους.

Εν τω μεταξύ, επειδή η JavaScript είναι ένα τέτοιο αναπόσπαστο κομμάτι της λειτουργικότητας του διαδικτύου, όλα τα μεγάλα προγράμματα περιήγησης διατίθενται με ενσωματωμένους κινητήρες που μπορούν να αποδώσουν JavaScript. Αυτό σημαίνει ότι οι εντολές JS μπορούν να δακτυλογραφούν απευθείας σε ένα έγγραφο HTML και οι browser ιστού θα μπορέσουν να τις καταλάβουν. Με άλλα λόγια, η χρήση του JavaScript δεν απαιτεί τη λήψη πρόσθετων προγραμμάτων ή μεταγλωττιστών.

4.1.3.3 Vanilla JavaScript

Παρακάτω θα δείτε πως φαίνεται το JavaScript, εξετάστε το ακόλουθο βασικό παράδειγμα του κώδικα JavaScript. Εάν θέλετε οι χρήστες να λαμβάνουν μήνυμα επιβεβαίωσης "ευχαριστίες για την εγγραφή" μετά την εγγραφή σας για μια προσφορά ή υπηρεσία σε έναν ιστότοπο, θα κωδικοποιούσατε απευθείας σε μια σελίδα HTML (μεταξύ ετικετών <script>) όπως αυτή:

```
<script>
    window.onload = initAll;
    function initAll() {
        document.getElementById("submit").onclick = submitMessage;
    }
    function submitMessage() {
        var greeting = document.getElementById("name").getAttribute("value");
        document.getElementById("headline").innerHTML = "Thank you for joining our email list," + greeting;
        return false;
    }
<script/>
```

Εικόνα 16 (Vanilla Javascript Greetings)

Γράφοντας αυτό το είδος κώδικα και εισάγοντας το JavaScript σε έγγραφα HTML ονομάζεται "Vanilla JavaScript." Μόνο η Vanilla JS μπορεί να χρησιμοποιηθεί για τη δημιουργία έργων JavaScript, αλλά καθώς εξοικειώνεστε περισσότερο με τη γλώσσα JavaScript, υπάρχουν διάφορα εργαλεία που μπορείτε να εφαρμόσετε για να κάνετε JS ευκολότερη και πιο αποδοτική στη χρήση.

4.1.3.4 JavaScript Tools

jQuery

Όταν εργάζεστε με το JavaScript, θα παρατηρήσετε τις λειτουργίες JS και τις λειτουργίες που εμφανίζονται τακτικά σε πολλαπλούς ιστότοπους ή εφαρμογές ιστού-πράγματα όπως κινούμενα εφέ μενού και εξαλείψεις, φόρμες φόρτωσης αρχείων και γκαλερί εικόνων. Ενώ θα μπορούσατε να κωδικοποιήσετε κάθε ένα από αυτά τα πράγματα από το έδαφος κάθε φορά που χρειάζεστε ένα, η κωδικοποιητική ζωή σας θα αισθάνεται πολύ πιο εύκολη εάν χρησιμοποιήσετε βιβλιοθήκες κωδικοποίησης όπως το jQuery.

Η βιβλιοθήκη jQuery αποτελείται από λειτουργίες κωδικοποίησης JavaScript που μπορούν να εκτελεστούν μέσω εντολών jQuery μιας γραμμής. Για παράδειγμα, το παραπάνω παράδειγμα κώδικα JavaScript μοιάζει με το παραπάνω που ανέφερα, αντ' αυτού εκτελείται χρησιμοποιώντας τον κώδικα jQuery:

```
<script>
    $("#submit").click(function () {
        var greeting = $("#name").val();
        $("#headline").html("Thank you for joining our email list, " + greeting);
        return false;
    });
<script/>
```

Εικόνα 17(jQuery Greetings)

Όπως μπορείτε να δείτε, η προσέγγιση προγραμματισμού jQuery είναι πολύ πιο συνοπτική και μπορεί να χρησιμοποιηθεί ξανά οποιαδήποτε στιγμή θέλετε να εκτελέσετε την ίδια λειτουργία JavaScript ενώ κωδικοποιείτε έναν ιστότοπο ή μια εφαρμογή ιστού.

Εκτός από παραδείγματα όπως αυτά που αναφέρονται παραπάνω (τα οποία θεωρούνται αποσπάσματα jQuery, αποσπάσματα κώδικα που έχουν εισαχθεί απευθείας από τη βιβλιοθήκη jQuery για την εκτέλεση αφιερωμένων λειτουργιών), ο κώδικας jQuery μπορεί να συνδυαστεί για να δημιουργήσει πιο περίπλοκα πρόσθετα. Τα jQuery plugins μπορούν να βρεθούν απευθείας από τον αποθετήριο jQuery UI (User Interface), όπου ο πηγαίος κώδικας είναι διαθέσιμος για αντιγραφή και επικόλληση.

JavaScript Frameworks

Ενώ οι βιβλιοθήκες JS όπως το jQuery λειτουργούν ως ψηφιακά μαχαίρια ελβετικού στρατού για μεμονωμένες ανάγκες κωδικοποίησης, οι προγραμματιστές του frontend μπορούν να κάνουν τα πράγματα ένα βήμα πιο μακριά, χρησιμοποιώντας εργαλεία που ονομάζονται frameworks JavaScript.

Με την επέκταση του παρελθόντος της λειτουργικότητας του jQuery, τα πλαίσια JS παρέχουν προγραμματιστές JavaScript με πρότυπα για ιστοσελίδες ή εφαρμογές ιστού που θα χτιστούν γύρω. Τα πλαίσια JS δημιουργούν χώρους σε ένα πρότυπο όπου συνιστάται ο κωδικός JS, καθώς και ένας προειδοποιητικός κώδικας (όπως το jQuery) που μπορεί να συνδεθεί σε αυτούς τους χώρους.

Αν και υπάρχουν αρκετά πλαίσια που μπορούν να θεωρηθούν ως πρότυπα για τη βιομηχανία (π.χ. Vue, React, Angular, Ember, Backbone και Meteor), το καλύτερο στοίχημά σας όταν επιλέγετε κάποιον που θα μάθει είναι να εξετάσει τους πιθανούς εργοδότες ή τους πελάτες και να προσδιορίσει ποιο πλαίσιο JS και αν υπάρχει. Και να έχετε κατά νου σας, μόλις μάθετε ένα framework μετά είναι σχετικά εύκολο να μάθετε και όλα τα υπόλοιπα.

4.1.4 Η χρήση του Framework Bootstrap 4

4.1.4.1 Γενικά

Αν έχετε κάποιο ενδιαφέρον για την ανάπτυξη ιστού, πιθανότατα έχετε ακούσει για το Framework Bootstrap. Σύμφωνα με τον επίσημο ιστότοπο, το Bootstrap είναι το πιο δημοφιλές Framework HTML, CSS και JS για την ανάπτυξη πρωτοποριακών και όμορφων έργων στο διαδίκτυο.

4.1.4.2 Τώρα πώς θα το χρησιμοποιήσω;

Ο οδηγός εγκατάστασης είναι πράγματι μια σειρά από χρήσιμες πληροφορίες, συνδέσεις με CDNs, εξηγήσεις σχετικά με τον τρόπο εγκατάστασης με το Bower, Npm και Composer, πληροφορίες σχετικά με την ενσωμάτωση με το Autoprefixer και LESS, μια δέσμη προτύπων, άδειες και μεταφράσεις αλλά δεν είναι getting start documentation για ξεκινήσετε.

Όταν ανακάλυψα το Bootstrap πριν από λίγα χρόνια, ο ευαίσθητος σχεδιασμός εξακολουθούσε να κερδίζει δημοτικότητα, και όχι απαραίτητα ο αναμενόμενος κανόνας. Αφού κάποτε κάνανε ιστοσελίδες από το μηδέν. Φαντάζομαι ότι ακόμα και για αρχάριους οι οποίοι τώρα αναμένεται να μάθουν τις φιλοσοφίες σχεδίασης και τις βιβλιοθήκες Bootstrap και JavaScript, σε σχέση με το να ξεκινήσουν HTML, CSS και JS.

Αυτός ο οδηγός θεωρείται ως μια πρώτη ματιά στο Bootstrap για αρχάριους, οπότε δεν πρόκειται να εισέλθουν στην ενσωμάτωση LESS και Sass, οι οποίες είναι πιο ενδιάμεσες / προηγμένες έννοιες. Ενώ είναι γραμμένο για την τρέχουσα, σταθερή έκδοση Bootstrap 3, οι έννοιες θα παραμείνουν οι ίδιες για μελλοντικές εκδόσεις.

4.1.4.3 Τι είναι η Bootstrap;

Η Bootstrap βασίζεται σε τρία βασικά αρχεία:

- bootstrap.css - το CSS πλαίσιο
- bootstrap.js - το JavaScript/jQuery πλαίσιο
- glyphicons – και τα fonts libraries

Επιπλέον, η Bootstrap απαιτεί τη λειτουργία του jQuery. Το jQuery είναι μια εξαιρετικά δημοφιλής και ευρέως χρησιμοποιούμενη βιβλιοθήκη JavaScript, που απλοποιεί και προσθέτει πολλαπλή συμβατότητα browser με JavaScript.

Όλα τα άλλα που μπορούν να συμβούν σε όλη τη διάρκεια της μελέτης της τεκμηρίωσης του Bootstrap - Grunt, Gulp, Sass, LESS, Bower, npm κ.λπ. δεν είναι απαραίτητα για να ξεκινήσετε με το Bootstrap. Αυτοί είναι οι δρομείς, οι προ επεξεργαστές, τα βοηθήματα εγκατάστασης και οι διαχειριστές πακέτων, οπότε μην αποθαρρύνεστε αν δεν ξέρετε πώς να χρησιμοποιήσετε κάποιο από αυτά ακόμα.

4.1.4.4 Γιατί είναι σημαντικό το Framework; Χρειάζεται να χρησιμοποιήσω ένα;

Δεν χρειάζεται απολύτως να χρησιμοποιήσετε ένα Framework. Ωστόσο, τα Frameworks είναι πολύ δημοφιλή και έχουν πολλά οφέλη, οπότε είναι σημαντικό να μάθουμε πώς να συνεργαστούμε μαζί τους.

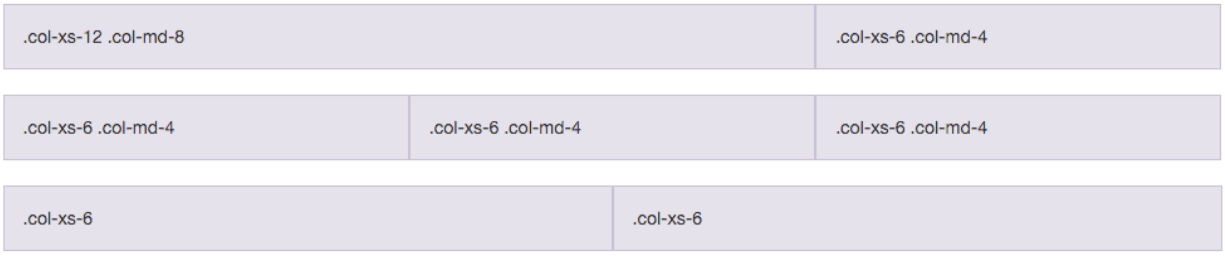
Μερικοί από τους τρόπους με τους οποίους τα πλαίσια μπορούν να σας βοηθήσουν:

- Αποτρέψτε την επανάληψη μεταξύ των έργων
- Χρησιμοποιήστε το σχεδιασμό που ανταποκρίνεται στις ανάγκες σας για να επιτρέψετε στον ιστότοπό σας να προσαρμόζεται σε διάφορα μεγέθη οθόνης, κινητά, υπολογιστές γραφείου κ.α.

- Προσθέστε συνέπεια στη σχεδίαση και κωδικοποίηση μεταξύ έργων και μεταξύ προγραμματιστών.
- Δοκιμάστε γρήγορα και εύκολα πρωτότυπα νέα σχέδια.
- Εξασφαλίστε συμβατότητα μεταξύ των browser.

Κάθε πρόγραμμα περιήγησης πρέπει να ανταποκρίνεται και να τρέχει σωστά το Web Project σας, αλλά σίγουρα θα υπάρχουν κάποιες αδυναμίες σε πιο παλαιότερα προγράμματα περιήγησης. Η Bootstrap έχει μια τεράστια κοινότητα ανοιχτού κώδικα που λειτουργεί για την κάλυψη αυτού. Επιπλέον, όταν πολλοί προγραμματιστές γνωρίζουν όλοι το ίδιο σύστημα, μπορούν να εργαστούν με μεγαλύτερη αρμονία και επίσης διευκολύνουν τους νεοεισερχόμενους σε ένα έργο που θα μπορούσαν να επιταχύνουν πολύ πιο γρήγορα.

Το grid είναι ίσως μια από τις πιο ουσιαστικές πτυχές του framework. Είναι η βάση πάνω στην οποία δημιουργείται ολόκληρη η διάταξη. Πέρα από αυτό, το κεντρικό CSS του Bootstrap θα προσθέσει επίσης χρήσιμο στυλ σε φόρμες, πίνακες, κουμπιά, λίστες και εικόνες καθώς και πλήρως navigation bars, ενώ ο πυρήνας JavaScript θα προσθέσει χρήσιμο κώδικα για τη δημιουργία modals, carousels, alerts, popups, dropdowns και accordions.



Εικόνα 18 (Bootstrap Grid System)

4.1.4.5 Ας φτιάξουμε ένα βασικό template με την χρήση τις Bootstrap.

Το Bootstrap έρχεται με μερικά πολύ απλά παραδείγματα για να ξεκινήσετε, αλλά είναι εξίσου εύκολο να ξεκινήσετε από το μηδέν και αυτό θα κάνουμε. Πρώτα απ' όλα, θα χρησιμοποιήσω μόνο την Bootstrap για ξεκινήσω, τότε θα προσθέσουμε το δικό μας προσαρμοσμένο στυλ στην κορυφή για να κάνουμε κάτι διασκεδαστικό και μοντέρνο.

Το πρώτο βήμα είναι να κατεβάσετε την Bootstrap. Το αρχείο zip θα συνοδεύεται από CSS, Fonts και JS. Αποσυμπιέστε το και αποθηκεύστε τα αρχεία σε κάποιο κατάλογο. Το Bootstrap δεν έρχεται με οποιαδήποτε HTML, αλλά έχει ένα "Hello, World!" για να ξεκινήσετε από την τεκμηρίωση, οπότε θα το χρησιμοποιήσουμε ως index.html.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Bootstrap 101 Template</title>
    <link href="css/bootstrap.min.css" rel="stylesheet" />
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <h1>Hello, world!</h1>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

Εικόνα 19 (Bootstrap Example)

Αρκετά εύκολο να ξεκινήσετε. Έχουμε τις βασικές ετικέτες μας doctype, html, head και body. Η ετικέτα meta name = "viewport" είναι ιδιαίτερα σημαντική για τον σχεδιασμό που ανταποκρίνεται, εξασφαλίζει ότι ο ιστότοπός σας έχει αναλογία 1: 1 με το παράθυρο προβολής (μέγεθος οθόνης).

Πέρα από αυτό, προσθέτουμε απλά το CSS Core Bootstrap στο head:

```html
<head>
    <link href="css/bootstrap.min.css" rel="stylesheet" />
</head>
```

Εικόνα 20 (Import Bootstrap CSS)

jQuery από το CDN τις Google πριν από την ετικέτα κλεισίματος body:

```html
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
</body>
```

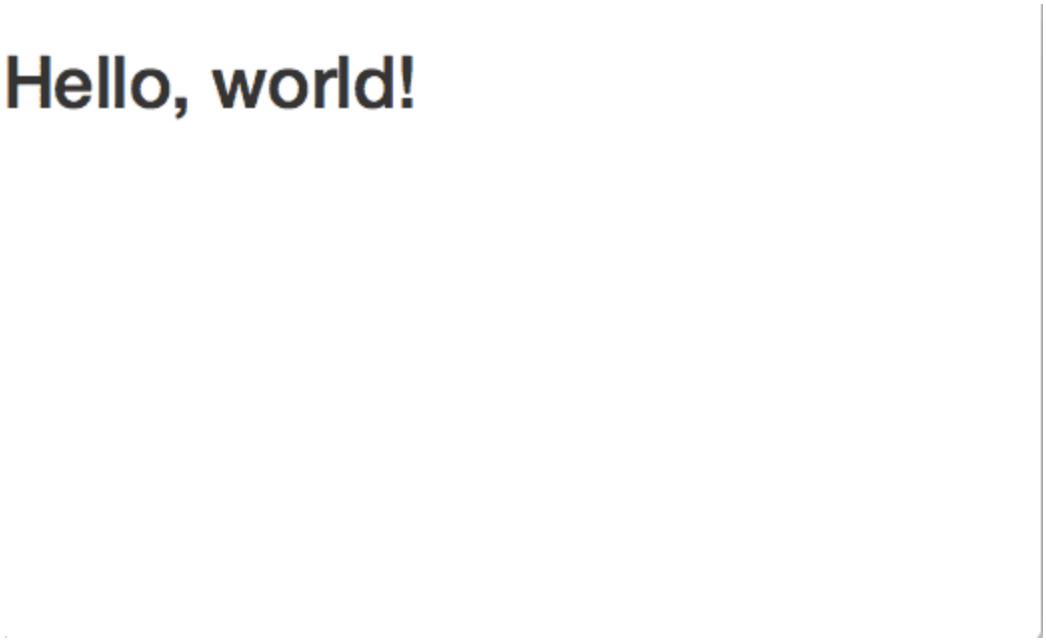
Εικόνα 21 (Import jQuery for Bootstrap)

και το JavaScript core του Bootstrap:

```html
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
</body>
```

Εικόνα 22 (Import Bootstrap JS)

Αυτά είναι όσα θα χρειαστούμε για να δουλέψομε με την βιβλιοθήκη Bootstrap. Ας δούμε τον υπέροχο νέο μας ιστότοπο.



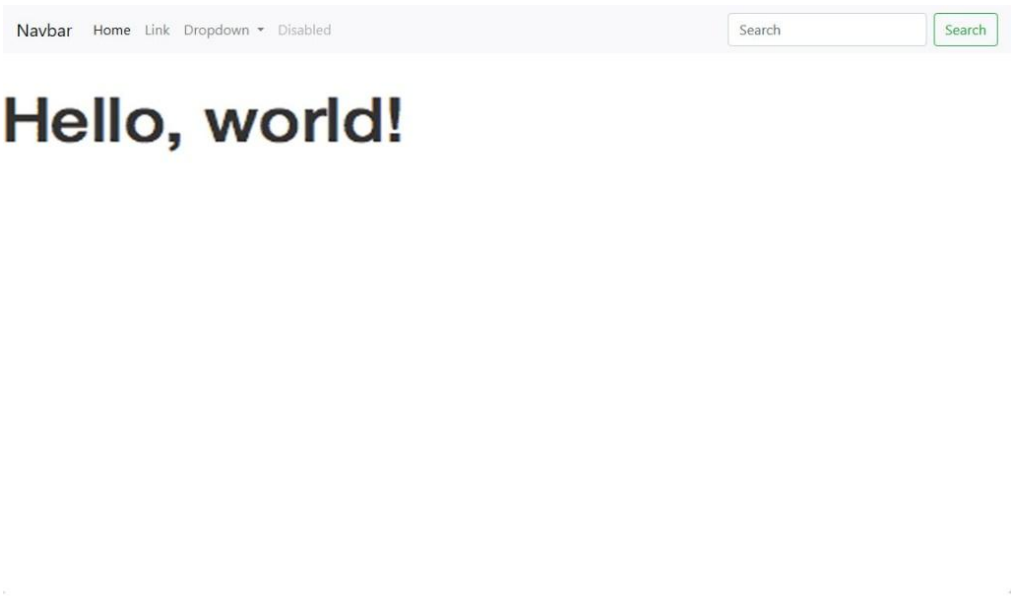
Εικόνα 23 (Bootstrap Hello World)

4.1.4.6 Navigation Bar

Παρόλο που δεν έχουμε τίποτα, σε ελάχιστο χρόνο μπορούμε να κάνουμε αντιγραφή και επικόλληση από τα έγγραφα και να έχουμε έναν ωραίο και λειτουργικό ιστότοπο. Πρώτα απ' όλα, προσθέτουμε το Navbar Bootstrap. Έκανα μια απλοποιημένη έκδοση του παραδείγματός μας για το navbar. Τοποθετήστε αυτόν τον κώδικα ακριβώς κάτω από την ετικέτα <body> που ανοίγει.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Dropdown
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>
```

Εικόνα 24 (Bootstrap Navigation Bar Code)



Εικόνα 25 (Bootstrap Navigation Bar Example)

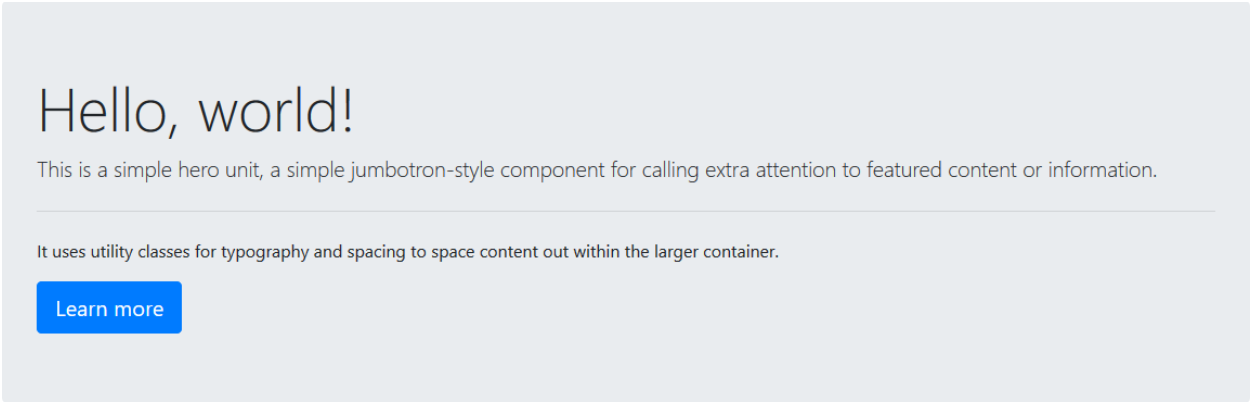
4.1.4.7 Jumbotron Header

Αποφάσισα ότι θέλω ένα από αυτά τα μεγάλα space-wasting, attention-grabbing headers, που ονομάζεται jumbotron σε όρους Bootstrap. Δεν έχει πολλά να δει εδώ, απλά ένα Jumbotron με <a href>.

```
<div class="jumbotron">
    <h1 class="display-4">Hello, world!</h1>
    <p class="lead">This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.</p>
    <hr class="my-4">
    <p>It uses utility classes for typography and spacing to space content out within the larger container.</p>
    <a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>
</div>
```

Εικόνα 26 (Bootstrap Jumbotron Code)

Υπάρχει ένας επιπλέον χώρος που δεν θέλουμε, αλλά θέλω να δω πόσο μακριά μπορεί να μας πάρει το Bootstrap χωρίς να έχει κάνω override κάποιο style. Όπως μπορείτε να δείτε, έχουμε ήδη μια αρκετά συμπαθητική, προσαρμόσιμη διάταξη χωρίς να έχουμε γράψει μία γραμμή CSS.



Εικόνα 27 (Bootstrap Jumbotron Example)

4.1.4.8 Grid System

Το τελευταίο πράγμα που θα κάνω είναι να προσθέσω σε κάποιο κύριο περιεχόμενο, το οποίο θα έχει τη μορφή ενός grid. Το grid έχει rows.

```
<div class="row"></div>
```

Εικόνα 28 (Bootstrap Row)

Το οποίο περιέχει στήλες.

```
<div class="row">
  <div class="col-md-6"></div>
  <div class="col-md-6"></div>
</div>
```

Εικόνα 29 (Bootstrap Cols)

Η Bootstrap λειτουργεί σε ένα σύστημα με 12 columns, αρκεί να προσθέτετε έως και 12. Το παραπάνω παράδειγμα περιέχει δύο columns πλάτους 50% (6/12), οι οποίες θα στοιβάζονται στο κινητό και θα καθίστανται 100% πλάτος.

4.1.4.9 Icons

Έχω προσθέσει επίσης σε μερικά glyphicons για διακόσμηση. Τα Glyphicons είναι το ενσωματωμένο εικονίδιο που συνοδεύει την Bootstrap. Εάν δεν φορτώσετε τις γραμματοσειρές σας ή εάν τις μετακινήσετε σε διαφορετικό κατάλογο, αυτά τα εικονίδια δεν θα λειτουργήσουν.

```
<span class="glyphicon glyphicon-floppy-disk" aria-hidden="true"></span>
```

Εικόνα 30 (Bootstrap Class Icons)

Η χρήση ενός glyphicon θα είναι πάντα ο ίδιος κώδικας και μόνο η κλάση glyphicon-floppy-disk θα αλλάξει σε κάποια άλλη κλάση. Μπορείτε να βρείτε εδώ μια λίστα από icons της βιβλιοθήκης Bootstrap.

4.2 Backend

4.2.1 ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PHP

4.2.1.1 Τι είναι η PHP;

Η PHP θεωρείτε ως μια server side scripting γλώσσα, που την χρησιμοποιούμε για την ανάπτυξη στατικών ιστότοπων ή δυναμικών ιστότοπων ή εφαρμογών Web. Η PHP αντιπροσωπεύει το Hypertext Pre-processor, το οποίο νωρίτερα ανήκε στην προσωπική αρχική σελίδα.

Τα scripts PHP μπορούν να ερμηνευτούν μόνο σε ένα διακομιστή που έχει εγκαταστήσει PHP.

Οι υπολογιστές-πελάτες που έχουν πρόσβαση στα σενάρια PHP απαιτούν μόνο ένα πρόγραμμα περιήγησης στο Web.

Ένα αρχείο PHP περιέχει ετικέτες PHP και τελειώνει με την επέκταση ".php".

Παρακάτω θα αναφέρω ονομαστικά τι θα μάθουμε για την PHP:

- Τι είναι η scripting γλώσσα;
- Γλώσσα προγραμματισμού vs scripting γλώσσα.
- Τι σημαίνει PHP;
- Σύνταξη PHP.
- Γιατί να χρησιμοποιήσετε την PHP;
- Που χρησιμοποιείται η PHP για το Market Share.
- PHP files extensions

4.2.1.2 Τι είναι η scripting γλώσσα;

Ένα σενάριο είναι ένα σύνολο οδηγιών προγραμματισμού που ερμηνεύεται κατά το χρόνο εκτέλεσης.

Μια γλώσσα scripting είναι μια γλώσσα που ερμηνεύει σενάρια κατά το χρόνο εκτέλεσης. Τα scripts είναι συνήθως ενσωματωμένα σε άλλα περιβάλλοντα λογισμικού.

Ο σκοπός των σεναρίων είναι συνήθως η βελτίωση της απόδοσης ή η εκτέλεση εργασιών ρουτίνας για μια εφαρμογή.

Τα server side scripts ερμηνεύονται στο διακομιστή, ενώ τα client side scripts ερμηνεύονται από την εφαρμογή-πελάτη.

Η PHP είναι ένα server side script που ερμηνεύεται στο διακομιστή ενώ το JavaScript είναι ένα παράδειγμα client side script που ερμηνεύεται από το πρόγραμμα περιήγησης-πελάτη.

Τόσο η PHP όσο και η JavaScript μπορούν να ενσωματωθούν σε σελίδες HTML.

4.2.1.3 Γλώσσα προγραμματισμού vs scripting γλώσσα.

Programming language	Scripting language
Has all the features needed to develop complete applications.	Mostly used for routine tasks
The code has to be compiled before it can be executed	The code is usually executed without compiling
Does not need to be embedded into other languages	Is usually embedded into other software environments.

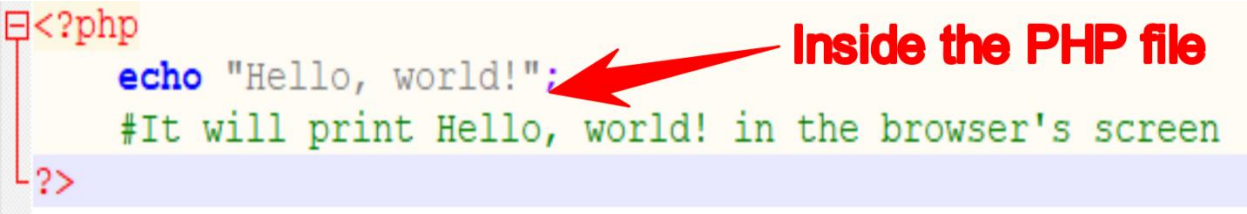
Πίνακας 3 (Programming vs Scripting)

4.2.1.4 Τι σημαίνει PHP;

Η PHP σημαίνει Personal Home Page, αλλά τώρα σημαίνει το αναδρομικό backronym PHP: Hypertext Preprocessor.

Μπορούμε να χρησιμοποιήσουμε τον κώδικας PHP μαζί με κώδικα HTML ή να τον χρησιμοποιήσουμε με διαφορά συστήματα προτύπων ιστού, Content Management Systems ιστού και Web Frameworks.

4.2.1.5 Σύνταξη PHP

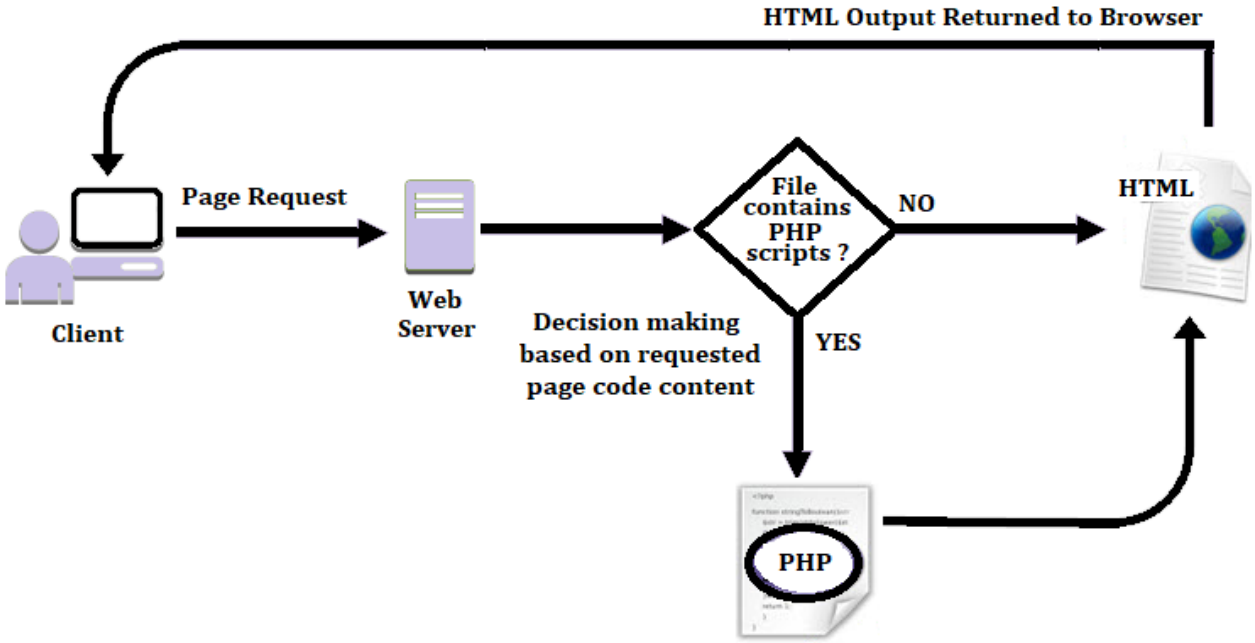


Εικόνα 31 (PHP syndax code)

Ένα αρχείο PHP μπορεί επίσης να περιέχει ετικέτες όπως HTML και script side clients όπως JavaScript.

- Η HTML αποτελεί ένα πρόσθετο πλεονέκτημα κατά την εκμάθηση γλώσσας PHP. Μπορείτε ακόμη να μάθετε PHP χωρίς να γνωρίζετε HTML, αλλά συνιστάται να γνωρίζετε τουλάχιστον τα βασικά του HTML.
- Συστήματα διαχείρισης βάσεων δεδομένων DBMS.
- Για πιο προηγμένα θέματα, όπως διαλογικές εφαρμογές και υπηρεσίες ιστού, θα χρειαστείτε JavaScript και XML.

Το διάγραμμα ροής που φαίνεται παρακάτω απεικονίζει τη βασική αρχιτεκτονική μιας εφαρμογής web PHP και τον τρόπο με τον οποίο ο διακομιστής χειρίζεται τις αιτήσεις.



Εικόνα 32 (Html - Php work)

4.2.1.6 Γιατί να χρησιμοποιήσετε την PHP;

Έχετε προφανώς πολλές γλώσσες προγραμματισμού εκεί έξω. Ίσως να αναρωτιέστε γιατί θα θέλαμε να χρησιμοποιήσουμε την PHP ως το δηλητήριό μας για τον προγραμματισμό ιστοσελίδων. Παρακάτω είναι μερικοί από τους επιτακτικούς λόγους.

- Η PHP είναι open source και free.
- Σύντομη καμπύλη μάθησης σε σύγκριση με άλλες γλώσσες όπως JSP, ASP κλπ.
- Μεγάλο community.
- Οι περισσότεροι web hosting servers υποστηρίζουν την PHP από προεπιλογή σε αντίθεση με άλλες γλώσσες όπως ASP που χρειάζονται IIS server τις Microsoft. Αυτό κάνει την PHP μια οικονομικά αποδοτική επιλογή.
- Η PHP ενημερώνεται τακτικά για να συμβαδίζει με τις τελευταίες τεχνολογικές τάσεις.
- Το άλλο πλεονέκτημα που λαμβάνετε με την PHP είναι ότι είναι μια γλώσσα scripting για διακομιστές. Αυτό σημαίνει ότι πρέπει μόνο να εγκαταστήσετε στον διακομιστή και οι υπολογιστές-πελάτες που ζητούν τα request από το διακομιστή δεν χρειάζεται να έχουν εγκαταστήσει την PHP αλλά μόνο ένα πρόγραμμα περιήγησης ιστού θα ήταν αρκετό.
- Η PHP έχει ενσωματωμένη υποστήριξη με MySQL. Αλλά προφανώς και μπορεί να χρησιμοποιηθεί και με άλλα συστήματα διαχείρισης βάσεων δεδομένων.
  Μπορείτε ακόμα να χρησιμοποιήσετε την PHP με:
  - PostgreSQL
  - Oracle
  - MongoDB

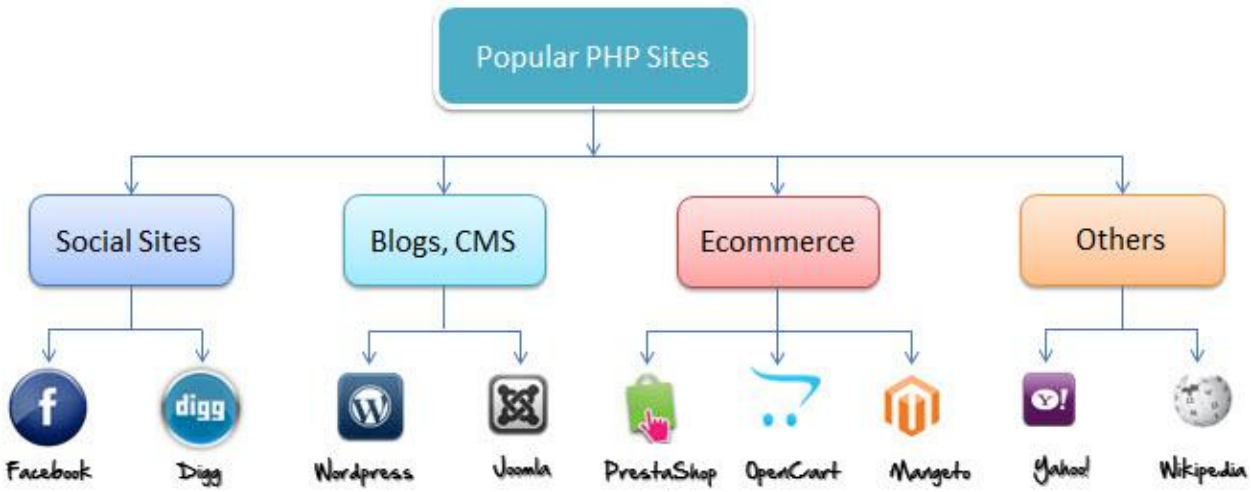
- MariaDB
- Η PHP είναι cross platform. Αυτό σημαίνει ότι δεν έχει κάποια σημασία σε την λειτουργικό σύστημα θα τρέξει είτε αυτό είναι Windows, Linux, Mac OS κλπ.

4.2.1.7 Που χρησιμοποιείται η PHP για το Market Share;

Όσον αφορά το μερίδιο αγοράς, υπάρχουν πάνω από 20 εκατομμύρια ιστότοποι και η εφαρμογές στο διαδίκτυο που αναπτύχθηκαν χρησιμοποιώντας τη γλώσσα προγραμματισμού PHP.

Αυτό μπορεί να αποδοθεί στα σημεία που αναφέρθηκαν παραπάνω.

Το παρακάτω διάγραμμα δείχνει μερικές από τις δημοφιλείς τοποθεσίες που χρησιμοποιούν PHP



Εικόνα 33 (Popular PHP sites)

4.2.1.8 PHP files extensions

Επέκταση αρχείου και ετικέτες για να μπορέσει ο διακομιστής να αναγνωρίσει τα αρχεία και τα γραφήματα PHP, πρέπει να αποθηκεύσουμε το αρχείο με την επέκταση ".php". ΟΙ παλαιότερες επεκτάσεις αρχείων PHP περιλαμβάνουν:

- .phtml
- .php3
- .php4
- .php5
- .phps

Η PHP σχεδιάστηκε για να λειτουργεί με HTML, και ως εκ τούτου, μπορεί να ενσωματωθεί στον κώδικα HTML.

Μπορείτε να δημιουργήσετε αρχεία PHP χωρίς ετικέτες html και που ονομάζεται αρχείο Pure PHP.

Ο διακομιστής ερμηνεύει τον κώδικα PHP και εξάγει τα αποτελέσματα ως κώδικα HTML στα προγράμματα περιήγησης ιστού.

Προκειμένου ο διακομιστής να αναγνωρίσει τον κώδικα PHP από τον κώδικα HTML, πρέπει πάντα να περνάμε τον κώδικα PHP σε ετικέτες PHP.

Μια ετικέτα PHP ξεκινά με το λιγότερο από το σύμβολο που ακολουθείται από το ερωτηματικό και μετά την λέξη "php".

Η PHP είναι μια γλώσσα ευαίσθητη σε πεζά, το VAR είναι διαφορετικό με το var.

Οι ίδιες οι ετικέτες της PHP δεν είναι ευαίσθητες σε πεζά, αλλά συνιστάται να χρησιμοποιούμε μικρά γράμματα. Ο παρακάτω κωδικός απεικονίζει το παραπάνω σημείο.

```
<?php … ?>
```

Θα αναφερθούμε στις γραμμές PHP του κώδικα ως δηλώσεις. Οι δηλώσεις PHP τελειώνουν με ένα semi colon (;). Εάν έχετε μόνο μία δήλωση, μπορείτε να παραλείψετε το semi colon. Εάν έχετε περισσότερες από μία δηλώσεις, τότε πρέπει να τερματίσετε κάθε γραμμή με ένα semi colon. Για λόγους συνέπειας, σας συνιστούμε να τερματίζετε πάντα τις δηλώσεις σας με ένα semi colon. Οι δέσμες ενεργειών PHP εκτελούνται στον διακομιστή. Η έξοδος επιστρέφεται με τη μορφή HTML.

4.2.2 Η χρήση του Framework Laravel

4.2.2.1 Γενικά

Το Laravel είναι ένα web application framework με εκφραστική, κομψή σύνταξη. Πιστεύω ότι η ανάπτυξη πρέπει να είναι μια ευχάριστη, δημιουργική εμπειρία για να εκπληρώσει πραγματικά. Το Laravel επιχειρεί να βγάλει τον πόνο από την ανάπτυξη διευκολύνοντας τα κοινά καθήκοντα που χρησιμοποιούνται στην πλειοψηφία των έργων ιστού, όπως το authentication, routing, sessions, και caching.

Το Laravel στοχεύει να καταστήσει την διαδικασία ανάπτυξης web εφαρμογών πιο ευχάριστη για τον developer χωρίς να αλλάξει τον τρόπο λειτουργίας της εφαρμογής. Οι ευτυχείς προγραμματιστές κάνουν τον κώδικα καλύτερο. Για το σκοπό αυτό το Laravel, προσπάθησε να συνδυάσει τα καλύτερα από όσα έχουμε δει σε άλλα web frameworks, συμπεριλαμβανομένων των framework που εφαρμόζονται σε άλλες γλώσσες, όπως το Ruby on Rails, το ASP.NET MVC και το Sinatra.

Το Laravel είναι προσβάσιμο, αλλά ισχυρό, παρέχοντας ισχυρά εργαλεία που απαιτούνται για μεγάλες και ανθεκτικές εφαρμογές. Μια θαυμάσια αναστροφή του container ελέγχου, του migration συστήματος και την υποστήριξη unit testing σας δίνουν τα εργαλεία που χρειάζεστε για να χτίσετε οποιαδήποτε εφαρμογή.

4.2.2.2 Μερικά γεγονότα για τον Laravel

- Το Laravel αναπτύχθηκε από τον Taylor Otwell τον Ιούλιο του 2011 και απελευθερώθηκε περισσότερο από πέντε χρόνια μετά την κυκλοφορία του Codeigniter.
- Το Laravel είναι ένα PHP-based framework όπως το Codeigniter.
- Το Laravel είναι ένα από τα framework ανοιχτού κώδικα PHP.
- Το Laravel ακολουθεί το αρχιτεκτονικό πρότυπο (MVC).
- Το Laravel είναι ένα από τα πιο δημοφιλή PHP framework μετά τον Codeigniter.

4.2.2.3 Χαρακτηριστικά του Laravel

Θα αναφέρω ονομαστικά κάποια από τα βασικά χαρακτηριστικά του Laravel και παρακάτω θα διατυπώσω μερικά λόγια για το καθένα.

- Event and Command Bus
- Modularity
- ORM (Object Relational Mapper) features
- Authentication and authorization of users
- Routing controllers
- Configuration management
- Testability
- Provides template engine
- Building schemas
- E-mailing facilities
- Redis cache
- Queue Cron jobs

4.2.2.4 Τι είναι composer;

Ο composer είναι ένα εργαλείο που περιλαμβάνει όλες τις εξαρτήσεις και τις βιβλιοθήκες. Επιτρέπει σε ένα χρήστη να δημιουργήσει ένα έργο σε σχέση με το προαναφερθέν framework (για παράδειγμα, αυτά που χρησιμοποιούνται στην εγκατάσταση του Laravel). Οι βιβλιοθήκες τρίτων μπορούν να εγκατασταθούν εύκολα με τη βοήθεια του composer.

Όλα τα dependencies σημειώνονται στο αρχείο composer.json που τοποθετείται στο φάκελο προέλευσης.

4.2.2.5 Τι είναι Artisan;

Η διεπαφή γραμμής εντολών που χρησιμοποιείται στο Laravel ονομάζεται Artisan. Περιλαμβάνει ένα σύνολο εντολών που βοηθά στην κατασκευή μιας εφαρμογής ιστού. Αυτές οι εντολές ενσωματώνονται από το Symphony framework, με αποτέλεσμα πρόσθετα χαρακτηριστικά στο Laravel 5.

Στην παρακάτω εικόνα θα δούμε μερικά από τα βασικά artisan commands.

Command#	Description & Output
	Check the current version of laravel installation?
php artisan –version	Output: Laravel Framework version 5.2.45
	Put Laravel application in "maintenance mode"
php artisan down	Output: Application is now in maintenance mode.
	Display the environment laravel is running
php artisan env	Output: Current application environment: local
php artisan migrate	Run Database migrations This executes all the defined migrations and create database tables.
php artisan serve	To start Laravel project. By, default this hosts the application locally at localhost:8000 You can server with different hostname and post using "-host" and "-port" options respectively.
php artisan up	Bring UP the laravel application out of maintenance mode
php artisan auth:clear-resets	Flush the expired password tokens
php artisan cache:clear	Flush the application cache
php artisan cache:table	Create a migration for the cache database table
php artisan config:cache	Create a cache file for faster configuration loading
php artisan config:clear	Remove the configuration cache file
php artisan make:auth	Scaffold basic login and registration views and routes
php artisan make:controller TechCluesBlog	Create a new controller class using artisan command
php artisan make:migration my_blog_post	Create a new migration file. Output: Created Migration: 2019_01_27_094045_my_blog_post The new migration file 2019_01_27_094045_my_blog_post.php will be created under ../database/migrations./
php artisan make:model MyBlogPost	Create a new Eloquent model class. Output: Model created successfully. The new Model file "MyBlogPost.php" will be created in app/Models or ../app/..
php artisan route:list	List all registered routes
php artisan route:clear	Remove the route cache file
php artisan route:cache	Create a route cache file for faster route registration
php artisan vendor:publish	Publish any publishable assets from vendor packages
php artisan view:clear	Clear all compiled view files

Εικόνα 34 (Laravel Artisan Commands)

4.2.2.6 Modularity

Το Laravel παρέχει 20 ενσωματωμένες βιβλιοθήκες και ενότητες που βοηθούν στην ενίσχυση της εφαρμογής. Κάθε μονάδα είναι ενσωματωμένη με τον διαχειριστή Composer Dependency που διευκολύνει τις ενημερώσεις.

4.2.2.7 Testability

Το Laravel περιλαμβάνει χαρακτηριστικά και βοηθήματα που βοηθούν στη δοκιμή μέσω διαφόρων δοκιμαστικών εργαλείων. Αυτή η λειτουργία βοηθά στη διατήρηση του κώδικα σύμφωνα με τις απαιτήσεις.

4.2.2.8 Routing

Το Laravel παρέχει μια ευέλικτη προσέγγιση στο χρήστη για τον καθορισμό των διαδρομών στην εφαρμογή web. Η δρομολόγηση βοηθάει στην κλιμάκωση της εφαρμογής με καλύτερο τρόπο και αυξάνει την απόδοσή της.

4.2.2.9 Configuration Management

Μια εφαρμογή web σχεδιασμένη στο Laravel θα λειτουργεί σε διαφορετικά περιβάλλοντα, πράγμα που σημαίνει ότι θα υπάρξει μια σταθερή αλλαγή στη διαμόρφωσή της. Το Laravel παρέχει μια συνεπή προσέγγιση για να χειριστεί τη διαμόρφωση με αποτελεσματικό τρόπο.

4.2.2.10 Query Builder and ORM

Το Laravel ενσωματώνει ένα εργαλείο δημιουργίας ερωτημάτων που βοηθά στην αναζήτηση βάσεων δεδομένων χρησιμοποιώντας διάφορες μεθόδους απλής αλυσίδας. Παρέχει την εφαρμογή ORM (Object Relational Mapper) και την εφαρμογή Active Record που ονομάζεται Eloquent.

4.2.2.11 Schema Builder

Το Schema Builder διατηρεί τους ορισμούς βάσεων δεδομένων και το σχήμα σε κώδικα PHP. Διατηρεί επίσης ένα ίχνος αλλαγών σε σχέση με τα database migrations.

4.2.2.12 Template Engine

Το Laravel χρησιμοποιεί το blade template engine, μια ελαφριά γλώσσα προτύπου που χρησιμοποιείται για το σχεδιασμό ιεραρχικών μπλοκ και layout με προκαθορισμένα μπλοκ που περιλαμβάνουν δυναμικό περιεχόμενο.

4.2.2.13 E-mail εγκαταστάσεις

Το Laravel περιλαμβάνει μια κλάση e-mail που βοηθά στην αποστολή αλληλογραφίας με πλούσιο περιεχόμενο και συνημμένα από την εφαρμογή Ιστού.

4.2.2.14 Authentication

Ο έλεγχος ταυτότητας χρήστη είναι μια κοινή λειτουργία στις εφαρμογές ιστού. Το Laravel διευκολύνει το σχεδιασμό του ελέγχου ταυτότητας καθώς περιλαμβάνει χαρακτηριστικά όπως register, forgot password και υπενθυμίσεις κωδικού πρόσβασης.

4.2.2.15 Redis Caching System

Το Laravel χρησιμοποιεί το Redis για να συνδεθεί με ένα υπάρχον session και μια προσωρινή μνήμη γενικού σκοπού. Ο Redis αλληλοεπιδρά άμεσα με τα session.

4.2.2.16 Queues

Το Laravel περιλαμβάνει υπηρεσίες ουράς όπως η αποστολή ηλεκτρονικού ταχυδρομείου σε μεγάλο αριθμό χρηστών ή μια καθορισμένη εργασία Cron. Αυτές οι ουρές βοηθούν στην ολοκλήρωση των εργασιών με ευκολότερο τρόπο, χωρίς να περιμένουν την ολοκλήρωση της προηγούμενης εργασίας.

4.2.2.17 Event and Command Bus

Από την έκδοση Laravel 5.1 και μετά συμπεριλαμβάνετε το Command Bus, το οποίο βοηθά στην εκτέλεση εντολών και την αποστολή συμβάντων με έναν απλό τρόπο. Οι εντολές στο Laravel ενεργούν σύμφωνα με τον κύκλο ζωής της εφαρμογής.

4.3 Application Programming Interface (API)

4.3.1 JSON

4.3.1.1 Τι είναι Json;

Το JSON αλλιώς JavaScript Object Notation είναι ένας τρόπος για την αποθήκευση πληροφοριών με έναν οργανωμένο και εύκολο τρόπο πρόσβασης. Με λίγα λόγια, μας δίνει μια ανθρώπινη συλλογή δεδομένων που μπορούμε να έχουμε πρόσβαση με έναν πραγματικά λογικό τρόπο.

Αποθήκευση δεδομένων JSON

Ως απλό παράδειγμα, το πως μπορεί να γραφτεί ένα JSON είναι ο εξής τρόπος:

```
var Name = {
   "age" : "25",
   "hometown" : "Aigio, GREECE",
   "gender" : "male"
};
```

Εικόνα 35 (JSON example)

Αυτό δημιουργεί ένα αντικείμενο στο οποίο έχουμε πρόσβαση χρησιμοποιώντας τη μεταβλητή Name. Με την ενσωμάτωση της τιμής της μεταβλητής σε brackets, δηλώνουμε ότι η τιμή είναι ένα αντικείμενο. Μέσα στο αντικείμενο, μπορούμε να δηλώσουμε οποιοδήποτε αριθμό ιδιοτήτων χρησιμοποιώντας ένα ζεύγος "όνομα": "τιμή", διαχωρισμένο με κόμματα. Για να αποκτήσετε πρόσβαση στις πληροφορίες που είναι αποθηκευμένες στο Name, μπορούμε απλά να αναφέρουμε το όνομα που χρειαζόμαστε. Για παράδειγμα, για να αποκτήσετε πρόσβαση σε πληροφορίες σχετικά με εμένα, θα μπορούσαμε να χρησιμοποιήσουμε τα ακόλουθα αποσπάσματα:

```
document.write('George is ' Name.age); // Output: George is 24

document.write('George is a ' Name.gender); // Output: George is a male
```

Εικόνα 36 (JSON exmple output)

Αποθήκευση δεδομένων JSON σε Array

Ένα ελαφρώς πιο περίπλοκο παράδειγμα περιλαμβάνει την αποθήκευση δύο ανθρώπων σε μια μεταβλητή. Για να γίνει αυτό, περικλείουμε πολλαπλά αντικείμενα σε αγκύλες, που σημαίνει μια διάταξη. Για παράδειγμα, εάν έπρεπε να συμπεριλάβω πληροφορίες για τον εαυτό μου και τον αδερφό μου σε μια μεταβλητή, θα μπορούσα να χρησιμοποιήσω τα εξής:

```
var family = [{
   "name" : "George",
   "age" : "25",
   "gender" : "male"
 },{
   "name" : "Michael",
   "age" : "23",
   "gender" : "male"
}];
```

Εικόνα 37 (JSON example array)

Για να πάρετε πρόσβαση σε αυτές τις πληροφορίες, πρέπει να αποκτήσετε πρόσβαση στο ευρετήριο συστοιχιών του ατόμου που επιθυμούμε να αποκτήσουμε πρόσβαση. Για παράδειγμα, θα χρησιμοποιούσαμε το ακόλουθο απόσπασμα για την πρόσβαση στις πληροφορίες που είναι αποθηκευμένες στην μεταβλητή family:

```
document.write(family[1].name); // Output: Michael

document.write(family[0].age); // Output: 25
```

Εικόνα 38 (JSON example array output)

Nesting JSON Data

Ένας άλλος τρόπος για να αποθηκεύσετε πολλούς ανθρώπους στη μεταβλητή μας θα ήταν να φωλιάζουμε αντικείμενα. Για να γίνει αυτό, θα δημιουργούσαμε κάτι παρόμοιο με τον εξής τρόπο:

```
var family = {
    "george" : {
        "name" : "George Tsachrelias",
        "age" : "25",
        "gender" : "male"
    },
    "mick" : {
        "name" : "Michael Tsachrelias ",
        "age" : "23",
        "gender" : "male"
    }
}
```

Εικόνα 39 (JSON Example Object)

Η πρόσβαση σε πληροφορίες σε ένθετα αντικείμενα είναι λίγο πιο εύκολη στην κατανόηση. Για την πρόσβαση σε πληροφορίες στο αντικείμενο, θα χρησιμοποιήσαμε το ακόλουθο απόσπασμα:

```
document.write(family.george.name); // Output: George Tsachrelias

document.write(family.mick.age); // Output: 23

document.write(family.george.gender); // Output: male
```

Εικόνα 40 (JSON Example Object Output)

Τα nested JSON και τα arrays μπορούμε να συνδυάσουμε με βάση τις ανάγκες μας ανάλογα με το τι θέλουμε να αποθηκεύσομε και πόσος θα είναι ο όγκος αποθήκευσης.

4.3.1.2 Γιατί χρειαζόμαστε το Json;

Με την άνοδο των web site που λειτουργούν την μέθοδο AJAX, όλο και πιο σημαντικό είναι να μπορούν οι τοποθεσίες να φορτώνουν δεδομένα γρήγορα και ασύγχρονα ή στο παρασκήνιο χωρίς να καθυστερούν την απόδοση της σελίδας. Η εναλλαγή των περιεχομένων ενός συγκεκριμένου στοιχείου στο καθορισμένο μας layout χωρίς να απαιτείται ανανέωση η σελίδα προσθέτει έναν παράγοντα "wow" στις εφαρμογές μας, για να μην αναφέρουμε την πρόσθετη ευκολία για τους χρήστες μας. Λόγω της δημοτικότητας και της ευκολίας των κοινωνικών μέσων, πολλοί ιστότοποι βασίζονται στο περιεχόμενο που παρέχεται από ιστότοπους όπως το Twitter, το Flickr και άλλοι. Αυτοί οι ιστότοποι παρέχουν ροές RSS, οι οποίες είναι εύκολο να εισαχθούν και να χρησιμοποιηθούν από την πλευρά του διακομιστή, αλλά αν προσπαθήσουμε να τις φορτώσουμε με AJAX, αυτό είναι αδύνατον γιατί μπορούμε να φορτώσουμε μια ροή RSS μόνο αν το ζητάμε από τον ίδιο τομέα στον οποίο φιλοξενείται. Μια προσπάθεια φόρτωσης της ροής RSS του λογαριασμού μου στο Flickr μέσω της μεθόδου $.ajax () της jQuery έχει ως αποτέλεσμα το ακόλουθο σφάλμα JavaScript:

```
[Exception... "Access to restricted URI denied" code: "1012"

nsresult: "0x805303f4 (NS_ERROR_DOM_BAD_URI)"

location: "http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js Line: 19"]
```

Εικόνα 41 (Ajax Error JS Json)

Το JSON μας επιτρέπει να ξεπεράσουμε το πρόβλημα cross domain επειδή μπορούμε να χρησιμοποιήσουμε μια μέθοδο που ονομάζεται JSONP που χρησιμοποιεί μια λειτουργία επανάκλησης για να στείλει τα δεδομένα JSON πίσω στον domain μας. Είναι αυτή η δυνατότητα που κάνει το JSON τόσο απίστευτα χρήσιμο, καθώς ανοίγει πολλές πόρτες που πριν ήταν δύσκολο να εργαστούν.

4.3.2 Τι είναι RESTful API;

4.3.2.1 Γενικά

Το REST είναι ακρωνύμιο για την μεταφορά κρατικής αντιπροσωπείας. Είναι αρχιτεκτονικό στυλ για διανεμημένα συστήματα hypermedia και παρουσιάστηκε από τον Roy Fielding το 2000 για πρώτη φορά στη διάσημη διατριβή του.

Όπως κάθε άλλο αρχιτεκτονικό στυλ, το REST έχει επίσης 6 δικούς του περιορισμούς που πρέπει να ικανοποιηθούν εάν μια διεπαφή πρέπει να αναφέρεται ως RESTful. Αυτές οι αρχές παρατίθενται παρακάτω.

- **Client-Server** - Διαχωρίζοντας την διεπαφής του client από την αποθήκευσης δεδομένων, βελτιώνουμε τη ευκολία τις διεπαφής client σε πολλαπλές πλατφόρμες και φτιάχνουμε την δυνατότητα κλιμάκωσης απλοποιώντας τα στοιχεία του server.
- **Stateless** - Κάθε αίτημα client-server πρέπει να έχει όλες τις πληροφορίες που είναι απαραίτητες για την κατανόηση του αιτήματος και δεν μπορεί να εκμεταλλευτεί οποιοδήποτε αποθηκευμένο περιβάλλον στο διακομιστή. Συνεπώς, όλη η σύνδεση διατηρείται από τον client.
- **Cacheable** – Η cache απαιτεί τα δεδομένα εντός μιας απάντησης σε ένα αίτημα να είναι σιωπηρά ή ρητά επισημασμένα ως αποθηκευμένα στο αρχείο ή μη αποθηκευμένα στο αρχείο. Εάν η απόκριση είναι αποθηκευμένη στο αρχείο cache, τότε η μνήμη cache του πελάτη έχει το δικαίωμα να επαναχρησιμοποιήσει αυτά τα δεδομένα απόκρισης για μεταγενέστερα, ισοδύναμα αιτήματα.
- **Uniform Interface** - Με την εφαρμογή της αρχής της software engineer τεχνολογίας λογισμικού στη διεπαφή στοιχείων, η γενική αρχιτεκτονική του συστήματος απλοποιείται και βελτιώνεται η ορατότητα των αλληλεπιδράσεων. Προκειμένου να επιτευχθεί ομοιόμορφη διεπαφή, απαιτούνται πολλοί αρχιτεκτονικοί περιορισμοί για την καθοδήγηση της συμπεριφοράς των εξαρτημάτων. Το REST ορίζεται από τέσσερις περιορισμούς στη διασύνδεση: προσδιορισμός των πόρων, χειραγώγηση των πόρων μέσω παραστάσεων, self-descriptive μηνύματα και hypermedia.
- **Layered system** - Το στρώμα του στυλ συστήματος επιτρέπει σε μια αρχιτεκτονική να αποτελείται από ιεραρχικά στρώματα περιορίζοντας τη συμπεριφορά των συστατικών

έτσι ώστε κάθε συστατικό να μην μπορεί να "δει" πέρα από το άμεσο στρώμα με το οποίο αλληλοεπιδρούν.

- **Code on demand (προαιρετικό)** - Η λειτουργία REST επιτρέπει την επέκταση της λειτουργικότητας του πελάτη με τη λήψη και την εκτέλεση κώδικα με form of applets ή scripts. Αυτό απλοποιεί τους πελάτες μειώνοντας τον αριθμό των λειτουργιών που απαιτούνται για την προετοιμασία τους.

4.3.2.2. Resource

Η βασική αφαίρεση των πληροφοριών στο REST είναι ένας πόρος. Οποιεσδήποτε πληροφορίες μπορούν να ονομαστούν μπορούν να είναι ένας πόρος: ένα έγγραφο ή μια εικόνα, μια χρονική υπηρεσία, μια συλλογή από άλλους πόρους, ένα μη εικονικό αντικείμενο (π.χ. ένα άτομο) κ.ο.κ.

Το REST χρησιμοποιεί ένα αναγνωριστικό πόρων για τον προσδιορισμό του συγκεκριμένου πόρου που εμπλέκεται σε μια αλληλεπίδραση μεταξύ των στοιχείων.

Η κατάσταση του πόρου σε κάθε συγκεκριμένη χρονική σήμανση είναι γνωστή ως αντιπροσώπευση πόρων. Μια αναπαράσταση αποτελείται από δεδομένα, μεταδεδομένα που περιγράφουν τις συνδέσεις δεδομένων και hypermedia που μπορούν να βοηθήσουν τους πελάτες που βρίσκονται σε μετάβαση στην επόμενη επιθυμητή κατάσταση.

Η μορφή δεδομένων μιας αναπαράστασης είναι γνωστή ως τύπος μέσου. Ο τύπος μέσου προσδιορίζει μια προδιαγραφή που καθορίζει τον τρόπο επεξεργασίας μιας παράστασης. Ένα πραγματικά αληθινό API μοιάζει με υπερκειμένου. Κάθε διευθυνόμενη μονάδα πληροφοριών μεταφέρει μια διεύθυνση, είτε ρητά (π.χ. χαρακτηριστικά σύνδεσης και ταυτότητας) είτε σιωπηρά (π.χ. που προέρχεται από τον ορισμό του τύπου μέσων και τη δομή αναπαράστασης).

4.3.2.3 Resources Methods

Ένα άλλο σημαντικό πράγμα που συνδέεται με το REST είναι οι μέθοδοι πόρων που πρέπει να χρησιμοποιηθούν για την εκτέλεση της επιθυμητής μετάβασης. Ένας μεγάλος αριθμός ατόμων σχετίζεται λανθασμένα με μεθόδους πόρων σε μεθόδους HTTP GET / PUT / POST / DELETE.

Ο Roy Fielding ουδέποτε ανέφερε οποιαδήποτε σύσταση σχετικά με τη μέθοδο που θα χρησιμοποιηθεί σε ποια κατάσταση. Το μόνο που τονίζει είναι ότι πρέπει να είναι ομοιόμορφη διεπαφή. Εάν αποφασίσετε ότι το HTTP POST θα χρησιμοποιηθεί για την ενημέρωση ενός πόρου και όχι όπως οι περισσότεροι άνθρωποι συνιστούν τον HTTP PUT, τότε η διεπαφή εφαρμογών θα είναι RESTful.

Στην ιδανική περίπτωση, όλα όσα χρειάζονται για να αλλάξουν την κατάσταση πόρων θα αποτελούν μέρος της απόκρισης API για τον συγκεκριμένο πόρο συμπεριλαμβανομένων μεθόδων και σε ποια κατάσταση θα εγκαταλείψουν την εκπροσώπηση.

Το REST και το HTTP δεν είναι ίδια παρόλα αυτά, επειδή το REST σκοπεύει επίσης να καταστήσει τον ιστό (internet), συνίσταται να χρησιμοποιούμε αυστηρότερα τις αρχές REST. Και αυτό είναι από όπου οι άνθρωποι προσπαθούν να αρχίσουν να συγκρίνουν το REST με τον ιστό (HTTP). Ο Roy, με τη διδακτορική του διατριβή, δεν ανέφερε καμία οδηγία εφαρμογής

συμπεριλαμβανομένης οποιασδήποτε προτίμησης πρωτοκόλλου και HTTP. Μέχρι την ώρα, τιμάτε τις 6 κατευθυντήριες αρχές του REST, μπορείτε να καλέσετε τη διεπαφή σας RESTful.

Με απλούστερα λόγια, στο αρχιτεκτονικό στυλ REST, τα data και η λειτουργικότητα λέγονται πόροι και προσπελάζονται με χρήση Ενιαίων Αναγνωριστικών Πόρων (Uniform Resource Identifiers - URI). Οι πόροι λειτουργούν με τη χρήση ενός συνόλου απλών, καλά καθορισμένων λειτουργιών. Οι πελάτες και οι διακομιστές ανταλλάσσουν παραστάσεις πόρων χρησιμοποιώντας μια τυποποιημένη διεπαφή και πρωτόκολλα HTTP.

Οι πόροι αποσυνδέονται από την αναπαράστασή τους, για να δείξουμε το περιεχόμενο σε διαφορετικές μορφές και να είναι προσβάσιμο σε HTML, XML, απλό κείμενο, PDF, JPEG, JSON και άλλα.

Τα metadata σχετικά με τον πόρο είναι διαθέσιμα, για τον έλεγχο της προσωρινής αποθήκευσης, το debug σφαλμάτων μετάδοσης, την κατάλληλης μορφής αναπαράστασης και την εκτέλεση ελέγχου πρόσβασης. Και το πιο σημαντικό, κάθε αλληλεπίδραση με έναν πόρο είναι stateless.

Όλες αυτές οι αρχές βοηθούν τις εφαρμογές RESTful να είναι απλές, ελαφρές και γρήγορες.

4.4 Mobile Applications

4.4.1 Γλώσσα Προγραμματισμού Android

4.4.1.1 Γενικά

Παρόλο που μια εφαρμογή Android μπορεί να διατίθεται από προγραμματιστές μέσω των ιστοτόπων τους, οι περισσότερες εφαρμογές Android μεταφορτώνονται και δημοσιεύονται στο Android Store, ένα ηλεκτρονικό κατάστημα αφιερωμένο σε αυτές τις εφαρμογές. Το Android Store διαθέτει και δωρεάν εφαρμογές και εφαρμογές με τιμές.

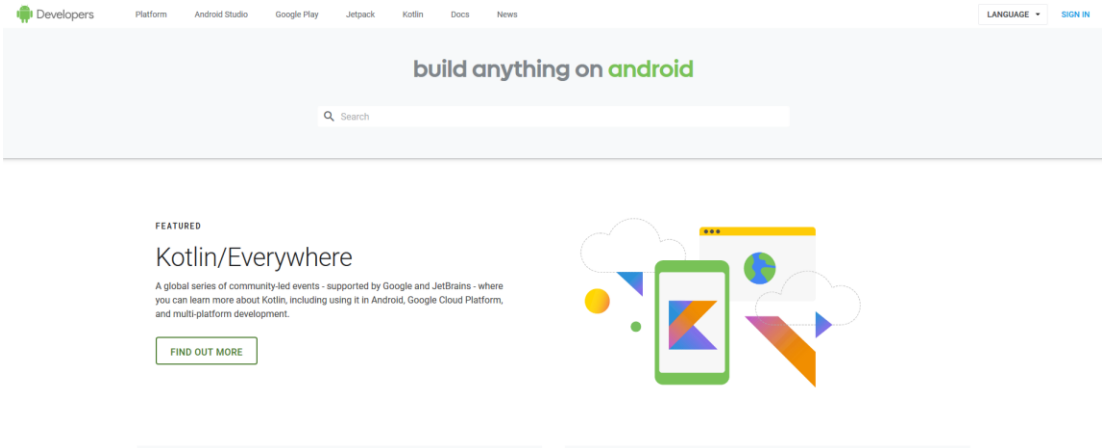
Τα Android Application συντάσσονται στην γλώσσα προγραμματισμού Java και χρησιμοποιούν τις βασικές βιβλιοθήκες Java. Καταρτίζονται για πρώτη φορά σε εκτελέσιμα αρχεία Dalvik για να τρέχουν στην virtual machine Dalvik, η οποία είναι μια VM ειδικά σχεδιασμένη για κινητές συσκευές.

Οι προγραμματιστές μπορούν να κάνουν λήψη του KIT ανάπτυξης λογισμικού Android (SDK) από την τοποθεσία Android. Το SDK περιλαμβάνει εργαλεία, δείγμα κώδικα και σχετικά έγγραφα για τη δημιουργία εφαρμογών Android.

Οι αρχάριοι προγραμματιστές που απλά θέλουν να παίξουν με τον προγραμματισμό Android μπορούν να κάνουν χρήση του App Inventor. Χρησιμοποιώντας αυτήν την ηλεκτρονική εφαρμογή, ένας χρήστης μπορεί να κατασκευάσει μια εφαρμογή Android σαν να συνθέτει κομμάτια ενός παζλ.

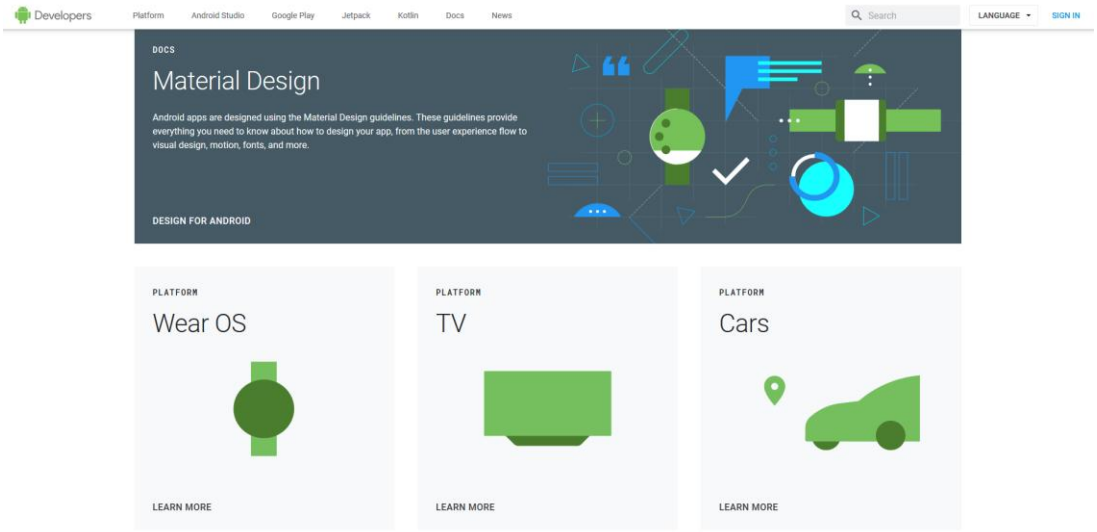
4.4.1.2 Βήμα - Βήμα πως να ξεκινήσω

1. Official Android Website



Επισκεφτείτε τον επίσημο ιστότοπο Android Developer. Για να είμαι ειλικρινής, είναι πολύ σημαντικό να διαβάσετε ολόκληρη την ιστοσελίδα για να καταλάβετε το οικοσύστημα και την ποικιλία λύσεων, ιδεών και τεχνολογιών που συνδέονται με το Android. Προφανώς, δεν θα καταλάβετε και θα θυμάστε τα πάντα, αλλά θα ξέρετε πού να αναζητήσετε στο μέλλον.

2. Γνωρίστε το Material Design

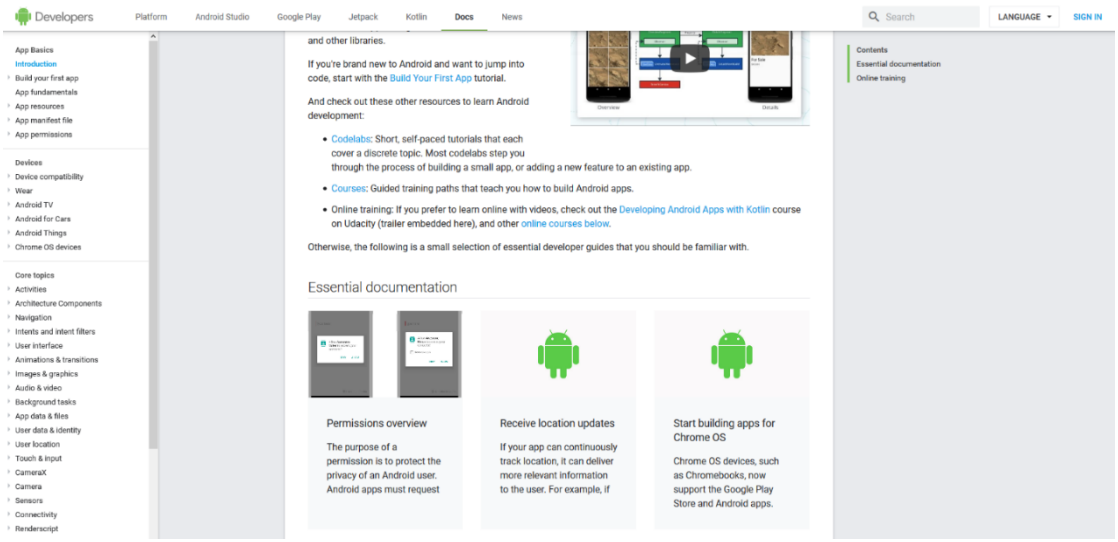


Ο ίδιος κανόνας όπως παραπάνω σάρωση ολόκληρου του ιστότοπου για να καταλάβετε πώς είναι χτισμένο και ποια είναι η ιδέα πίσω από αυτό. Προσπαθήστε να το αισθανθείτε. Να είστε το υλικό. Σκεφτείτε σαν υλικό. Συμπεριλαμβάνετε το υλικό.

3. Κατεβάστε το Android Studio IDE

Είναι αναμφισβήτητα το καλύτερο IDE για το Android και είναι από την Google.Αν είναι ένα νέο IDE για εσάς μην φοβάστε.

4. Γράψε κώδικα



Ήρθε η ώρα να κοιτάξουμε λίγο τον κώδικα και να γράψουμε κάτι. Η πρακτική μαζί με μια θεωρία είναι ένας από τους καλύτερους τρόπους μάθησης.

Επιστρέψτε στην επίσημη ιστοσελίδα Android και αυτή τη φορά πηγαίνετε στο Develop. Υπάρχουν δύο πιο σημαντικά τμήματα:

a. **Training** - μπορείτε να βρείτε πολλά χρήσιμα παραδείγματα. Πραγματοποιήστε όλα αυτά τα μαθήματα πριν ξεκινήσετε τον προγραμματισμό πραγματικά.
b. **API Guides** - Ξέρω ότι θα είναι δύσκολο να περάσετε από όλα αυτά τα κείμενα, περιγραφές, παραδείγματα και ούτω καθεξής, αλλά αξίζει τον κόπο. Αυτό το τμήμα είναι πολύ σημαντικό.

4.4.2 Η Χρήση του Framework React Native / JSX

4.4.2.1 Τι είναι JSX;

Σε κάθε συνοπτική αναζήτηση στο Διαδίκτυο που αναζητά υλικό React, δεν υπάρχει αμφιβολία ότι έχετε ήδη εισέλθει στους όρους JSX, ES5 και ES6. Αυτά τα αδιαφανή ακρωνύμια μπορούν να μπερδευτούν γρήγορα.

Το ES5 (το ES σημαίνει ECMAScript) είναι βασικά "κανονικό JavaScript". Η πέμπτη ενημέρωση για το JavaScript, ES5 ολοκληρώθηκε το 2009. Έχει υποστηριχθεί από όλα τα

μεγάλα προγράμματα περιήγησης για αρκετά χρόνια. Επομένως, εάν έχετε γράψει ή έχετε δει κάποιο JavaScript στο πρόσφατο παρελθόν, πιθανόν να ήταν ES5.

Το ES6 είναι μια νέα έκδοση του JavaScript που προσθέτει μερικές ωραίες συντακτικές και λειτουργικές προσθήκες. Ολοκληρώθηκε το 2015. Το ES6 υποστηρίζεται σχεδόν πλήρως από όλα τα μεγάλα προγράμματα περιήγησης. Αλλά θα υπάρξει κάποιο χρονικό διάστημα μέχρι να καταργηθούν σταδιακά οι παλαιότερες εκδόσεις των προγραμμάτων περιήγησης ιστού. Για παράδειγμα, ο Internet Explorer 11 δεν υποστηρίζει το ES6, αλλά έχει περίπου το 12% του μεριδίου αγοράς του προγράμματος περιήγησης.

4.4.2.2 Τι είναι React Native;

Γενικά

React Native είναι ένα framework JavaScript για τη σύνταξη πραγματικών εφαρμογών κινητής τηλεφωνίας για iOS και Android. Βασίζεται στη βιβλιοθήκη JavaScript React, του Facebook για τη δημιουργία διεπαφών χρήστη, αλλά αντί να στοχεύει το πρόγραμμα περιήγησης, στοχεύει σε φορητές πλατφόρμες. Με άλλα λόγια: οι υπεύθυνοι ανάπτυξης ιστού μπορούν τώρα να γράψουν εφαρμογές για κινητά που φαίνονται και νιώθουν αληθινά "native", όλα από την άνεση μιας βιβλιοθήκης JavaScript που ήδη γνωρίζουμε και αγαπάμε. Επιπλέον, επειδή το μεγαλύτερο μέρος του κώδικα που γράφετε μπορεί να μοιραστεί μεταξύ των πλατφορμών, το React Native καθιστά εύκολη την ταυτόχρονη ανάπτυξη τόσο για Android όσο και για iOS.

Παρόμοια με το React για τον Παγκόσμιο Ιστό, οι React Native εφαρμογές είναι γραμμένες χρησιμοποιώντας ένα μείγμα JavaScript και XML-esque markup, γνωστό ως JSX. Στη συνέχεια, κάτω από την κουκούλα, το React Native "bridge" επικαλείται τα εγγενή APIs rendering σε Objective-C (για iOS) ή Java (για Android). Έτσι, η εφαρμογή σας θα κάνει χρήση πραγματικών στοιχείων κινητού UI, όχι προβολές ιστού, και θα φαίνεται και θα νιώθει σαν οποιαδήποτε άλλη εφαρμογή για κινητά. Αντίστοιχα, το Native εκθέτει διεπαφές JavaScript για API πλατφόρμες, έτσι ώστε οι εφαρμογές σας React Native να έχουν πρόσβαση σε λειτουργίες πλατφόρμας όπως η κάμερα του τηλεφώνου ή η τοποθεσία του χρήστη.

Αντ' αυτού, το Native υποστηρίζει και για iOS και για Android και έχει τη δυνατότητα να επεκταθεί και σε μελλοντικές πλατφόρμες. Παρακάτω θα αναλύσουμε και θα καλύψουμε τόσο το iOS όσο και το Android. Η συντριπτική πλειοψηφία του κώδικα που γράφουμε θα είναι cross-platform. Και ναι μπορείτε πραγματικά να χρησιμοποιήσετε το React Native για την κατασκευή κινητών εφαρμογών έτοιμων για παραγωγή.

a. Πλεονεκτήματα του React Native
   Το γεγονός ότι το React Native αποδίδει πραγματικά τα API πρότυπων rendering της πλατφόρμας φιλοξενίας του επιτρέπει να ξεχωρίζει από τις περισσότερες υπάρχουσες μεθόδους ανάπτυξης εφαρμογών μεταξύ των πλατφορμών, όπως Cordova ή Ionic. Οι υπάρχουσες μέθοδοι γραφής κινητών εφαρμογών που χρησιμοποιούν συνδυασμούς

JavaScript, HTML και CSS τυπικά αποδίδουν χρησιμοποιώντας τις προβολές ιστού. Ενώ η προσέγγιση αυτή μπορεί να λειτουργήσει, επίσης έρχεται με μειονεκτήματα, ειδικά γύρω από τις επιδόσεις. Επιπλέον, συνήθως δεν έχουν πρόσβαση στο σύνολο των φυσικών στοιχείων UI της πλατφόρμας υποδοχής. Όταν αυτά τα framework προσπαθούν να μιμηθούν φυσικά στοιχεία UI, τα αποτελέσματα συνήθως πέφτουν έξω. Η αντίστροφη μηχανουργική επεξεργασία όλων των λεπτών λεπτομερειών για πράγματα όπως κινούμενα σχέδια παίρνει μια τεράστια προσπάθεια και μπορούν γρήγορα να ξεπεραστούν.

Αντίθετα, το React Native μεταφράζει στην πραγματικότητα τη σήμανσή σας σε πραγματικά, εγγενή στοιχεία UI, αξιοποιώντας τα υπάρχοντα μέσα για την απόδοση απόψεων σε οποιαδήποτε πλατφόρμα εργάζεστε. Επιπλέον, το React λειτουργεί ξεχωριστά από το κύριο νήμα του UI, έτσι ώστε η εφαρμογή σας να μπορεί να διατηρεί υψηλή απόδοση χωρίς να θυσιάζει τη δυνατότητα. Ο κύκλος επικαιροποίησης του React Native είναι ο ίδιος με αυτόν του React: όταν υποστηρίζει ή αλλάζει κατάσταση, το React Native επαναφέρει τις προβολές. Η κύρια διαφορά μεταξύ του React Native και του React στο πρόγραμμα περιήγησης είναι ότι το React Native το κάνει αυτό με την αξιοποίηση των βιβλιοθηκών UI της πλατφόρμας φιλοξενίας του, αντί να χρησιμοποιεί σήμανση HTML και CSS.

Για προγραμματιστές που έχουν συνηθίσει να εργάζονται στο Web με το React, αυτό σημαίνει ότι μπορείτε να γράφετε εφαρμογές για κινητά με την απόδοση και την εμφάνιση μιας εγγενής εφαρμογής, ενώ χρησιμοποιείτε οικεία εργαλεία. Αντίστοιχα, το Native αντιπροσωπεύει μια βελτίωση σε σχέση με την κανονική εξέλιξη της κινητής τηλεφωνίας σε δύο άλλους τομείς: την εμπειρία των προγραμματιστών και το αναπτυξιακό δυναμικό μεταξύ των πλατφορμών.

Εμπειρία προγραμματιστών
Αν έχετε αναπτύξει προηγουμένως για κινητά, ίσως να εκπλαγείτε από το πόσο εύκολο είναι να εργαστείτε με την React Native. Η ομάδα React Native έχει επεξεργαστεί ισχυρά εργαλεία ανάπτυξης και σημαντικά μηνύματα λάθους στο πλαίσιο, οπότε η συνεργασία με τα ισχυρά εργαλεία αποτελεί φυσικό μέρος της αναπτυξιακής σας εμπειρίας.

Για παράδειγμα, επειδή το React Native είναι "απλά" JavaScript, δεν χρειάζεται να ξαναφτιάξετε την εφαρμογή σας για να δείτε τις αλλαγές σας. Αντίθετα, μπορείτε να πατήσετε το Command + R για να ανανεώσετε την εφαρμογή σας όπως θα κάνατε σε οποιαδήποτε άλλη ιστοσελίδα.

Επιπλέον, η React Native σάς επιτρέπει να επωφεληθείτε από τα έξυπνα εργαλεία εντοπισμού σφαλμάτων και την αναφορά σφαλμάτων. Εάν αισθάνεστε άνετα με τα εργαλεία ανάπτυξης προγραμματιστών του Chrome ή του Safari, θα χαρούμε να γνωρίζουμε ότι μπορείτε να τα χρησιμοποιήσετε και για την ανάπτυξη κινητών τηλεφώνων. Ομοίως, μπορείτε να χρησιμοποιήσετε τον επεξεργαστή κειμένου που προτιμάτε για την επεξεργασία του JavaScript. Επιπλέον η Native δεν σας αναγκάζει να εργαστείτε στο XCode για να αναπτύξετε το iOS ή το Android Studio για ανάπτυξη τις Android εφαρμογή σας.

Εκτός από τις καθημερινές βελτιώσεις στην εμπειρία ανάπτυξης, το React Native έχει επίσης τη δυνατότητα να επηρεάσει θετικά τον κύκλο εκτόξευσης του προϊόντος σας. Για παράδειγμα, η Apple επιτρέπει αλλαγές βασισμένες στη JavaScript σε μια συμπεριφορά μιας εφαρμογής που θα φορτωθεί στον αέρα, χωρίς να απαιτείται επιπλέον κύκλος ανασκόπησης.

Όλα αυτά τα μικρά πλεονεκτήματα προστίθενται στην εξοικονόμηση χρόνου και ενέργειας για εσάς και τους συναδέλφους σας, επιτρέποντάς σας να εστιάσετε στα πιο ενδιαφέροντα μέρη της δουλειάς σας και να είστε πιο παραγωγικοί συνολικά.

b. Κίνδυνοι και μειονεκτήματα

Όπως και με οτιδήποτε, η χρήση του React Native δεν είναι χωρίς τα μειονεκτήματά του και το αν το React Native είναι μια καλή ιδέα για την ομάδα σας εξαρτάται πραγματικά από την προσωπική σας κατάσταση.

Ο μεγαλύτερος κίνδυνος είναι πιθανώς η ωριμότητα του React Native, καθώς το έργο παραμένει σχετικά νέο. Η υποστήριξη iOS κυκλοφόρησε τον Μάρτιο του 2015 και η υποστήριξη Android κυκλοφόρησε τον Σεπτέμβριο του 2015. Η τεκμηρίωση σίγουρα έχει περιθώρια βελτίωσης και συνεχίζει να εξελίσσεται. Ορισμένες λειτουργίες σε iOS και Android εξακολουθούν να μην υποστηρίζονται και η κοινότητα εξακολουθεί να ανακαλύπτει τις βέλτιστες πρακτικές.

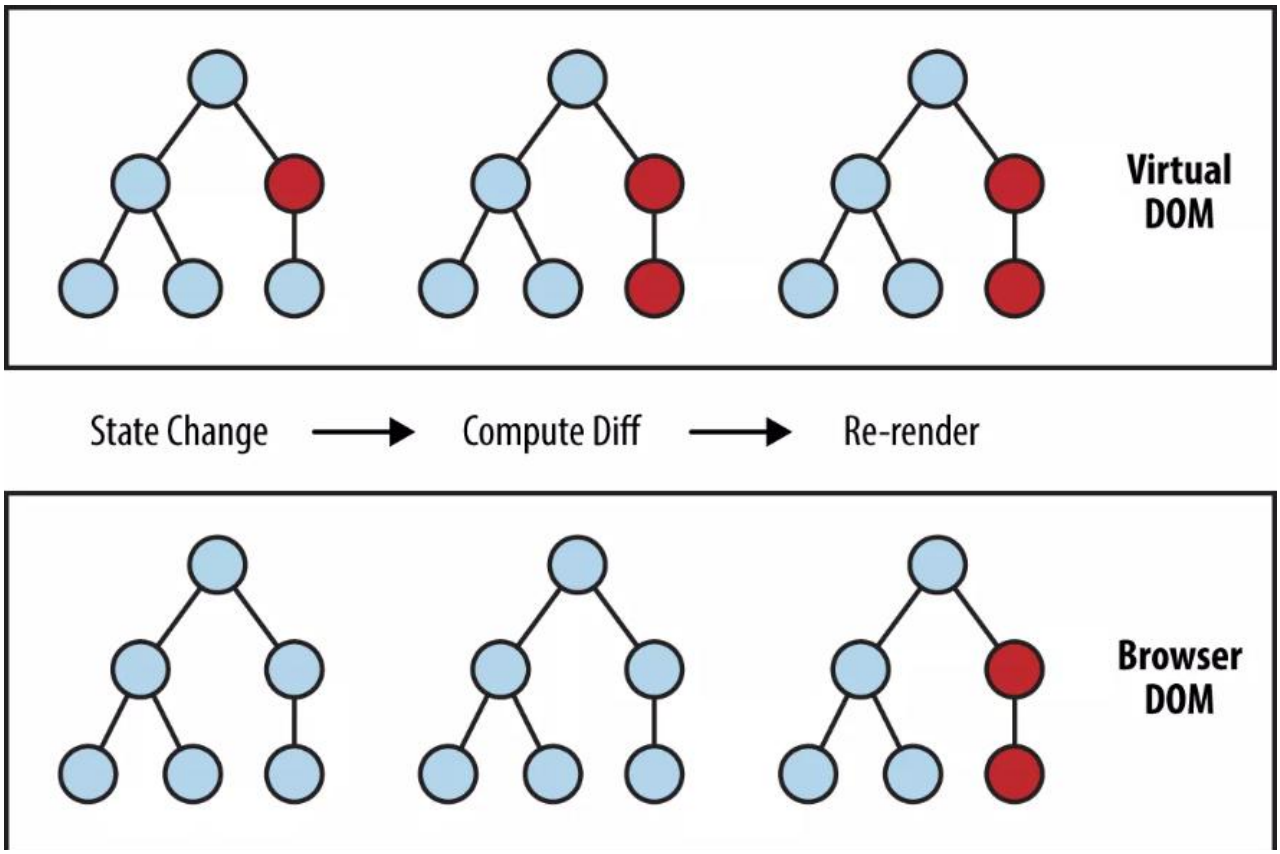
Επειδή η React Native εισάγει ένα άλλο στρώμα στο έργο σας, μπορεί επίσης να κάνει το debugging hairier, ειδικά στη διασταύρωση του React και της πλατφόρμας φιλοξενίας.

Αντίστοιχα, το Native είναι ακόμα νέο και ισχύουν οι συνήθεις προειδοποιήσεις που συνοδεύουν τη συνεργασία με τις νέες τεχνολογίες. Ωστόσο, γενικά, νομίζω ότι θα δείτε ότι τα οφέλη αντισταθμίζουν τους κινδύνους.

Πως λειτουργεί η React Native;

Η ιδέα να γράφετε κινητές εφαρμογές στο JavaScript είναι λίγο περίεργη. Πώς είναι δυνατή η χρήση τις React σε περιβάλλον κινητού τηλεφώνου; Για να κατανοήσουμε τα τεχνικά χαρακτηριστικά του τις React Native, πρώτα θα χρειαστεί να ανακαλέσουμε ένα από τα χαρακτηριστικά του React, το Virtual DOM.\

Στην React, το Εικονικό DOM λειτουργεί ως στρώμα ανάμεσα στην περιγραφή του προγραμματιστή για το πώς πρέπει να δούμε τα πράγματα και το έργο που έχει γίνει για την πραγματική απόδοση της αίτησής σας στη σελίδα. Για την απόδοση δια δραστικών διεπαφών χρήστη σε ένα πρόγραμμα περιήγησης, οι προγραμματιστές πρέπει να επεξεργαστούν το DOM του προγράμματος περιήγησης ή το μοντέλο αντικειμένου εγγράφου. Αυτό είναι ένα ακριβό βήμα, και οι υπερβολικές σημειώσεις στο DOM έχουν σημαντικό αντίκτυπο στην απόδοση. Αντί να κάνει απευθείας αλλαγές στη σελίδα, η React υπολογίζει τις απαραίτητες αλλαγές χρησιμοποιώντας μια έκδοση εντός της μνήμης DOM και επαναφέρει την ελάχιστη απαραίτητη ποσότητα. Το παρακάτω σχήμα δείχνει πώς λειτουργεί αυτό.



Εικόνα 42 (React State DOM)

Δημιουργία της πρώτης εφαρμογής σας

Η δημιουργία του development περιβάλλοντος θα σας δώσει τη δυνατότητα να ακολουθήσετε μαζί με τα παραδείγματα στο βιβλίο και θα σας επιτρέψει να γράψετε τις δικές σας εφαρμογές!

Οδηγίες για την εγκατάσταση της React Native μπορείτε να βρείτε στο επίσημο documentation της React Native. Ο επίσημος ιστότοπος θα είναι το πιο ενημερωμένο σημείο αναφοράς για συγκεκριμένα βήματα εγκατάστασης, αλλά θα τα αναλύσουμε και εδώ.

Θα χρειαστεί να χρησιμοποιήσετε το NodeJS, ένα κοινό package manager για το Microsoft Windows και στην περίπτωση που θέλουμε για Mac OS τότε θα χρησιμοποιήσουμε Homebrew, για να εγκαταστήσετε τα dependencies του React Native. Εδώ θα μάθουμε πως θα αναπτύξετε στο Windows OS, το οποίο σας επιτρέπει να γράφετε τόσο εφαρμογές iOS όσο και εφαρμογές Android.

Υπάρχουν δυο διαφορετικές κατηγορίες που μπορούμε να ξεκινήσουμε την React Native. Αν θέλουμε να κάνουμε build μια εφαρμογή και για Android και για iOS τότε η React μας συνιστά το expo package :

```
npm install -g expo-cli

Expo init TestPorject
```

```
Cd TestProject

Expo start
```

Σε αντίθετη περίπτωση γράφουμε καθαρό Native κώδικα και χρειαζόμαστε να κάνουμε build την Android εφαρμογή μας σε Android Studio η σε IntelliJ IDE και στην περίπτωση που θέλουμε και iOS τότε πρέπει να έχουμε Mac OS με XCode IDE ώστε να κάνουμε το build. Παρακάτω βλέπουμε τι κάνουμε εγκατάσταση για να κάνουμε Init το project μας :

```
Npm install -g react-native-cli

React-native init TestProject

Cd TestProject

React-native run-android
```

Components για mobile apps

Οι διεπαφές για κινητά βασίζονται σε διαφορετικά πρωτότυπα στοιχεία UI από τις ιστοσελίδες και επομένως πρέπει να χρησιμοποιήσουμε διαφορετικά στοιχεία.

Σε αυτήν την παράγραφο ξεκινάω με μια επισκόπηση των βασικών στοιχείων: <Text>, <View>, <Image>. Στη συνέχεια, θα συζητήσουμε πώς το touch και gesture επηρεάζουν τα React Native components και πώς χειρίζονται τα touch events. Στη συνέχεια, θα καλύψουμε στοιχεία υψηλότερου επιπέδου, όπως τα <NavigatorView>, <ListView> και <TabView>, τα οποία σας επιτρέπουν να συνδυάσετε views σε standard mobile interfaces.

HTML	React Native
div	View
img	Image
span, p	Text
ul/ol, li	ListView, child items

Εικόνα 43 (React Native Component Compare with HTML)

Το render κειμένου είναι μια παραπλανητική βασική λειτουργία. Σχεδόν οποιαδήποτε εφαρμογή θα χρειαστεί να κάνει κείμενο κάπου. Ωστόσο, το κείμενο στο πλαίσιο της ανάπτυξης

του React Native και του κινητού λειτουργεί διαφορετικά από την απόδοση κειμένου για τον Ιστό. Κάπως έτσι θα αποτυπώναμε κείμενο σε React Native:

```
<View>
  <Text>Text Message Here.<Text>
</View>
```

Η βασική χρήση του component <Image> είναι απλή. Απλά ορίστε την πηγή προέλευσης καπως ετσι:

```
<Image source={require('../images/nameofimage.jpg')} />

<Image source={{uri:'https://api.jewelshop.tk/nameofimage.jpg'}} />
```

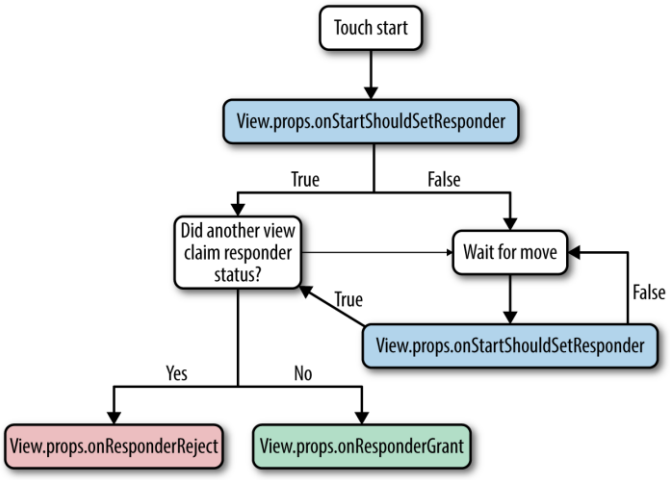
Το Gesture Responder σύστημα

Αντ' αυτού, η Native εκθέτει δύο APIs για προσαρμοσμένο χειρισμό αφής: Gesture Responder και Pan Responder. Το Gesture Responder είναι API κατωτέρου επιπέδου, ενώ το Pan Responder παρέχει ένα χρήσιμο abstraction. Αρχικά θα δούμε πώς λειτουργεί το σύστημα Gesture Responder, επειδή είναι η βάση για το API Pan Responder.

Το πάτημα στο κινητό είναι αρκετά περίπλοκο. Οι περισσότερες φορητές πλατφόρμες υποστηρίζουν multitouch, πράγμα που σημαίνει ότι μπορούν να υπάρχουν πολλαπλά σημεία επαφής ενεργά στην οθόνη ταυτόχρονα. (Δεν είναι απαραίτητα όλα αυτά τα δάχτυλα, σκεφτείτε τη δυσκολία, για παράδειγμα, να ανιχνεύσετε την παλάμη του χρήστη που στηρίζεται στη γωνία της οθόνης.) Επιπλέον, υπάρχει το θέμα της προβολής που θα πρέπει να χειρίζεται μια δεδομένη αφή. Αυτό το πρόβλημα είναι παρόμοιο με το πώς επεξεργάζονται τα συμβάντα του ποντικιού στον Ιστό, και η προεπιλεγμένη συμπεριφορά είναι επίσης παρόμοια: το παιδί από την κορυφή χειρίζεται το συμβάν αφής από προεπιλογή. Ωστόσο, με το σύστημα Gesture Responder της React Native, μπορούμε να παρακάμψουμε αυτή τη συμπεριφορά αν το επιλέξουμε.

Ο Touch Responder είναι η προβολή που χειρίζεται ένα δεδομένο συμβάν αφής. Στο προηγουμένη ενότητα, είδαμε το component <TouchableHighlight. Μπορούμε να αναγκάσουμε τα εξαρτήματά μας να γίνουν και οι αισθητήρες επαφής. Ο κύκλος ζωής με τον οποίο διαπραγματεύεται αυτή η διαδικασία είναι λίγο περίπλοκος. Μια άποψη που επιθυμεί να αποκτήσει την κατάσταση απόκρισης σε επαφή με το άγγιγμα θα πρέπει να εφαρμόζει τέσσερις σκηνές:

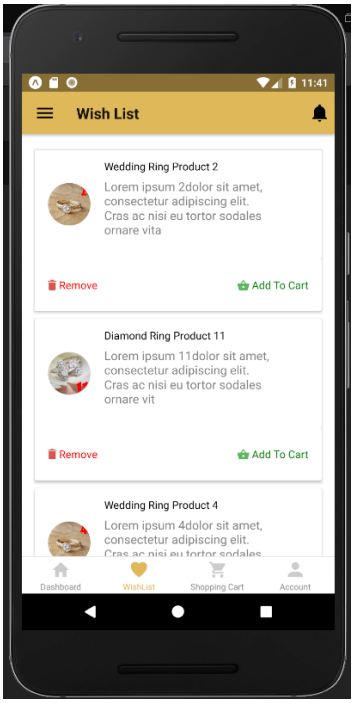
- `View.props.onStartShouldSetResponder`
- `View.props.onMoveShouldSetResponder`
- `View.props.onResponderGrant`
- `View.props.onResponderReject`



Εικόνα 44 (React Native Touchable System)

Working with Organizational Components

Σε αυτή την ενότητα, θα εξετάσουμε τα οργανωτικά components που μπορείτε να χρησιμοποιήσετε για να ελέγξετε τη γενική ροή μέσα στην εφαρμογή σας. Αυτό περιλαμβάνει τα <ListView>, <TabView> και <NavigatorView>, τα οποία όλα εφαρμόζουν μερικές από τις πιο κοινές μορφές αλληλεπίδρασης και πλοήγησης για κινητά. Μόλις προγραμματίσετε τη ροή πλοήγησης της εφαρμογής σας, θα διαπιστώσετε ότι αυτά τα στοιχεία είναι πολύ χρήσιμα για να κάνετε την εφαρμογή σας πραγματικότητα.



Ας ξεκινήσουμε χρησιμοποιώντας το στοιχείο <ListView>. Σε αυτήν την παράγραφο, πρόκειται να δημιουργήσουμε μια εφαρμογή που εμφανίζει τη λίστα αγαπημένων στο κατάστημα Jewel Shop και μας επιτρέπει να βλέπουμε δεδομένα για κάθε προϊόν, όπως φαίνεται στην εικόνα παρακάτω.

Οι λίστες είναι εξαιρετικά χρήσιμες για την ανάπτυξη κινητών και θα παρατηρήσετε ότι πολλές διεπαφές χρήστη για κινητά τις χαρακτηρίζουν ως κεντρικό στοιχείο. Ένα <ListView> είναι κυριολεκτικά απλώς μια λίστα με προβολές, οι οποίες μπορεί να έχουν dividers, headers, και footers.

Κεφάλαιο 5: Συστήματα Διαχείρισης Περιεχομένου (CMS)

5.1 Εισαγωγή

Ένα σύστημα διαχείρισης περιεχομένου (CMS) διαχειρίζεται τη δημιουργία και την τροποποίηση ψηφιακού περιεχομένου. Συνήθως υποστηρίζει πολλούς χρήστες σε ένα συνεργατικό περιβάλλον.

Οι λειτουργίες του CMS ποικίλλουν ευρέως. Χρησιμοποιούνται ευρέως είτε για διαχείριση περιεχομένου επιχειρήσεων είτε για διαχείριση περιεχομένου ιστού. Τα περισσότερα CMS περιλαμβάνουν την δημοσίευση στο Web, τη διαχείριση της μορφής, την επεξεργασία ιστορικού και τον έλεγχο έκδοσης, την ευρετηρίαση, την αναζήτηση και την ανάκτηση. Από τη φύση τους, τα συστήματα διαχείρισης περιεχομένου υποστηρίζουν το διαχωρισμό του περιεχομένου και της παρουσίασης.

Ένα σύστημα διαχείρισης περιεχομένου ιστού (WCM ή WCMS) είναι ένα CMS που έχει σχεδιαστεί για να υποστηρίζει τη διαχείριση του περιεχομένου των ιστοσελίδων. Τα πιο δημοφιλή CMS’s είναι επίσης WCM’s. Το περιεχόμενο Web περιλαμβάνει κείμενο και ενσωματωμένα γραφικά, φωτογραφίες, βίντεο, ήχο, χάρτες και κώδικα προγράμματος (όπως για εφαρμογές) που εμφανίζει περιεχόμενο ή αλληλοεπιδρά με τον χρήστη.

Ένα τέτοιο σύστημα διαχείρισης περιεχομένου (CMS) έχει συνήθως δύο βασικά στοιχεία. Μια εφαρμογή διαχείρισης περιεχομένου (CMA) είναι η διασύνδεση χρήστη front-end που επιτρέπει σε έναν χρήστη, ακόμη και με περιορισμένη τεχνογνωσία, να προσθέτει, να τροποποιεί και να καταργεί περιεχόμενο από έναν ιστότοπο χωρίς την παρέμβαση ενός webmaster. Μια εφαρμογή παράδοσης περιεχομένου (CDA) συγκεντρώνει αυτές τις πληροφορίες και ενημερώνει τον ιστότοπο. Τα ψηφιακά συστήματα διαχείρισης περιουσιακών στοιχείων είναι ένα άλλο είδος CMS. Διαχειρίζονται περιεχόμενο με σαφώς καθορισμένο συγγραφέα ή ιδιοκτησία, όπως έγγραφα, ταινίες, εικόνες, αριθμούς τηλεφώνου και επιστημονικά δεδομένα. Οι εταιρείες χρησιμοποιούν επίσης CMS για την αποθήκευση, τον έλεγχο, την αναθεώρηση και τη δημοσίευση τεκμηρίωσης.

Υπάρχουν επίσης συστήματα διαχείρισης περιεχομένου (CCMS), τα οποία είναι CMS που διαχειρίζονται περιεχόμενο σε ένα αρθρωτό επίπεδο παρά ως σελίδες ή άρθρα. Τα CCMS χρησιμοποιούνται συχνά στην τεχνική επικοινωνία όπου πολλές δημοσιεύσεις επαναχρησιμοποιούν το ίδιο περιεχόμενο.

5.2 Τι σημαίνει CMS;

Το CMS σημαίνει σύστημα διαχείρισης περιεχομένου.

Το CMS είναι λογισμικό υπολογιστή ή εφαρμογή που χρησιμοποιεί μια βάση δεδομένων για τη διαχείριση όλου του περιεχομένου και μπορεί να χρησιμοποιηθεί κατά την ανάπτυξη ενός ιστότοπου.

Επομένως, ένα CMS μπορεί να χρησιμοποιηθεί για την ενημέρωση του περιεχομένου ή / και της δομής του ιστότοπού σας.

Οι περισσότερες λύσεις CMS είναι ανοιχτού κώδικα, πράγμα που σημαίνει ότι είναι ελεύθεροι και προσβάσιμοι σε όλους. Ωστόσο, μερικοί ενδέχεται να περιέχουν επιπλέον πληρωμένες επιλογές, όπως πρότυπα και προσθήκες.

5.3 Γιατί χρειαζόμαστε τα CMS;

Όπως θα αναφέρω και παρακάτω τις δυνατότητες ενός CMS, σε αυτήν την παράγραφο θα πω ότι ένα CMS μας κάνει την ζωή πιο εύκολη και δεν χρειάζεται να έχουμε απαραίτητες γνώσεις ώστε να αναβαθμίζουμε μια ιστοσελίδα που έχουμε, αρκεί μόνο να βασίζεται όλο το περιεχόμενό της σε κάποιο από τα πιο γνωστά CMS. Τέλος θα αναφέρω ότι δεν χρειάζεται να ξέρουμε για το πως να κάνουμε την ιστοσελίδα μας πιο δημοφιλή (τα λεγόμενα SEO) και να έχει υψηλό ranking γιατί αυτό μας το προσφέρει το CMS μας.

5.4 Δυνατότητες των CMS;

- Μπορείτε να ενημερώσετε τον ιστότοπό σας με τους δικούς σας όρους

  Ας υποθέσουμε ότι θέλετε να προσθέσετε γρήγορα ένα συμβάν ή μια εικόνα σε μια από τις ιστοσελίδες σας. Εάν ο ιστότοπός σας ήταν hand coded από μια third party εταιρεία ανάπτυξης ιστοσελίδων, θα πρέπει να ζητήσετε από αυτούς να προσθέσουν το στοιχείο. Αντί να κρατάτε κάποιον από την εταιρεία ανάπτυξης ιστοσελίδων σας, ένα CMS κόβει τον μεσαίο άνθρωπο και σας δίνει τη δυνατότητα να ενημερώσετε και να επεξεργαστείτε το περιεχόμενο του ιστότοπού σας. Θα αποκτήσετε τη δυνατότητα να κάνετε τον ιστότοπό σας πιο δυναμικό, πιο επίκαιρο και πιο πολύτιμο για τους επισκέπτες σας.

- Δεν υπάρχει γνώση της HTML; Κανένα πρόβλημα.

  Το CMS θα έρθει με ένα πρόγραμμα επεξεργασίας κειμένου το οποίο ονομάζεται WYSIWYG (αυτό που βλέπετε είναι αυτό που λαμβάνετε). Αν γνωρίζετε πώς να δημιουργήσετε ένα απλό έγγραφο στο Microsoft Word, δεν θα έχετε κανένα πρόβλημα χρησιμοποιώντας το CMS σας για να δημιουργήσετε ιστοσελίδες, blog posts, articles και μηνύματα ηλεκτρονικού ταχυδρομείου.

- Ο επανασχεδιασμός του ιστότοπού σας δεν απαιτεί πλήρη αναμόρφωση του ιστότοπου.

  Με τους ιστότοπους που είναι ενσωματωμένοι σε ένα CMS, ο σχεδιασμός δημιουργείται ξεχωριστά από το περιεχόμενο. Έτσι, όταν είστε έτοιμοι να αλλάξετε την εμφάνιση του ιστότοπού σας, ο παλιός σχεδιασμός μπορεί να ανυψωθεί και να αντικατασταθεί με νέο σχεδιασμό χωρίς προβλήματα.

- Αποκτήστε πρόσβαση στις πιο πρόσφατες λειτουργίες του ιστού.

  Οποιοσδήποτε CMS αξίζει γιατί θα παρέχει στους χρήστες συχνές αναβαθμίσεις, καθώς και εργαλεία κατάρτισης σχετικά με τον τρόπο χρήσης νέας λειτουργικότητας. Πολλοί

ιδιοκτήτες CMS θα έρθουν επίσης με μια ομάδα υποστήριξης που οι χρήστες μπορούν να καλέσουν ή να στείλουν e-mail.

- Διαχειριστείτε ολόκληρη τη στρατηγική μάρκετινγκ Διαδικτύου πολύ εύκολα με την χρήση CMS.

  Το καλύτερο CMS θα περιλαμβάνει όλα όσα χρειάζεστε για να εφαρμόσετε μια ολοκληρωμένη στρατηγική στο διαδίκτυο. Τα περισσότερα CMS θα περιλαμβάνουν εργαλεία για SEO, μάρκετινγκ ηλεκτρονικού ταχυδρομείου, μάρκετινγκ κοινωνικών μέσων και blogging. Μπορείτε επίσης να χρησιμοποιήσετε ένα CMS για να δημιουργήσετε φόρμες εγγραφής γεγονότων, να συλλέξετε αμοιβές και δωρεές και να αποθηκεύσετε τις πληροφορίες των μελών.

5.5 Είδη και κατηγορίες CMS;

5.5.1 Ειδή CMS

1. Component Content Management System (CCMS)

   Ένα σύστημα διαχείρισης περιεχομένου συνιστωσών ή CCMS διαφέρει από ένα τυπικό CMS στο ότι οργανώνει περιεχόμενο σε κοκκώδες επίπεδο. Αντί να διαχειρίζεται περιεχόμενο σελίδας ανά σελίδα, παίρνει λέξεις, φράσεις, παραγράφους ή φωτογραφίες (επίσης γνωστά ως "components") και τις αποθηκεύει σε κεντρικό αποθετήριο.

   Σχεδιασμένο για μέγιστη επαναχρησιμοποίηση περιεχομένου, τα στοιχεία αποθηκεύονται μόνο μία φορά. Το CCMS λειτουργεί ως σταθερή πηγή έμπνευσης που δημοσιεύει περιεχόμενο σε πολλές πλατφόρμες, συμπεριλαμβανομένων των κινητών, των PDF και των εκτυπώσεων.

   Οφέλη από ένα CCMS

   a. **Reusability**: Η επαναχρησιμοποίηση του περιεχομένου εντός ενός CCMS εξοικονομεί χρόνο κατά τη διάρκεια της φάσης γραφής, επεξεργασίας και δημοσίευσης και μειώνει σημαντικά το κόστος μετάφρασης.
   b. **Traceability**: Το CCMS σάς δίνει τη δυνατότητα να παρακολουθείτε λεπτομερώς το περιεχόμενο. Μπορείτε να δείτε ποιος έκανε τι, πότε και πού.
   c. **Single Sourcing**: Με ένα CCMS μπορείτε να μεταφέρετε περιεχόμενο σε πολλά κανάλια, όπως εκτύπωση, κινητά, web, chatbots, embedded help και πολλά άλλα.
   d. **Enhance Team Collaboration**: Βελτιώστε τη ροή εργασίας για την ομάδα ανάπτυξης περιεχομένου σας, ειδικά για εκείνους που εργάζονται εξ αποστάσεως.

2. Document Management System (DMS)

   Το χαρτί είναι σχεδόν εξαφανισμένο. Η παρακολούθηση των αρχείων επιχειρήσεων σε χαρτί είναι παρελθόν. Ένα σύστημα διαχείρισης εγγράφων (DMS) προσφέρει μια λύση χωρίς χαρτιά για τη διαχείριση, την αποθήκευση και την παρακολούθηση εγγράφων σε

ένα σύννεφο. Παρέχει μια αυτοματοποιημένη λύση για τη μεταφόρτωση, επεξεργασία και κοινή χρήση επιχειρηματικών εγγράφων χωρίς την ταλαιπωρία της εκτύπωσης, αντιγραφής ή σάρωσης.

Οφέλη από ένα DMS

a. **Eco-friendly**: Οργανώστε το περιεχόμενο ψηφιακά και εξοικονομήστε χαρτί ενώ το κάνετε!
b. **Security**: Το DMS προσφέρει πολλά επίπεδα ασφάλειας για να εξασφαλίσει ότι το εμπιστευτικό περιεχόμενο παραμένει στα σωστά χέρια.
c. **Mobile Advantage**: Με ένα σύστημα διαχείρισης εγγράφων, μπορείτε να έχετε πρόσβαση και να επεξεργάζεστε έγγραφα από απόσταση.

3. Enterprise Content Management System (ECMS)

Ένα σύστημα διαχείρισης περιεχομένου επιχειρήσεων συλλέγει, οργανώνει και παραδίδει τεκμηρίωση ενός οργανισμού, εξασφαλίζοντας ότι οι κρίσιμες πληροφορίες παρέχονται στο σωστό κοινό (εργαζόμενοι, πελάτες, επιχειρηματίες, κ.λπ.)

Το ECM παρέχει σε όλα τα μέλη μιας οργάνωσης εύκολη πρόσβαση στο περιεχόμενο που χρειάζονται για να ολοκληρώσουν τα έργα και να λάβουν σημαντικές αποφάσεις. Επιπλέον, ένα ECM διαγράφει αρχεία μετά από μια συγκεκριμένη περίοδο διατήρησης, διασφαλίζοντας ότι δεν υπάρχει περιττό περιεχόμενο που καταλαμβάνει χώρο.

Οφέλη από ένα ECM

a. **Flexible**: Το ECM σας επιτρέπει να καταγράφετε οποιοδήποτε τύπο αρχείου από οποιαδήποτε τοποθεσία και να την έχετε επεξεργαστεί τότε αποθηκεύεται αυτόματα.
b. **Increases Efficiency**: Τώρα που επικρατούν τα documentations, μπορείτε να είστε πιο παραγωγικοί στην καθημερινή σας ζωή.
c. **Reduce Storage Costs**: Το ECM εξοικονομεί χρήματα αποθηκεύοντας μόνο τα απαραίτητα αρχεία και διαγράφοντας τα υπόλοιπα.

4. Web Content Management System (WCMS)

Ένα σύστημα διαχείρισης περιεχομένου ιστού επιτρέπει στους χρήστες να διαχειρίζονται ψηφιακά στοιχεία ενός ιστότοπου χωρίς προηγούμενη γνώση των γλωσσών σήμανσης ή του προγραμματισμού ιστού. Ένα WCMS παρέχει εργαλεία συνεργασίας, συγγραφής και διαχείρισης για τη διαχείριση του ψηφιακού περιεχομένου. Σε αντίθεση με άλλα CMS, τα οποία ασχολούνται με το περιεχόμενο που προορίζεται τόσο για τον ιστό όσο και για την εκτύπωση, ένα WCMS χειρίζεται αποκλειστικά περιεχόμενο ιστού.

Οφέλη από ένα WCMS

a. **Personalization**: Το WCMS επιτρέπει στους χρήστες να προσαρμόζουν μια ιστοσελίδα με εξατομικευμένο σχεδιασμό και περιεχόμενο.
b. **Automation**: Το WCMS εξοικονομεί χρόνο και βελτιώνει τη διαχείριση της ροής εργασιών με αυτόματη δημοσίευση περιεχομένου.

c. **Scalable**: Ένα κλιμακωτό σύστημα όπως το WCMS επιτρέπει στις εταιρείες να μεγαλώνουν εκθετικά χωρίς να ανησυχούν για να ξεπεράσουν τα όρια του ιστοτόπου τους.

5. Digital Asset Management System (DAM)

Με ένα ψηφιακό σύστημα διαχείρισης περιουσιακών στοιχείων, οι χρήστες μπορούν να αποθηκεύουν, να οργανώνουν και να μοιράζονται το ψηφιακό περιεχόμενο με ευκολία. Ένα DAM προσφέρει μια απλή, συγκεντρωτική βιβλιοθήκη όπου οι πελάτες, οι υπάλληλοι ή οι εργολάβοι έχουν πρόσβαση στο ψηφιακό περιεχόμενο. Αυτά τα στοιχεία περιλαμβάνουν ήχο, δημιουργικά αρχεία, βίντεο, έγγραφα και παρουσιάσεις. Ένα DAM είναι βασισμένο σε σύννεφο, έτσι ώστε οι users να μπορούν να έχουν πρόσβαση σε περιεχόμενο από οπουδήποτε.

Οφέλη από ένα DAM

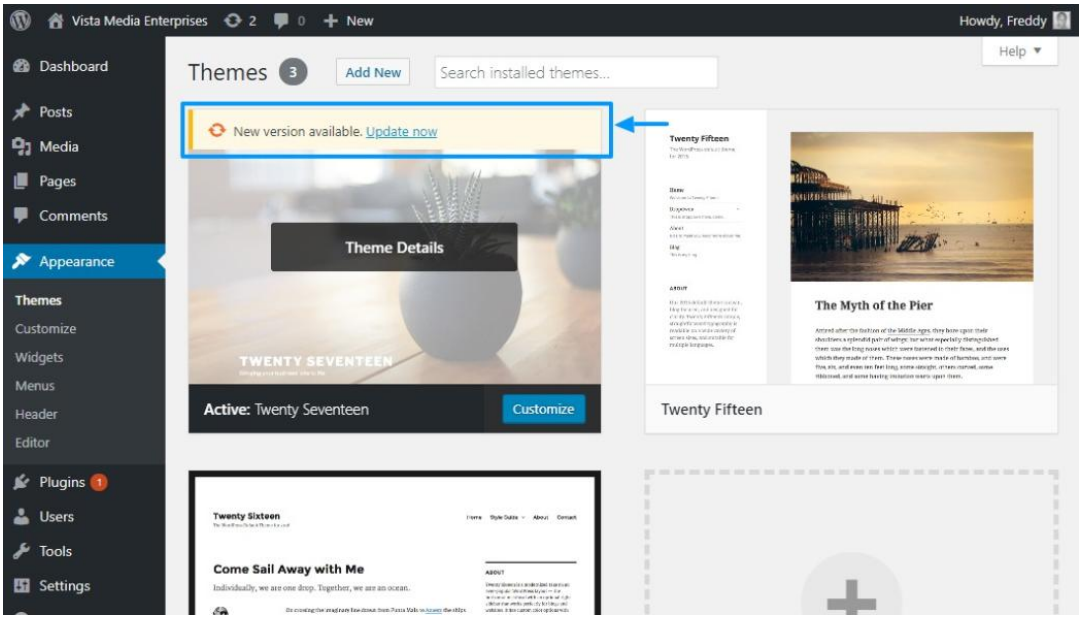
a. **Centralized Repository**: Το περιεχόμενο είναι ασφαλές και σε ένα μέρος.
b. **Effective Brand Management**: Ένα DAM σας επιτρέπει να διαχειριστείτε μια επώνυμη δικτυακή πύλη για τους χρήστες να έχουν πρόσβαση σε σημαντικά αρχεία.
c. **Digital Publishing**: Με το DAM, μπορείτε να προωθήσετε το ψηφιακό περιεχόμενο σε υπηρεσίες διανομής τρίτων, στα κανάλια κοινωνικών μέσων και πολλά άλλα.

Πριν επιλέξετε ένα CMS, είναι σημαντικό να εξερευνήσετε όλους τους τύπους συστημάτων διαχείρισης περιεχομένου. Ορισμένα συστήματα μπορεί να είναι πιο κατάλληλα για την επιχείρησή σας. Για παράδειγμα, αν δημοσιεύετε πολλές εκδόσεις προϊόντων σε ένα χρόνο σε πολλές γλώσσες, το καλύτερο στοίχημά σας είναι σίγουρα ένα CCMS. Εάν αναζητάτε περισσότερη ευελιξία για τη δημοσίευση σε έναν ιστότοπο, ένα σύστημα διαχείρισης περιεχομένου ιστού μπορεί να ταιριάζει καλύτερα στον λογαριασμό.

5.5.2 Κατηγορίες CMS

Παρακάτω θα αναλύσω τρεις από τις πιο μεγάλες κατηγορίες CMS αλλά σε αυτήν την παράγραφο θα αναφέρω μερικά CMS ονομαστικά και κάποια e-commerce CMS τα οποία κατατάσσονται στα Content Management Systems. Κάποια από αυτά είναι το WordPress, Joomla, Drupal, Neos CMS, Squarespace, Magnolia, Weebly, Wix, Medium, PrestaShop, Magento και το OpenCart.

1. WordPress



Τι έχει μείνει να πούμε για το WordPress που δεν έχει ήδη ειπωθεί στο παρελθόν; Η πλατφόρμα blogging PHP είναι μακράν το πιο δημοφιλές CMS για το blogging και ίσως το πιο δημοφιλές CMS συνολικά. Είναι μια μεγάλη πλατφόρμα για αρχάριους, χάρη στο εξαιρετικό documentation και στο super-quick installation wizard. Πέντε λεπτά σε ένα τρέχον CMS είναι αρκετά καλό. Για να μην αναφέρουμε το γεγονός ότι οι νεότερες εκδόσεις ενημερώνουν αυτόματα τον πυρήνα και τις προσθήκες μέσα από το backend, χωρίς να χρειάζεται να κάνουν λήψη ενός μόνο αρχείου.

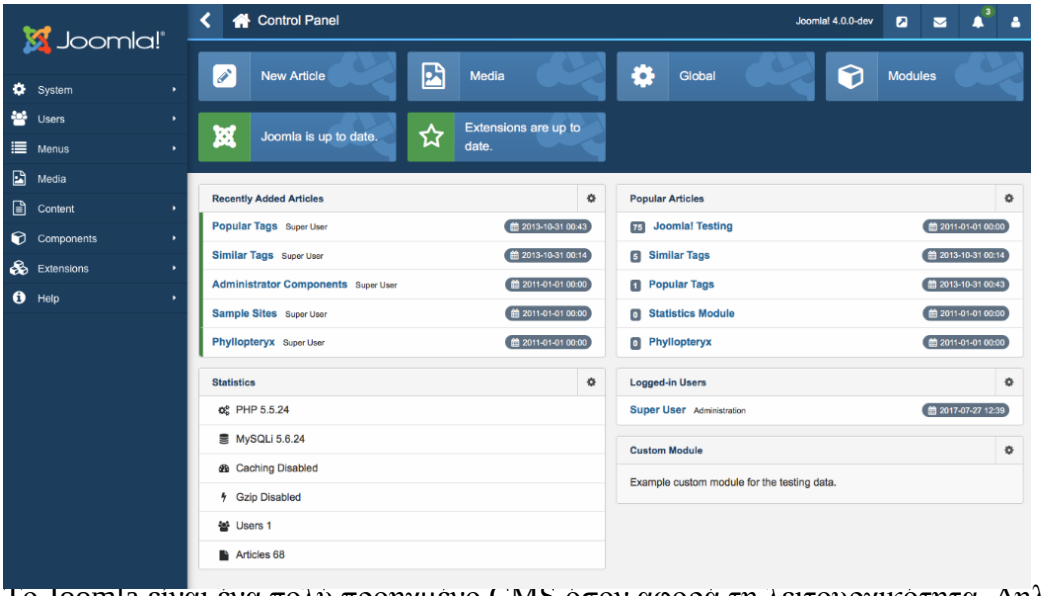
Για τους χρήστες που δεν είναι γνώστες με τη γλώσσα HTML ή άλλη γλώσσα σήμανσης, ένας επεξεργαστής WYSIWYG παρέχεται κατευθείαν από την WordPress. Το backend layout είναι απλοποιημένο και διαισθητικό και ένας νέος χρήστης θα πρέπει να μπορεί εύκολα να βρει το δρόμο του γύρω από το τμήμα διαχείρισης. Το WordPress έρχεται επίσης με ενσωματωμένη υποστήριξη μεταφόρτωσης εικόνων και πολυμέσων.

Για τους προγραμματιστές, τα themes είναι αρκετά εύκολα και απλά, καθώς και το API Plugin.

Η Κοινότητα WordPress είναι μια πιστή και ζωντανή δέσμη. Το WordPress έχει πιθανώς την ευρύτερη βάση plugins και themes για να διαλέξετε. Έχουμε διαθέσει χιλιάδες επαγγελματικά WordPress Themes και WordPress Plugins για πώληση στο Envato Market, με μια πλήρη σουίτα στυλ και επιλογές για να διαλέξετε.

Ένα μεγάλο μέρος για την κοινότητα του WordPress είναι το πόσο μπορεί να σας προσφέρει την βοήθειά του και τα online documentations που μπορείτε να βρείτε σχεδόν σε κάθε πτυχή της προσαρμογής του WordPress.

2. Joomla

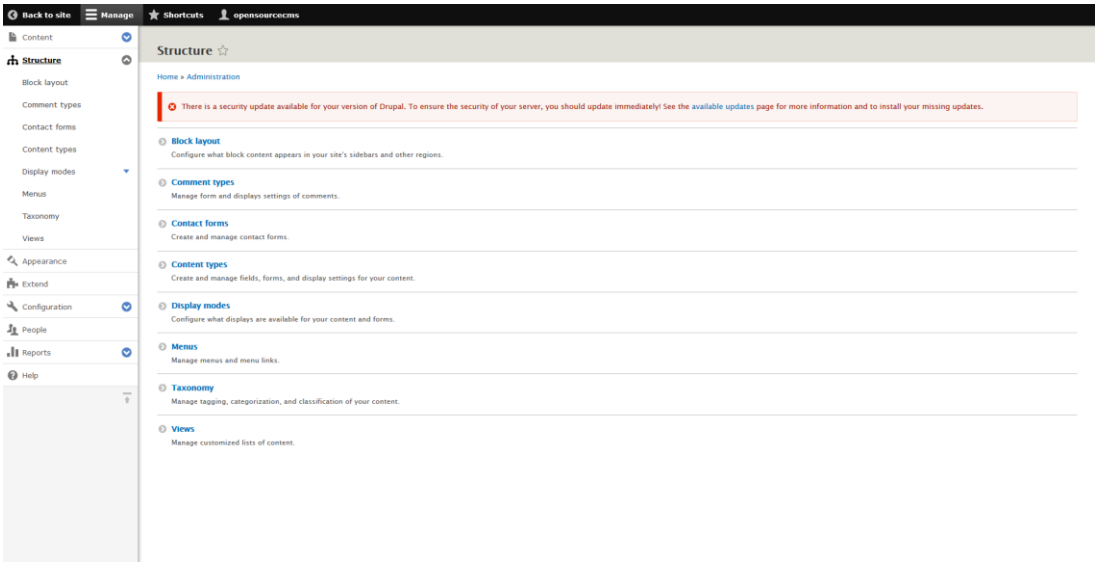


Το Joomla είναι ένα πολύ προηγμένο CMS όσον αφορά τη λειτουργικότητα. Δηλαδή, ξεκινώντας με το Joomla είναι αρκετά εύκολο, χάρη στο wizard installer του Joomla. Το πρόγραμμα εγκατάστασης του Joomla προορίζεται να λειτουργήσει σε κοινά κοινόχρηστα πακέτα φιλοξενίας και είναι πολύ απλό, λαμβάνοντας υπόψη το πόσο διαμορφωμένο είναι το λογισμικό.

Το Joomla είναι πολύ παρόμοιο με το Drupal στο ότι είναι ένα ολοκληρωμένο CMS και μπορεί να είναι λίγο πολύ για ένα απλό site portfolio. Έρχεται με μια ελκυστική διεπαφή διαχείρισης, με πλήρη διαισθητικά μενού και άλλα χαρακτηριστικά. Το CMS έχει επίσης μεγάλη υποστήριξη για πρωτόκολλα ελέγχου πρόσβασης όπως το LDAP, το OpenID ή ακόμα και το Gmail.com.

Η τοποθεσία του Joomla φιλοξενεί περισσότερες από 3.200 επεκτάσεις, ώστε να γνωρίζετε ότι η κοινότητα προγραμματιστών πίσω από το δημοφιλές CMS είναι ζωντανή. Όπως το WordPress, μπορείτε να προσθέσετε σχεδόν οποιαδήποτε λειτουργικότητα που χρειάζεστε με μια επέκταση. Ωστόσο, η κοινότητα θεμάτων και επεκτάσεων του Joomla εξαρτάται περισσότερο από τους πόρους που πληρώνονται, οπότε αν ψάχνετε για προσαρμογές, να είστε έτοιμοι να βγάλετε το πορτοφόλι σας. Μπορείτε επίσης να αρπάξετε τα plugins του Joomla ή να νοικιάσετε προγραμματιστές του Joomla για να σας βοηθήσουμε να ρυθμίσετε το κατάστημά σας σωστά.

3. Drupal



Το Drupal είναι ένα άλλο CMS που έχει μια πολύ μεγάλη, ενεργή κοινότητα. Αντί να επικεντρώνεται στο blogging ως πλατφόρμα, το Drupal είναι περισσότερο ένα καθαρό CMS. Μια απλή εγκατάσταση έρχεται με έναν τόνο προαιρετικών ενοτήτων που μπορούν να προσθέσουν πολλά ενδιαφέροντα χαρακτηριστικά όπως φόρουμ, blog χρηστών, OpenID, προφίλ και πολλά άλλα. Είναι ασήμαντο να δημιουργήσετε έναν ιστότοπο με κοινωνικά χαρακτηριστικά με μια απλή εγκατάσταση του Drupal. Στην πραγματικότητα, με μερικά third party modules μπορείτε να δημιουργήσετε μερικά ενδιαφέροντα clone sites με μικρή προσπάθεια.

Ένα από τα πιο δημοφιλή χαρακτηριστικά του Drupal είναι το module Taxonomy, ένα χαρακτηριστικό που επιτρέπει πολλαπλά επίπεδα και τύπους κατηγοριών για τύπους περιεχομένου. Και μπορείτε να βρείτε πολλά επαγγελματικά θέματα Drupal, τα οποία είναι έτοιμα να προσαρμοστούν και να εργαστούν με το περιβάλλον του Drupal. Μπορείτε επίσης να αρπάξετε το Drupal Plugins.

Το Drupal έχει επίσης μια πολύ ενεργή κοινότητα που τον τροφοδοτεί και έχει εξαιρετική υποστήριξη για plugins και άλλα γενικά ερωτήματα.

5.6 CMS κλειστού κώδικα vs ανοιχτού κώδικα.

5.6.1 Γενικά

Όταν ένας οργανισμός αποφασίζει ότι χρειάζεται ένα νέο δικτυακό τόπο, υπάρχουν πολλές αποφάσεις που πρέπει να ληφθούν, από το branding μέχρι την τεχνολογία που χρησιμοποιείται. Μία από τις μεγαλύτερες αποφάσεις είναι ποιο σύστημα διαχείρισης περιεχομένου (CMS) να επιλέξει και, αποφασιστικά, το αν θα επιλέξει ένα CMS ανοιχτού κώδικα ή μια ιδιόκτητη λύση.

Ένα CMS ανοιχτού κώδικα είναι ένα σύστημα διαχείρισης περιεχομένου που διατηρείται από την κοινότητα των προγραμματιστών, και όχι από μια εταιρεία που αναπτύσσεται και ανήκει σε μία μόνο εταιρεία. Ο πηγαίος κώδικας είναι διαθέσιμος σε όλους και ο καθένας μπορεί να το τροποποιήσει και να δημιουργήσει τη δική του λειτουργικότητα. Αποτελεσματικά, ο καθένας

μπορεί να κάνει αυτό που τους αρέσει. Ένα ιδιόκτητο CMS, από την άλλη πλευρά, χτίζεται και συντηρείται από μια συγκεκριμένη εταιρεία. Είναι κύριοι του CMS και άλλοι πληρώνουν license fee (συχνά σε μηνιαία βάση) για να μπορέσουν να το χρησιμοποιήσουν. Στις περισσότερες περιπτώσεις, η εταιρεία που κατέχει ένα ιδιόκτητο CMS θα δημιουργήσει τον ιστότοπο και θα το διατηρήσει.

Τόσο τα συστήματα διαχείρισης ανοικτού κώδικα όσο και τα ιδιόκτητα συστήματα διαχείρισης περιεχομένου έχουν τα οφέλη και τα μειονεκτήματά τους, ορισμένα από αυτά περιγράφονται στο υπόλοιπο αυτού του κεφαλαίου.

5.6.2 Ανοιχτού Κώδικα CMS

5.6.2.1 Τα οφέλη

Ένα από τα σημαντικότερα πλεονεκτήματα ενός συστήματος διαχείρισης περιεχομένου ανοικτού κώδικα είναι ότι γενικά έχουν μια μεγάλη βάση προγραμματιστών. Υπάρχει μια σειρά από οφέλη που προέρχονται από αυτό. Οι λύσεις ανοιχτού κώδικα, όπως το Drupal, το WordPress, το Joomla και το Umbraco, αναπτύσσονται συνεχώς. Πάντα εργάζονται και ενημερώνονται με κώδικα και οι ενότητες βελτιώνονται συνεχώς. Ο αριθμός των συνεισφερόντων μπορεί να είναι τεράστιος, για παράδειγμα το Drupal έχει “630.000 + χρήστες και προγραμματιστές” σύμφωνα με την ιστοσελίδα τους, drupal.org. Αυτό σημαίνει ότι ποτέ δεν σταματάει και πάντα εξελίσσεται σε ένα καλύτερο προϊόν, συνάδοντας με τις ανάγκες των χρηστών. Από πολλές απόψεις, είναι συνήθως ένα βήμα μπροστά από τις περισσότερες ιδιόκτητες λύσεις.

Ένας βασικός λόγος για την επιλογή πολλών οργανισμών στην επιλογή ενός CMS ανοιχτού κώδικα είναι ότι είναι πολύ πιο εύκολο για αυτούς να αλλάξουν πρακτορεία, αν προκύψει ανάγκη. Εάν ο ιστότοπός σας είναι χτισμένος σε CMS ανοιχτού κώδικα, τότε θα υπάρχουν αρκετοί άλλοι οργανισμοί ιστού που θα μπορούσαν να αναλάβουν τον ιστότοπό σας αν αποφασίσετε ότι δεν είστε ικανοποιημένοι με τον οργανισμό με τον οποίο συνεργάζεστε. Αυτό δεν συμβαίνει με ένα ιδιόκτητο CMS. Στις περισσότερες περιπτώσεις, θα χρειαστεί να το ξαναχτίσουν. Με ένα CMS ανοιχτού κώδικα, διαθέτετε τον κώδικα, ώστε να μπορεί να χρησιμοποιηθεί από κάποιον άλλον οργανισμός για να τον διατηρήσει και να τον αναπτύξει περαιτέρω, εάν είναι απαραίτητο.

Ένα άλλο μεγάλο πλεονέκτημα ενός CMS ανοιχτού κώδικα είναι ότι δεν υπάρχει το τρέχον κόστος που μπορεί να συσχετιστεί με ένα ιδιόκτητο σύστημα. Μόλις δημιουργηθεί ένας ιστότοπος, δεν υπάρχει license fee που πρέπει να καταβάλλεται σε συνεχή βάση.

5.6.2.2 Τα μειονεκτήματα

Η εμπειρία των τελικών χρηστών σε σχέση με τους προγραμματιστές είναι κάτι που δόθηκε ως μειονέκτημα των συστημάτων διαχείρισης περιεχομένου ανοιχτού κώδικα από ορισμένους. Δεν υπάρχει αμφιβολία ότι πολλά CMS’s είναι πιο κατάλληλα για τους προγραμματιστές από τους τελικούς χρήστες (τους πελάτες τους), με τους οποίους μπορεί μερικές φορές να φαίνονται πολύπλοκα.

Ένα άλλο πιθανό μειονέκτημα είναι ότι μπορούν να είναι ευαίσθητα σε θέματα ασφάλειας. Είναι πιο ευάλωτα στην επίθεση καθώς ο καθένας μπορεί να δει τον κώδικα, καθιστώντας έτσι ευκολότερο για τους hackers να το εκμεταλλευτούν. Οι παραβιάσεις ασφαλείας είναι επίσης πιθανότερο να δημοσιευθούν. Λόγω της δημοτικότητας ενός συγκεκριμένου CMS, εάν υπάρχει παραβίαση της ασφάλειας, είναι πιο πιθανό να συζητηθεί και να γραφτεί, κάτι που μπορεί να δει κάποιος ότι είναι μια ευκαιρία εκμετάλλευσης. Το προβάδισμα είναι ότι αυτό σημαίνει επίσης ότι οι άνθρωποι είναι πιο πιθανό να ενημερωθούν για τυχόν ζητήματα, ώστε να μπορούν να προστατεύσουν από αυτά, και λόγω του αριθμού των συνεισφερόντων, είναι πιο πιθανό να υπάρξει λύση και να μεταδοθεί στο σύνολο της κοινότητας.

5.6.3 Κλειστού κώδικα CMS (Ιδιόκτητα)

5.6.3.1 Τα οφέλη

Ένα από τα πλεονεκτήματα που μπορεί να έχει ένα ιδιόκτητο CMS σε μια λύση ανοιχτού κώδικα είναι ότι υπάρχουν οργανισμοί που έχουν δημιουργήσει συστήματα διαχείρισης περιεχομένου για να ταιριάζουν σε μια συγκεκριμένη αγορά. Για παράδειγμα, έχουν δημιουργηθεί αρκετά CMS για τη βιομηχανία ακίνητης περιουσίας, για όσους αγαπούν τους κτηματομεσίτες και τους πράκτορες ενοικίασης. Καθώς δημιουργούνται για έναν συγκεκριμένο τύπο ιστότοπου, μπορούν να αναπτυχθούν ώστε να ταιριάζουν με τη λειτουργικότητα που απαιτούν αυτές οι τοποθεσίες. Για να χρησιμοποιήσει ξανά το χώρο της ακίνητης περιουσίας, ένα CMS για αυτόν τον τομέα θα έχει λειτουργίες που επιτρέπουν στους οργανισμούς να είναι σε θέση να φορτώνουν εύκολα νέες ιδιότητες. Από την πλευρά του πελάτη, έχουν ένα προϊόν που είναι προσαρμοσμένο σε αυτά, το οποίο δεν απαιτεί από την αντιπροσωπεία τους να κατασκευάσουν αυτή τη λειτουργία από την αρχή, αποδεικνύοντας έτσι πιο αποδοτικό σε ορισμένες περιπτώσεις.

Ένα ιδιόκτητο CMS μπορεί να είναι πιο επικεντρωμένο στον χρήστη από ό, τι στον σχεδιαστή. Τείνουν να είναι λιγότερο προσαρμόσιμα και επομένως απλούστερα. Αυτό μπορεί να είναι επωφελές για οργανισμούς που δεν απαιτούν υπερβολικά περίπλοκη λειτουργικότητα, αλλά επιθυμούν να μπορούν να προσθέτουν και να επεξεργάζονται το ίδιο το περιεχόμενο.

Εάν επιλέγετε ένα ιδιόκτητο CMS που ανήκει στον οργανισμό που δημιούργησε και συντηρεί τον ιστότοπό σας, τότε αυτό έχει το πλεονέκτημα ότι γνωρίζετε ακριβώς ποιος θα είναι υπεύθυνος αν υπάρχει κάποιο πρόβλημα. Δεν είναι μόνο υπεύθυνοι για τον ιστότοπό σας, αλλά και για ολόκληρο το CMS. Ως εκ τούτου, είναι ευθύνη τους να επιλύσουν τυχόν ζητήματα και γνωρίζετε ότι μπορείτε να τους κρατήσετε υπεύθυνους. Μπορείτε επίσης να πληρώνετε για την άδεια CMS σε μηνιαία βάση, πράγμα που σημαίνει ότι μπορείτε να αρνηθείτε να πληρώσετε αν κάτι πάει στραβά, κάτι που δεν θα θέλουν να κάνετε. Αυτό μπορεί να επιταχύνει τη διαδικασία των απαιτούμενων διορθώσεων που εκτελούνται.

5.6.3.2 Τα μειονεκτήματα

Το μεγαλύτερο μειονέκτημα σχετικά με ένα ιδιόκτητο CMS είναι ότι μπορείτε να κολλήσετε με ένα web agency. Αν κατέχουν το CMS, είναι πολύ δύσκολο, και ενδεχομένως ακριβό, να πάει αλλού. Δεν θα μπορείτε να μεταβείτε απλά σε άλλη υπηρεσία για να αναλάβετε την παρούσα ιστοσελίδα σας. Θα χρειαστεί να ξαναχτίσετε ολόκληρο τον ιστότοπό σας σε διαφορετικό CMS. Εάν η προηγούμενη εταιρεία σας διέθετε τα πνευματικά δικαιώματα για το σχέδιο, θα απαιτούσε

επίσης έναν πλήρη επανασχεδιασμό. Αυτό μπορεί να είναι ένας τεράστιος πονοκέφαλος, που συχνά οδηγεί σε οργανώσεις που παραμένουν σε μια υπηρεσία με την οποία δεν είναι ικανοποιημένοι.

Στις περισσότερες περιπτώσεις, υπάρχουν περισσότεροι περιορισμοί με ένα ιδιόκτητο CMS. Μερικές φορές αγοράζετε ένα πολύ συγκεκριμένο προϊόν, χωρίς μεγάλη ελευθερία και επιλογή χαρακτηριστικών. Τα κορυφαία CMS ανοιχτού κώδικα προσφέρουν ατελείωτες δυνατότητες όσον αφορά το τι μπορείτε να κάνετε με τον ιστότοπό σας, κάτι που λείπει με τις περισσότερες ιδιόκτητες λύσεις.

Ένα άλλο ζήτημα είναι αυτό της ιδιοκτησίας. Εάν ο ιστότοπός σας είναι ενσωματωμένος σε ένα CMS ανοιχτού κώδικα, τότε είστε ο κάτοχος του κώδικα. Όταν είναι ενσωματωμένο σε ένα ιδιόκτητο σύστημα ανήκει στον οργανισμό που κατέχει τα πνευματικά δικαιώματα του CMS.

5.6.4 Πρέπει να επιλέξετε ένα Open Source ή ένα ιδιόκτητο CMS;

Υπάρχουν πολλοί παράγοντες που πρέπει να λάβετε υπόψη όταν αποφασίζετε αν χρειάζεστε ένα CMS ανοιχτού κώδικα ή ιδιοκτησίας. Εάν έχετε συγκεκριμένες ανάγκες που καλύπτονται από ένα συγκεκριμένο ιδιόκτητο CMS, τότε μπορεί να θέλετε να εξετάσετε τα οφέλη. Μπορεί να ανταποκρίνεται στις ανάγκες σας και να είναι αποδοτική από πλευράς κόστους σε σύγκριση με την ίδια λειτουργικότητα ενσωματωμένη σε μια λύση ανοιχτού κώδικα.

Στις περισσότερες περιπτώσεις, ωστόσο, συνιστούμε μια λύση ανοιχτού κώδικα. Είναι πολύ πιο προσαρμόσιμα και προσφέρουν μεγαλύτερη ελευθερία στους πελάτες. Η ευελιξία που προσφέρει επιτρέπει στους πελάτες να επιτύχουν ακριβώς αυτό που αναζητούν. Δεν περιορίζονται από τους περιορισμούς που μπορεί να έχει ένα ιδιόκτητο σύστημα. Το όφελος από την κατοχή του δικού σας κώδικα σημαίνει ότι αν επιλέξετε να το κάνετε στο μέλλον, μπορείτε να αλλάξετε πρακτορεία αν δεν είστε ευχαριστημένοι από το υπάρχον.

Ενώ μερικά ιδιόκτητα CMS’s προσφέρουν αξία, στις περισσότερες περιπτώσεις πιστεύουμε ότι μια λύση ανοιχτού κώδικα προσφέρει την καλύτερη αξία στους πελάτες.

Κεφάλαιο 6: Ανάπτυξη Custom CMS με την χρήση του Open Source πακέτου Voyager

6.1 Τι είναι το Voyager;

6.1.1 Γενικά

- Μια διεπαφή διαχειριστή για την εφαρμογή Laravel.
- Ένας εύκολος τρόπος για να προσθέσετε / επεξεργαστείτε / διαγράψετε δεδομένα για την εφαρμογή σας.
- Ένας menu builder(δημιουργία μενού στο Voyager για την εφαρμογή σας).
- Ένας media manager για τα αρχεία σας.
- Γεννήτρια CRUD / BREAD (μάθετε περισσότερα για το BREAD παρακάτω).

Το Voyager είναι απλώς ένας διαχειριστής της εφαρμογής Laravel. Ό,τι θέλετε να κάνει η εφαρμογή σας στο frontend εξαρτάται από εσάς. Έχετε τον έλεγχο της εφαρμογής σας και μπορείτε να χρησιμοποιήσετε το Voyager για να διευκολύνετε τη ζωή σας με την προσθήκη δεδομένων, την επεξεργασία χρηστών, τη δημιουργία μενού και πολλές άλλες διοικητικές εργασίες.

6.1.2 Τι δεν είναι το Voyager;

- Ένα σύστημα διαχείρισης περιεχομένου (CMS).
- Μια πλατφόρμα blogging.
- WordPress.

Το Voyager δεν είναι ένα CMS ή μια πλατφόρμα blogging. Αλλά μπορεί να χρησιμοποιηθεί για να δημιουργήσει ένα CMS ή μια πλατφόρμα blogging, αλλά έξω από το κουτί δεν είναι ούτε από αυτά. Όπως αναφέρθηκα πιο πάνω, έχετε πλήρη έλεγχο του τι θα κάνει η αίτησή σας και πώς θα λειτουργήσει.

Με το Laravel & Voyager μπορείτε να δημιουργήσετε κάθε είδους εφαρμογή που επιθυμεί η καρδιά σας. Το Laravel & Voyager είναι απλά εργαλεία για να διευκολύνεται την κατασκευή.

6.1.3 Τι θα μετατρέψουμε το Voyager;

- Σε ένα σύστημα διαχείρισης περιεχομένου (CMS).
- Σε μια πλατφόρμα blogging system.
- Σε μια πλατφόρμα που θα αλλάζετε themes.

Με την βοήθεια του ανοιχτού λογισμικού Voyager και Laravel θα μετατρέψω τον απλό CRUD generator της Voyager σε ένα σύστημα διαχείρισης περιεχομένου να δρα σαν να έχουμε ένα WordPress CMS και αναλυτικά να μπορεί να έχει το σύστημα διαχείριση για τους χρήστες με πολλαπλά δικαιώματα, να μπορεί επίσης να αλλάζει θέματα για το frontend ιστότοπο, να μπορεί να αλλάζει βασικά στοιχεία επικοινωνίας για την ιστοσελίδα, κάποια logo και για το Admin Panel και για το Frontend. Επίσης να μπορεί να προσθέτει Google Analytics ώστε να βλέπει κατευθείαν την κίνηση από το διαχειριστικό παράθυρο που βρίσκετε. Τέλος να μπορεί να ανανεώνει το Blog, τα Posts και τις κατηγορίες των post.

6.2 Γιατί επιλέξαμε Voyager;

Επέλεξα το voyager γιατί δεν είναι ένα CMS αλλά ένα πακέτο CRUD (το οποίο σημαίνει create,read,update,delete) το οποίο και θα μετατρέψω να διαχειρίζεται πράγματα δυναμικά σαν να είναι CMS. Επέλεξα Voyager πρώτον γιατί είναι ανοιχτού κώδικα και δεύτερον γιατί ήθελα να φτιάξω κάτι από την αρχή και να έχει πολλές δυνατότητες ώστε να το ονομάσω CMS. Τα βασικά προνόμια αυτού του πακέτου είναι το Media Manager, το Menu Builder, το Database Manager, το Bread/CRUD Builder και το User Access (Permissions και Roles).

6.3 Εγκατάσταση του Voyager

Το Voyager είναι εξαιρετικά εύκολο στην εγκατάσταση. Αφού δημιουργήσετε τη νέα εφαρμογή Laravel, μπορείτε να συμπεριλάβετε το πακέτο Voyager με την ακόλουθη εντολή:

```
composer require tcg/voyager
```

Εικόνα 45 (Voyager Download)

Στη συνέχεια, βεβαιωθείτε ότι έχετε δημιουργήσει μια νέα βάση δεδομένων και προσθέσετε τα credential της βάσης δεδομένων στο αρχείο .env. Επίσης, θα πρέπει να προσθέσετε τη διεύθυνση URL της εφαρμογής σας στη μεταβλητή APP_URL:

```
APP_URL=http://localhost
DB_HOST=localhost
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

Εικόνα 46 (Voyager Database Credential)

Τέλος, μπορούμε να εγκαταστήσουμε το Voyager. Μπορείτε να επιλέξετε να εγκαταστήσετε το Voyager με εικονικά δεδομένα ή χωρίς τα εικονικά δεδομένα. Τα εικονικά δεδομένα θα περιλαμβάνουν 1 λογαριασμό διαχειριστή (αν δεν υπάρχουν ήδη χρήστες), 1 σελίδα demo, 4 demo posts, 2 κατηγορίες και 7 ρυθμίσεις.

Για να εγκαταστήσετε το Voyager χωρίς εικονικά δεδομένα, απλώς εκτελέστε:

```
php artisan voyager:install
```

Εικόνα 47 (Voyager Install Command)

Εάν προτιμάτε να το εγκαταστήσετε με τα εικονικά δεδομένα, εκτελέστε την ακόλουθη εντολή:

```
php artisan voyager:install --with-dummy
```

Εικόνα 48 (Voyager Install with Dummies)

Ξεκινήστε ένα διακομιστή τοπικής ανάπτυξης με php artisan serve και επισκεφτείτε τη διεύθυνση http://localhost: 8000/admin στο πρόγραμμα περιήγησής σας.

Αν εγκαταστήσατε τα εικονικά δεδομένα, έχει δημιουργηθεί ένας χρήστης με τα ακόλουθα στοιχεία σύνδεσης:

> **email:** `admin@admin.com`
> **password:** `password`

Εικόνα 49 (Voyager Dummy Admin Credential)

Αν δεν πήγες με τον ψεύτικο χρήστη, ίσως θελήσεις να ορίσεις δικαιώματα διαχειριστή σε έναν υπάρχοντα χρήστη. Αυτό μπορεί εύκολα να γίνει με την εκτέλεση αυτής της εντολής:

```
php artisan voyager:admin your@email.com
```

Εικόνα 50 (Voyager Command Admin Account)

Αν θέλετε να δημιουργήσετε ένα νέο χρήστη διαχειριστή, μπορείτε να περάσετε τη σημαία `--create`, κάπως έτσι:

```
php artisan voyager:admin your@email.com --create
```

Εικόνα 51 (Voyager New Admin Account Create)

Και θα σας ζητηθεί το όνομα χρήστη και ο κωδικός πρόσβασης.

Για να προσθέσετε τον provider υπηρεσιών Voyager, ανοίξτε το αρχείο `config/app.php` της εφαρμογής σας και προσθέστε την κλάση `TCG\Voyager\VoyagerServiceProvider::class` στον provider πίνακα όπως παρακάτω:

```php
<?php

'providers' => [
    // Laravel Framework Service Providers...
    //...

    // Package Service Providers
    TCG\Voyager\VoyagerServiceProvider::class,
    // ...

    // Application Service Providers
    // ...
],
```

Εικόνα 52 (Voyager Service Provider)

Το πρώτο πράγμα που πρέπει να κάνετε είναι να δημοσιεύσετε τα assets που έρχονται με το Voyager. Μπορείτε να το κάνετε αυτό εκτελώντας τις ακόλουθες εντολές:

```
php artisan vendor:publish --provider="TCG\Voyager\VoyagerServiceProvider"
php artisan vendor:publish --provider="Intervention\Image\ImageServiceProviderLaravel5"
```

Εικόνα 53 (Voyager Assets Command)

Στη συνέχεια, καλέστε τον php artisan migrate για να ξανά εγκαταστήσετε όλους τους πίνακες του Voyager από την αρχή.

Τώρα τρέξτε:

`php artisan db:seed --class=VoyagerDatabaseSeeder`
Για να δημιουργήσετε ορισμένα απαραίτητα δεδομένα στη βάση δεδομένων σας:

`php artisan hook:setup`
για να εγκαταστήσετε το σύστημα hooks, και :

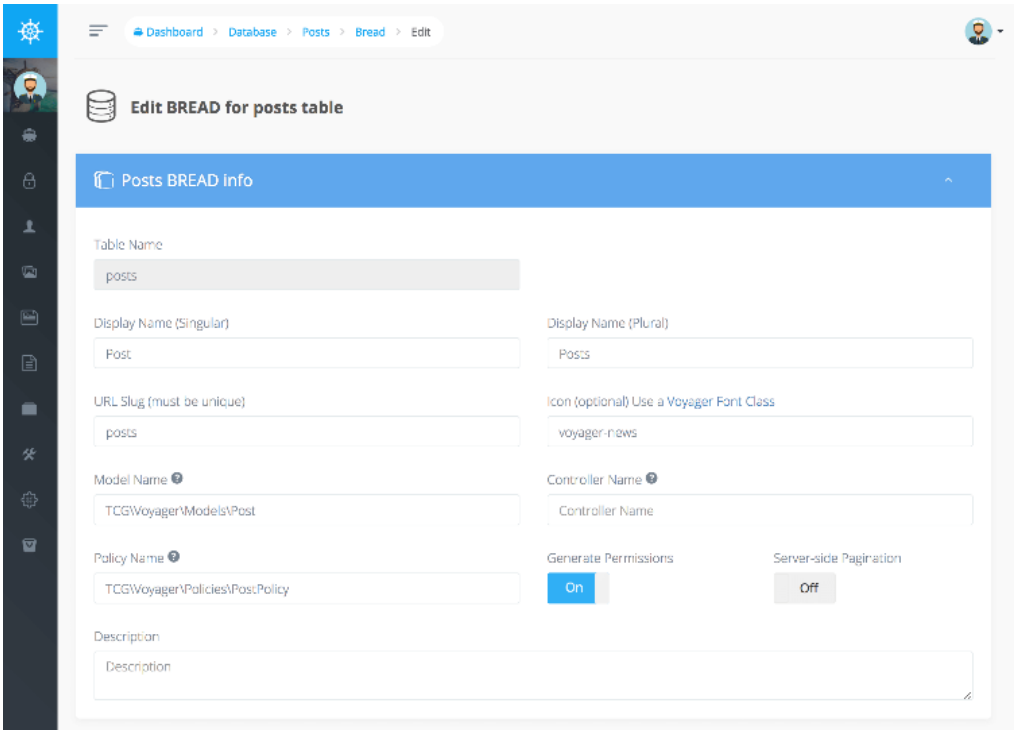
`php artisan storage:link`
για να δημιουργήσετε το storage symlink στον δημόσιο φάκελο.

Και τέλος, εκτελέστε το `dump-autoload` για να ολοκληρώσετε την εγκατάσταση!

6.4 Δυνατότητες του Voyager

6.4.1 Γενικά

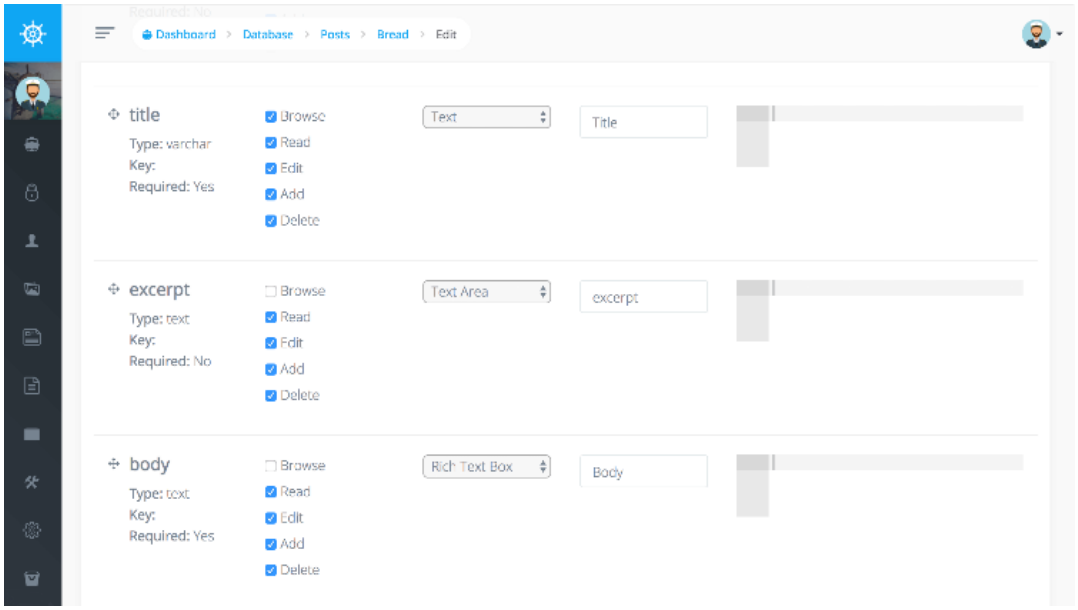
Όταν προσθέτετε ή επεξεργάζεστε το τρέχον BREAD για έναν πίνακα βάσης δεδομένων, θα δείτε πρώτα τις πληροφορίες BREAD που σας δίνουν τη δυνατότητα να ορίσετε τα Ονόματα εμφάνισης, το slug, το icon, το Model τον Controller Namespace και το όνομα Policy. Μπορείτε επίσης να επιλέξετε αν θα θέλατε να δημιουργήσετε δικαιώματα για αυτόν τον τύπο BREAD.



Εικόνα 54 (Voyager Bread System)

Όταν κάνετε κύλιση προς τα κάτω, θα δείτε κάθε μια από τις σειρές που σχετίζονται με τον συγκεκριμένο πίνακα όπου μπορείτε να επιλέξετε σε ποιες από τις προβολές σας θέλετε να δείτε καθένα από αυτά τα πεδία:

- **BROWSE** (το πεδίο θα εμφανιστεί κατά την περιήγηση στα τρέχοντα δεδομένα).
- **READ** (το πεδίο θα εμφανιστεί όταν κάνετε κλικ για να δείτε τα τρέχοντα δεδομένα).
- **EDIT** (το πεδίο θα είναι ορατό και θα σας επιτρέψει να επεξεργαστείτε τα δεδομένα).
- **ADD** (το πεδίο θα είναι ορατό όταν επιλέγετε να δημιουργήσετε νέο τύπο δεδομένων).
- **DELETE** (δεν αφορά τη διαγραφή, αλλα την ικανότητα να μπορεί να επιλεγεί ή να μη επιλεγεί).



Εικόνα 55 (Voyager BREAD Permission)

Μπορείτε επίσης να επιλέξετε να καθορίσετε τον τύπο της φόρμας που θέλετε να χρησιμοποιήσετε για κάθε πεδίο. Αυτό μπορεί να είναι Textbox, Textarea, Checkbox, Image και πολλοί άλλοι τύποι στοιχείων φόρμας.

Κάθε πεδίο έχει επίσης πρόσθετες λεπτομέρειες ή επιλογές που μπορούν να συμπεριληφθούν. Αυτοί οι τύποι είναι checkboxes, dropdowns, radio buttons και image.

6.4.2 Validation

Μέσα στην ενότητα "Προαιρετικά στοιχεία" για κάθε σειρά στο BREAD μπορείτε επίσης να καθορίσετε κανόνες επαλήθευσης με κάποιο απλό JSON. Ακολουθεί ένα παράδειγμα για τον τρόπο προσθήκης ενός κανόνα επικύρωσης για το required και μέγιστο μήκος 12.

```
{
    "validation": {
        "rule": "required|max:12"
    }
}
```

Εικόνα 56 (Voyager Validation Length)

Επιπλέον, μπορεί να θέλετε να προσθέσετε μερικά προσαρμοσμένα μηνύματα λάθους τα οποία μπορούν να υλοποιηθούν ως εξής:

```
{
    "validation": {
        "rule": "required|max:12",
        "messages": {
            "required": "This :attribute field is a must.",
            "max": "This :attribute field maximum :max."
        }
    }
}
```

Εικόνα 57 (Voyager Validation Messages)

Μπορείτε να ορίσετε ξεχωριστούς κανόνες επαλήθευσης για την επεξεργασία και την προσθήκη:

```
{
    "validation": {
        "rule": "required",
        "edit": {
            "rule": "sometimes|min:3"
        },
        "add": {
            "rule": "min:3"
        }
    }
}
```

Εικόνα 58 (Voyager different add/edit rules)

Μπορείτε να βρείτε μια λίστα όλων των διαθέσιμων κανόνων επικύρωσης στα έγγραφα Laravel.

6.4.3 Παραγωγή Slug

Χρησιμοποιώντας τον BREAD builder μπορεί να θέλετε να δημιουργήσετε αυτόματα slugs μιας συγκεκριμένης εισόδου. Ας πούμε ότι έχετε μερικές θέσεις, οι οποίες έχουν έναν τίτλο και ένα slug. Εάν θέλετε να δημιουργήσετε αυτόματα το slug από το χαρακτηριστικό τίτλου, μπορείτε να συμπεριλάβετε τα ακόλουθα προαιρετικά στοιχεία:

```
{
    "slugify": {
        "origin": "title",
        "forceUpdate": true
    }
}
```

Εικόνα 59 (Voyager Slugify)

Αυτό θα δημιουργήσει αυτόματα το slug από την είσοδο του πεδίου title. Αν ένα slug υπάρχει ήδη, θα ενημερωθεί μόνο αν έχει ενεργοποιηθεί η forceUpdate, από προεπιλογή αυτό είναι απενεργοποιημένο.

6.4.4 Επιλογές προβολής

Υπάρχουν επίσης μερικές επιλογές που μπορείτε να συμπεριλάβετε για να αλλάξετε τον τρόπο εμφάνισης του BREAD. Μπορείτε να προσθέσετε ένα κλειδί οθόνης στο αντικείμενο json σας και να αλλάξετε το πλάτος του συγκεκριμένου πεδίου και ακόμη και να καθορίσετε ένα προσαρμοσμένο αναγνωριστικό.

```
{
    "display": {
        "width": "3",
        "id": "custom_id"
    }
}
```

Εικόνα 60 (Voyager Validation Display Width)

Το πλάτος εμφανίζεται σε ένα σύστημα 12-grid. Η τοποθέτησή του με πλάτος 3 θα καλύπτει το 25% του πλάτους.

Το id θα σας επιτρέψει να ορίσετε ένα προσαρμοσμένο custom_id γύρω από το στοιχείο σας. παράδειγμα:

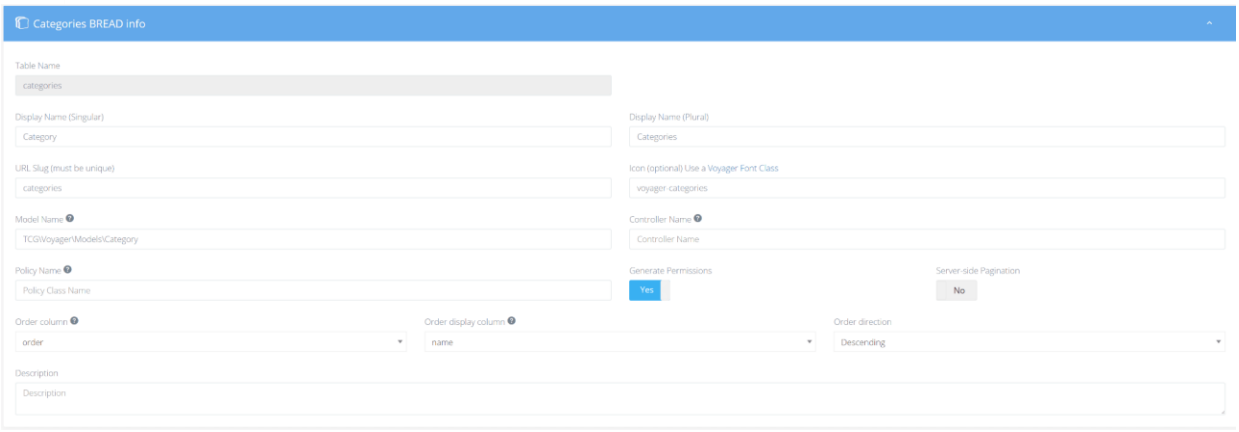
```
<div id="custom_id">
    <!-- Your field element -->
</div>
```

Εικόνα 61(Voyager Div custom_id)

6.4.5 Ταξινόμηση Bread Item

Μπορείτε να ορίσετε την προεπιλεγμένη σειρά για την αναζήτηση BREAD’s και να ταξινομήσετε τα στοιχεία σας BREAD με drag-and-drop.

Για αυτό πρέπει πρώτα να αλλάξετε τις ρυθμίσεις για το BREAD:



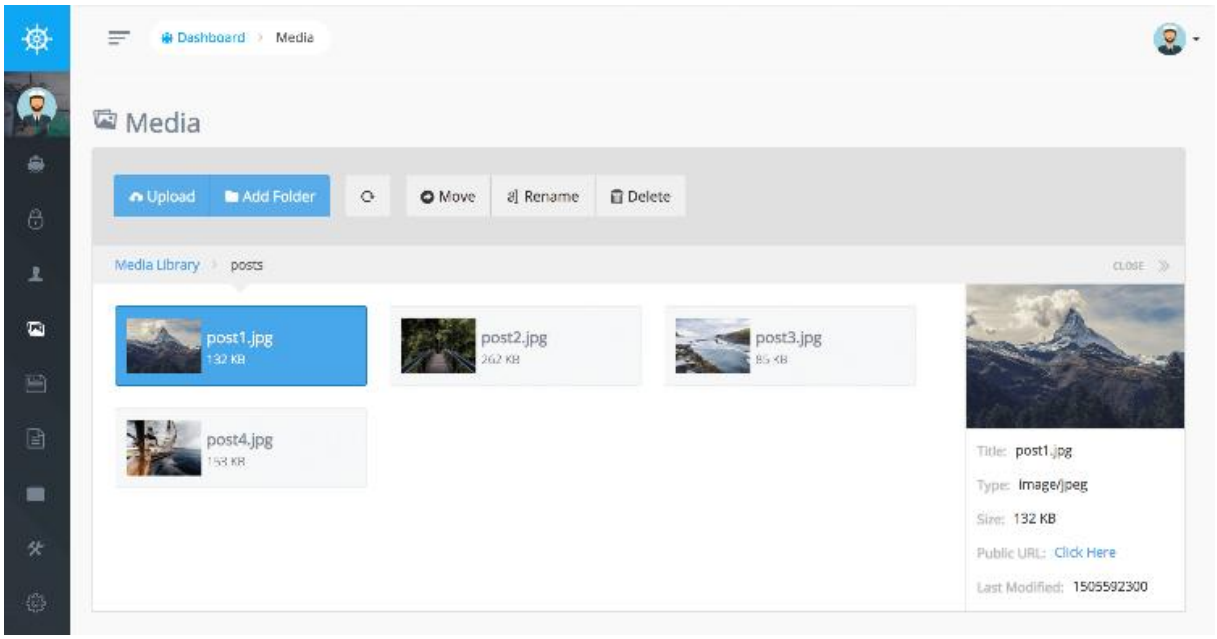
Εικόνα 62 (Voyager Bread Infos)

Η ταξινόμηση στήλης είναι το πεδίο στον πίνακα όπου η σειρά αποθηκεύεται ως ακέραιος αριθμός.
Η ταξινόμηση στήλη προβολής είναι το πεδίο που εμφανίζεται στη λίστα drag-drop.

Εικόνα 63 (Voyager Drag-Drop Order)

6.4.6 Media Manager

Το Voyager διαθέτει ένα ολοκληρωμένο πρόγραμμα διαχείρισης μέσων, το οποίο σας επιτρέπει να ανεβάζετε αρχεία, να ανακατεύετε αρχεία και να διαγράφετε αρχεία. Μπορείτε επίσης να προσθέσετε νέους φακέλους και να μετακινήσετε αρχεία / φακέλους. Βασικά οτιδήποτε θα μπορούσατε να κάνετε σε οποιοδήποτε τύπο Media Manager μπορείτε να το κάνετε και στον Voyager Media Manager.



Εικόνα 64 (Voyager Media Manager)

Μπορείτε επίσης να σύρετε και να αποθέσετε αρχεία στο κουμπί "upload" για να ανεβάσετε πολλά αρχεία.

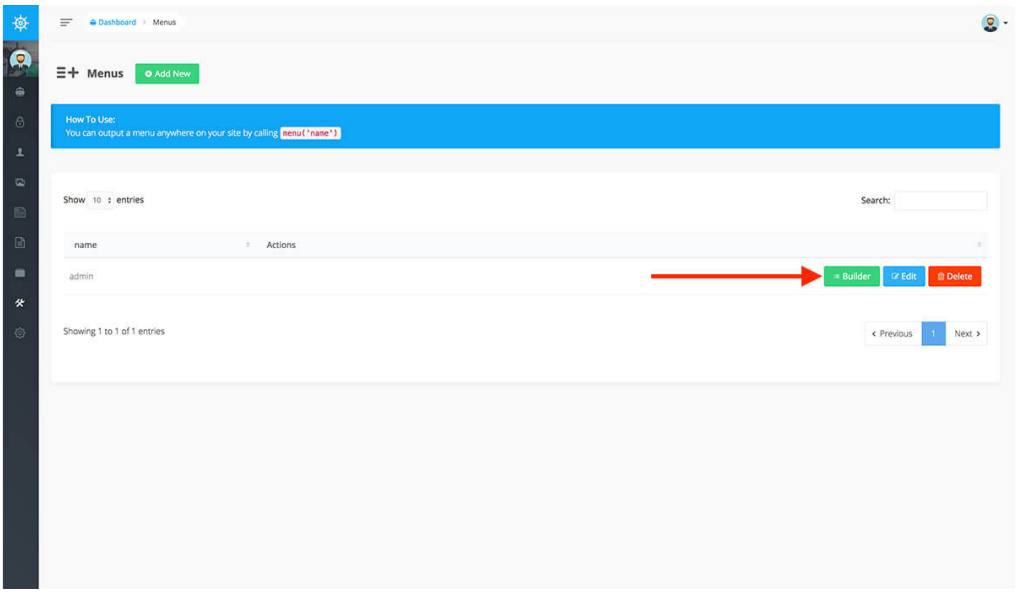
Ο Media Manager σάς επιτρέπει να δημιουργείτε thumbnails και να προσθέτετε watermarks σε μεταφορτωμένες εικόνες μέσω του αρχείου ρυθμίσεων.

Επισκεφτείτε το documentation για να μάθετε σχετικά με τις επιλογές διαμόρφωσης.

6.4.7 Menus και Menu Builder

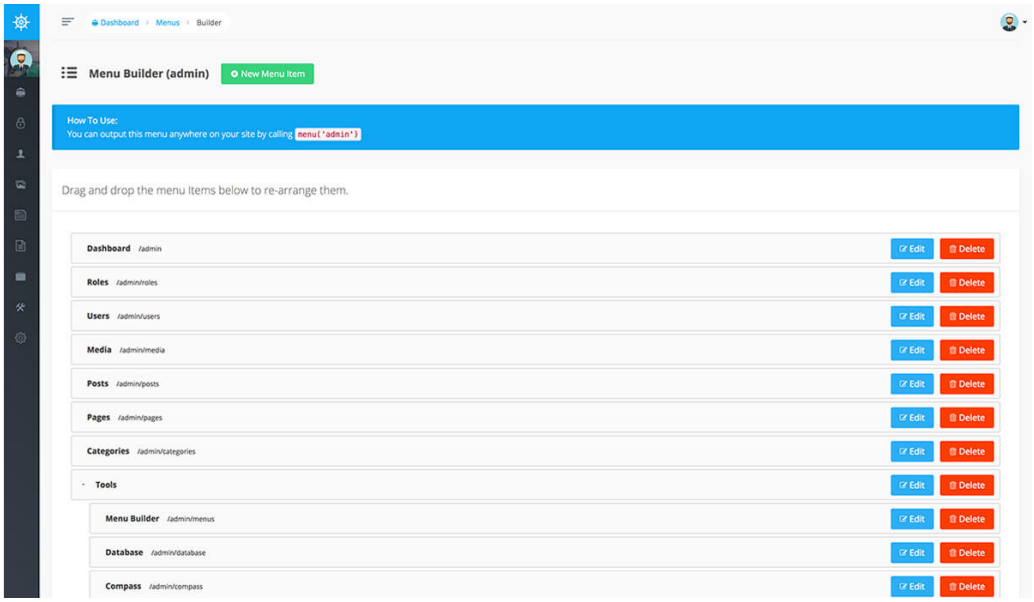
Με το Voyager μπορείτε εύκολα να δημιουργήσετε μενού για την εφαρμογή σας. Στην πραγματικότητα ο διαχειριστής του Voyager χρησιμοποιεί τον menu builder για την πλοήγηση που χρησιμοποιείτε στην αριστερή πλευρά.

Μπορείτε να δείτε τα τρέχοντα μενού σας κάνοντας κλικ στο κουμπί Tools-> Menu Builder. Μπορείτε να προσθέσετε, να επεξεργαστείτε ή να διαγράψετε οποιοδήποτε τρέχον μενού. Αυτό σημαίνει ότι μπορείτε να δημιουργήσετε ένα νέο μενού για το header, το sidebar και το footer του ιστότοπού σας.



Εικόνα 65 (Voyager Menu Builder)

Όταν είστε έτοιμοι να προσθέσετε στοιχεία μενού στο μενού σας, μπορείτε να κάνετε κλικ στο κουμπί builder του αντίστοιχου μενού:



Εικόνα 66 (Voyager Menu Builder Admin)

Αυτό θα σας οδηγήσει στο Builder μενού όπου μπορείτε να προσθέσετε, να επεξεργαστείτε και να διαγράψετε στοιχεία μενού.

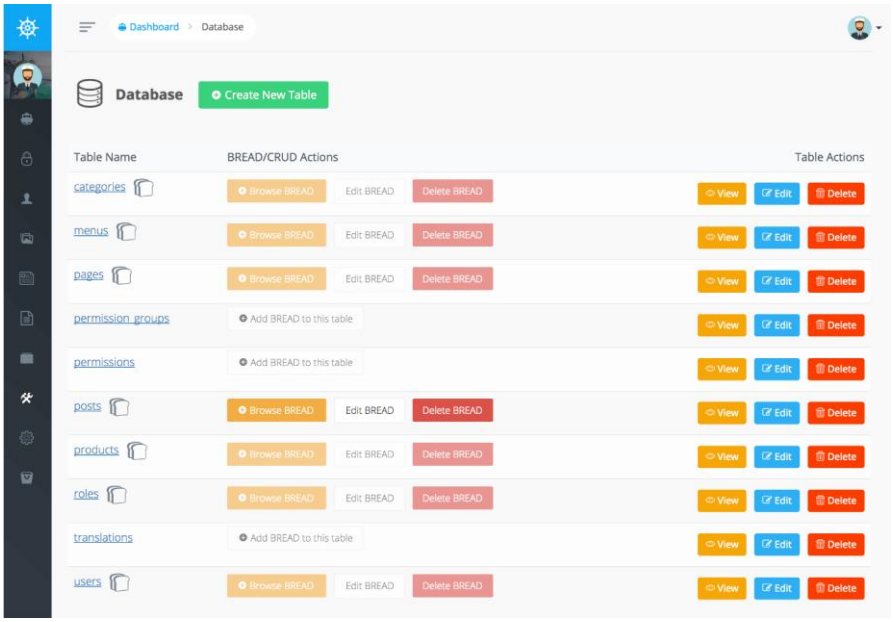
Αφού δημιουργήσετε και διαμορφώσετε το μενού σας, μπορείτε να εφαρμόσετε εύκολα αυτό το μενού στην εφαρμογή σας. Πείτε ότι έχουμε ένα μενού που ονομάζεται main. Μέσα από οποιοδήποτε αρχείο προβολής μπορούσαμε τώρα να εξάγουμε το μενού χρησιμοποιώντας τον ακόλουθο κώδικα:

```
menu('main');
```

Εικόνα 67 (Voyager menu code snippet)

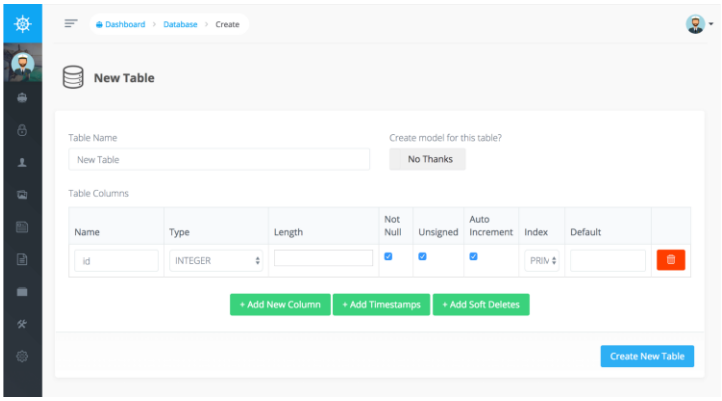
6.4.8 Database Manager

Το Voyager διαθέτει μερικά καταπληκτικά εργαλεία βάσης δεδομένων που σας επιτρέπουν να προσθέσετε / τροποποιήσετε / διαγράψετε ή να προβάλετε τους τρέχοντες πίνακες βάσης δεδομένων. Το άλλο δροσερό μέρος του Voyager είναι ότι μπορείτε να προσθέσετε τη λειτουργία BREAD ή (Browse, Read, Edit, Add, & Delete) σε οποιονδήποτε από τους πίνακες σας.



Εικόνα 68 (Voyager Database)

Μέσα στον πίνακα διαχείρισης μπορείτε να επισκεφθείτε το Tools->Database και θα μπορείτε να δείτε όλους τους τρέχοντες πίνακες στη βάση δεδομένων σας. Μπορείτε επίσης να κάνετε κλικ στο 'Δημιουργία νέου πίνακα' για να δημιουργήσετε έναν νέο πίνακα στη βάση δεδομένων σας. Όλοι οι πίνακες που δημιουργήθηκαν πρόσφατα θα χρησιμοποιούν το σύνολο χαρακτήρων που ορίζεται στην προεπιλεγμένη σύνδεσή σας στη βάση δεδομένων.



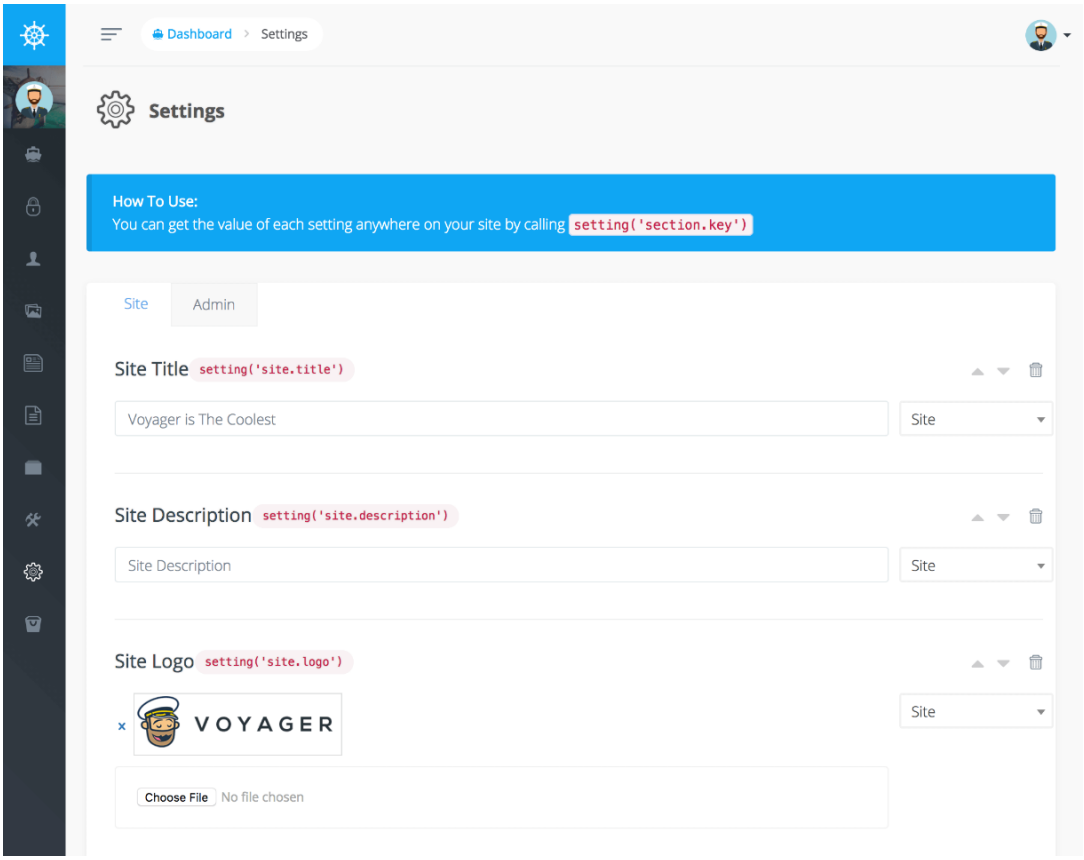
Εικόνα 69 (Voyager Create Database)

Εάν κάνετε κλικ στο όνομα του πίνακα, μπορείτε να δείτε το τρέχον σχήμα. Επιπλέον, μπορείτε να κάνετε κλικ στα κουμπιά Προβολή, Επεξεργασία ή Διαγραφή για να εκτελέσετε αυτήν την ενέργεια για αυτόν τον πίνακα.

Μπορείτε επίσης να επιλέξετε Προσθήκη BREAD (Αναζήτηση, ανάγνωση, επεξεργασία, προσθήκη και διαγραφή) για οποιονδήποτε από τους πίνακες βάσης δεδομένων σας. Μόλις ένας πίνακας έχει ήδη BREAD, μπορείτε να επιλέξετε να επεξεργαστείτε το τρέχον BREAD ή να διαγράψετε το BREAD για αυτόν τον πίνακα.

6.4.9 Settings

Η ενότητα "Ρυθμίσεις" σας επιτρέπει να προσθέσετε τυχόν ρυθμίσεις ευρείας περιοχής που θέλετε. Μπορείτε να προσθέσετε μια ρύθμιση μεταφόρτωσης εικόνας για το λογότυπο του ιστοτόπου σας ή ένα πλαίσιο κειμένου για την κύρια επικεφαλίδα στην αρχική σας σελίδα.

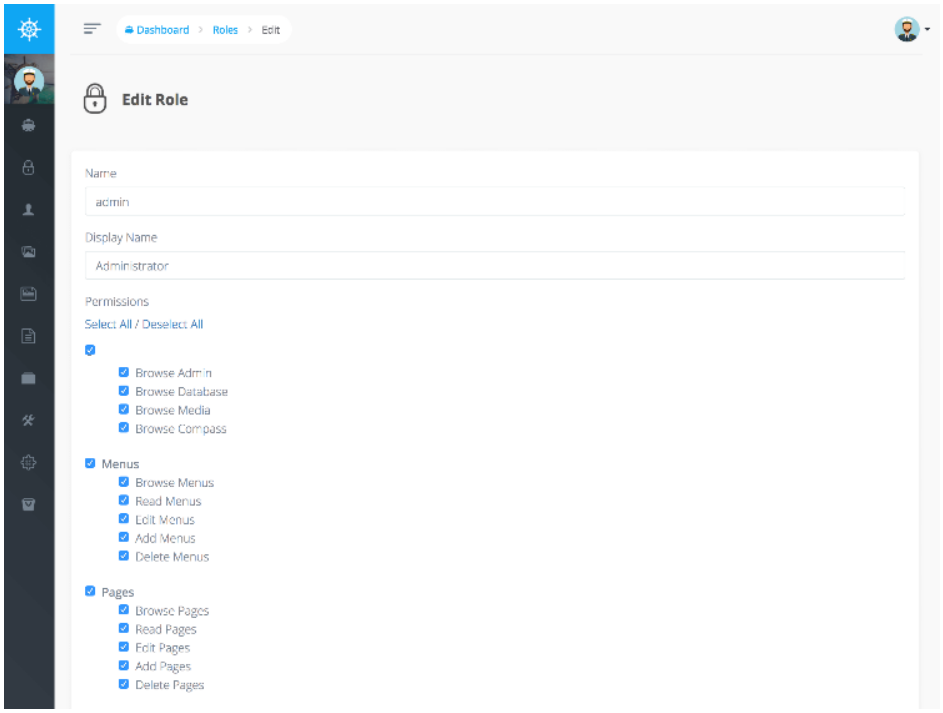


Εικόνα 70 (Voyager Settings)

6.4.10 Ρόλοι και Δικαιώματα

Το Voyager έρχεται από την αρχή με Ρόλους και δικαιώματα. Κάθε χρήστης έχει έναν ρόλο που έχει ένα σύνολο δικαιωμάτων.

Μέσα στον πίνακα οργάνων μπορείτε να επιλέξετε Προσθήκη, Επεξεργασία ή διαγραφή των τρεχόντων ρόλων. Επιπλέον, όταν κάνετε κλικ για να επεξεργαστείτε ένα συγκεκριμένο ρόλο, μπορείτε να καθορίσετε τα δικαιώματα BREAD.



Εικόνα 71 (Voyager Role Permission Edit)

Παρακάτω θα δείτε το σύστημα εξουσιοδότησης της Voyager ώστε να συμβαδίζει περισσότερο με το Laravel. Αυτό σημαίνει ότι μπορείτε να ελέγξετε για δικαιώματα με τους εξής τρόπους :

```
// via user object
$canViewPost = $user->can('read', $post);
$canViewPost = Auth::user()->can('read', $post);

// via controller
$canViewPost = $this->authorize('read', $post);
```

Εικόνα 72 (Voyager Permission Code)

Από default υπάρχουν ορισμένα δικαιώματα που μπορείτε να χρησιμοποιήσετε από προεπιλογή:

- browse_admin: Εάν ο χρήστης μπορεί να περιηγηθεί στον πίνακα διαχείρισης του Voyager.
- browse _database: Εάν ο χρήστης μπορεί να περιηγηθεί στην ενότητα μενού βάσης δεδομένων Voyager.
- browse _bread: Το αν ο χρήστης μπορεί να περιηγηθεί στην ενότητα μενού BREAD του Voyager.
- browse _media: Το αν ο χρήστης μπορεί να περιηγηθεί στην ενότητα των μέσων μαζικής ενημέρωσης Voyager.
- browse _menu: Εάν ο χρήστης μπορεί να περιηγηθεί στην ενότητα μενού Voyager.

- browse _settings: Εάν ο χρήστης μπορεί να περιηγηθεί στην ενότητα ρυθμίσεων Voyager.

- read_ settings: Εάν ο χρήστης μπορεί να δει ή να δει μια συγκεκριμένη ρύθμιση.
- edit_ settings: Εάν ο χρήστης μπορεί να επεξεργαστεί μια συγκεκριμένη ρύθμιση ή όχι.
- add_ settings: Εάν ο χρήστης μπορεί να προσθέσει μια νέα ρύθμιση ή όχι.
- delete_ settings: Εάν ο χρήστης μπορεί να διαγράψει μια συγκεκριμένη ρύθμιση.

Επιπλέον, μπορείτε να δημιουργήσετε δικαιώματα για κάθε τύπο BREAD που δημιουργείτε. Αυτό θα δημιουργήσει την άδεια περιήγησης, ανάγνωσης, επεξεργασίας, προσθήκης και διαγραφής.

Για παράδειγμα, ίσως δημιουργούμε έναν νέο τύπο BREAD από έναν πίνακα προϊόντων. Αν επιλέξετε να δημιουργήσετε δικαιώματα για τον πίνακα προϊόντων μας. Τα κλειδιά πρόσβασης θα είναι `browse_products, read_products, edit_products, add_products` και `delete_products.`

Κεφάλαιο 7: Εγκατάσταση της εφαρμογής CMS Online σε Virtual Server.

7.1 Τι είναι Virtual Machine Server;

Μια εικονική μηχανή είναι ένα αρχείο υπολογιστή, που ονομάζεται τυπικά μια εικόνα, η οποία συμπεριφέρεται σαν ένας πραγματικός υπολογιστής. Με άλλα λόγια, δημιουργώντας έναν υπολογιστή μέσα σε έναν υπολογιστή. Εκτελείται σε ένα παράθυρο, σαν οποιοδήποτε άλλο πρόγραμμα, δίνοντας στον τελικό χρήστη την ίδια εμπειρία σε μια εικονική μηχανή όπως θα είχε στο ίδιο το λειτουργικό σύστημα υποδοχής. Το εικονικό μηχάνημα είναι "sandboxed" από το υπόλοιπο σύστημα, πράγμα που σημαίνει ότι το λογισμικό μέσα σε μια εικονική μηχανή δεν μπορεί να διαφύγει ή να παραβιάσει τον ίδιο τον υπολογιστή. Αυτό δημιουργεί ένα ιδανικό περιβάλλον για τη δοκιμή άλλων λειτουργικών συστημάτων, συμπεριλαμβανομένων των beta κυκλοφοριών, την πρόσβαση σε δεδομένα μολυσμένα από ιούς, τη δημιουργία αντιγράφων ασφαλείας του λειτουργικού συστήματος και την εκτέλεση λογισμικού ή εφαρμογών σε λειτουργικά συστήματα για τα οποία δεν προορίζονταν αρχικά.

Πολλαπλές εικονικές μηχανές μπορούν να εκτελούνται ταυτόχρονα στον ίδιο φυσικό υπολογιστή. Για διακομιστές, τα πολλαπλά λειτουργικά συστήματα λειτουργούν δίπλα-δίπλα με ένα κομμάτι λογισμικού που ονομάζεται hypervisor για να τα διαχειρίζονται, ενώ οι επιτραπέζιοι υπολογιστές συνήθως χρησιμοποιούν ένα λειτουργικό σύστημα για να τρέχουν τα άλλα λειτουργικά συστήματα στα παράθυρα του προγράμματος. Κάθε εικονική μηχανή παρέχει δικό της εικονικό υλικό, συμπεριλαμβανομένων CPUs, μνήμης, σκληρών δίσκων, διεπαφών δικτύου και άλλων συσκευών. Το εικονικό υλικό στη συνέχεια αντιστοιχίζεται στο πραγματικό υλικό στο φυσικό μηχάνημα, το οποίο εξοικονομεί κόστος μειώνοντας την ανάγκη για συστήματα φυσικού υλικού μαζί με τα συνακόλουθα έξοδα συντήρησης που πηγαίνουν μαζί του καθώς και μειώνοντας τη ζήτηση ισχύος και ψύξης.

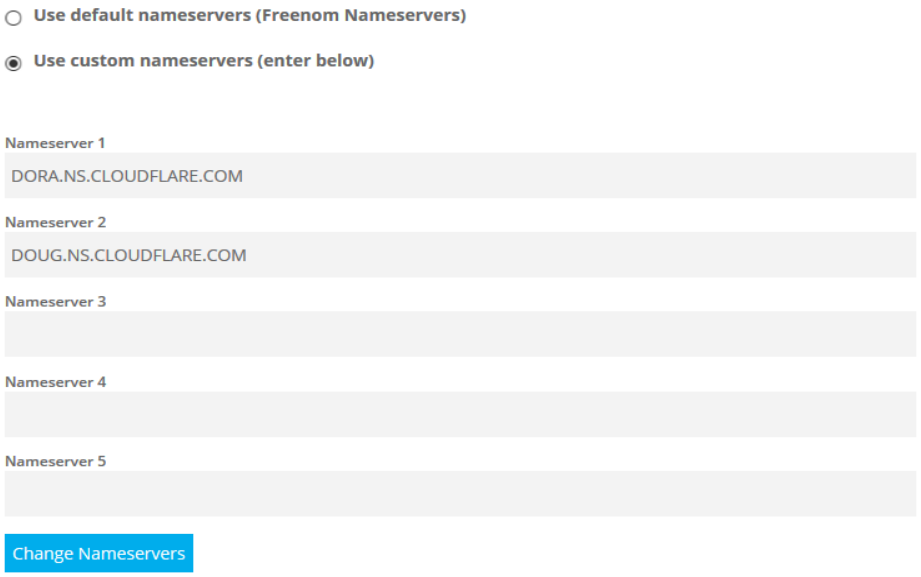
7.2 Αγορά domain name και χρήση σε virtual host στο VM

Έχω πάρει ένα δωρεάν domain name από την ιστοσελίδα freenom.com ώστε να χρησιμοποιήσω για την ιστοσελίδα μου. Παρακάτω στην εικόνα θα δείξω το domain και κάποια configuration που έχω κάνει με την Cloudflare για να κάνω ασφαλές (https) την ιστοσελίδα μου.

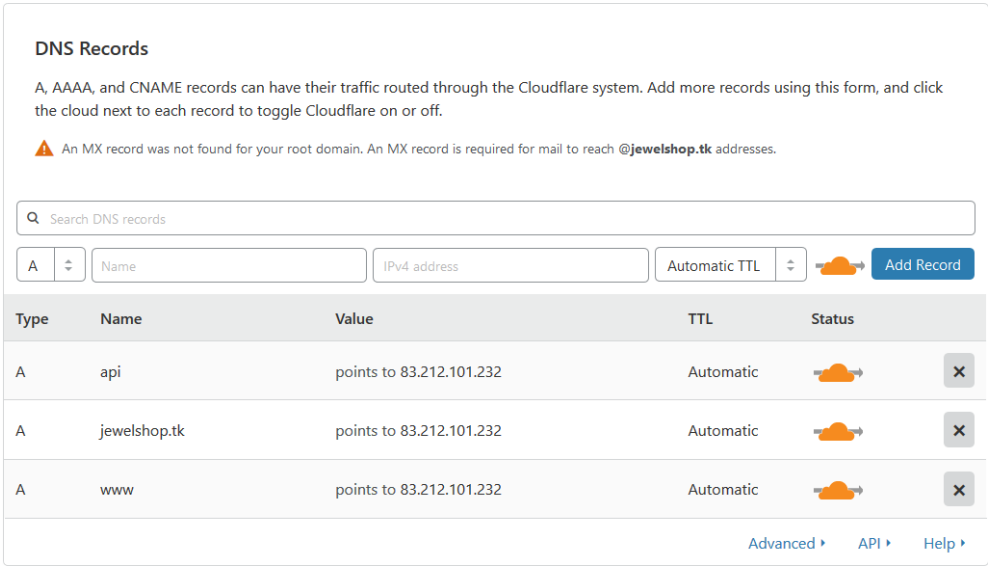
Domain	Registration Date	Expiry date	Status	Type	
jewelshop.tk	30/01/2019	30/01/2020	ACTIVE	Free	Manage Domain
gtsaxrelias.tk	31/01/2019	31/01/2020	ACTIVE	Free	Manage Domain

Εικόνα 73 (Active Domain Names)

Εδώ έχω τα nameservers της Cloudflare ώστε να μπορώ να ελέγχω από εκεί τα DNS records.



Εικόνα 74 (Use Custom Nameserver)



Εικόνα 75 (DNS Records)

Παρακάτω θα δείξω τον κώδικα που θα προσθέσω στο αρχείο httpd-vhosts.conf του Xampp ώστε να μπορεί να ξεχωρίσει το domain name με την μοναδική διεύθυνση του διακομιστή.

```
<VirtualHost *:80>
    ServerAdmin admin@jewelshop.tk
    DocumentRoot "C:\xampp\htdocs\jewelshop\public"
    ServerName jewelshop.tk
    ServerAlias www.jewelshop.tk
    ErrorLog "logs/jewelshop.tk-error.log"
    CustomLog "logs/jewelshop.tk-access.log" common
</VirtualHost>
```

Εικόνα 76 (Xampp VHosts Code)

Και τέλος για να μπορεί ο διακομιστής να λειτουργήσει και με πολλαπλά domain names χρειάζεται να γράψω στο host file των windows.

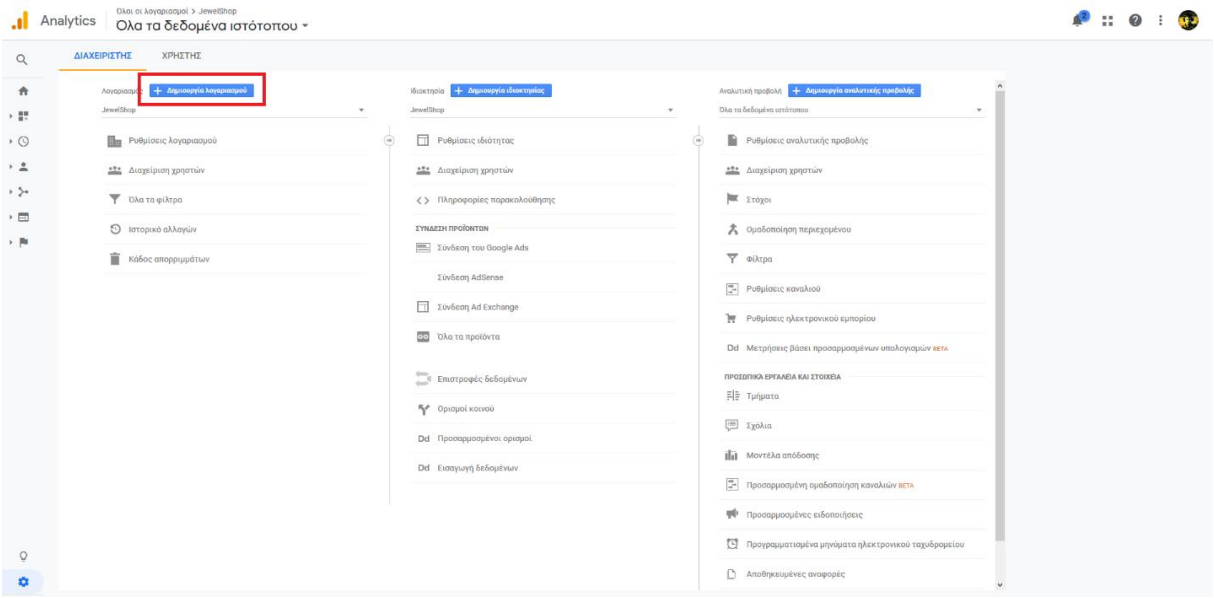
```
127.0.0.1  gtsaxrelias.tk
127.0.0.1  www.gtsaxrelias.tk
127.0.0.1  jewelshop.tk
127.0.0.1  api.jewelshop.tk
127.0.0.1  www.jewelshop.tk
```

Εικόνα 77 (Windows hosts File)

7.3 Εγγραφή για e-mail server (Mail gun)

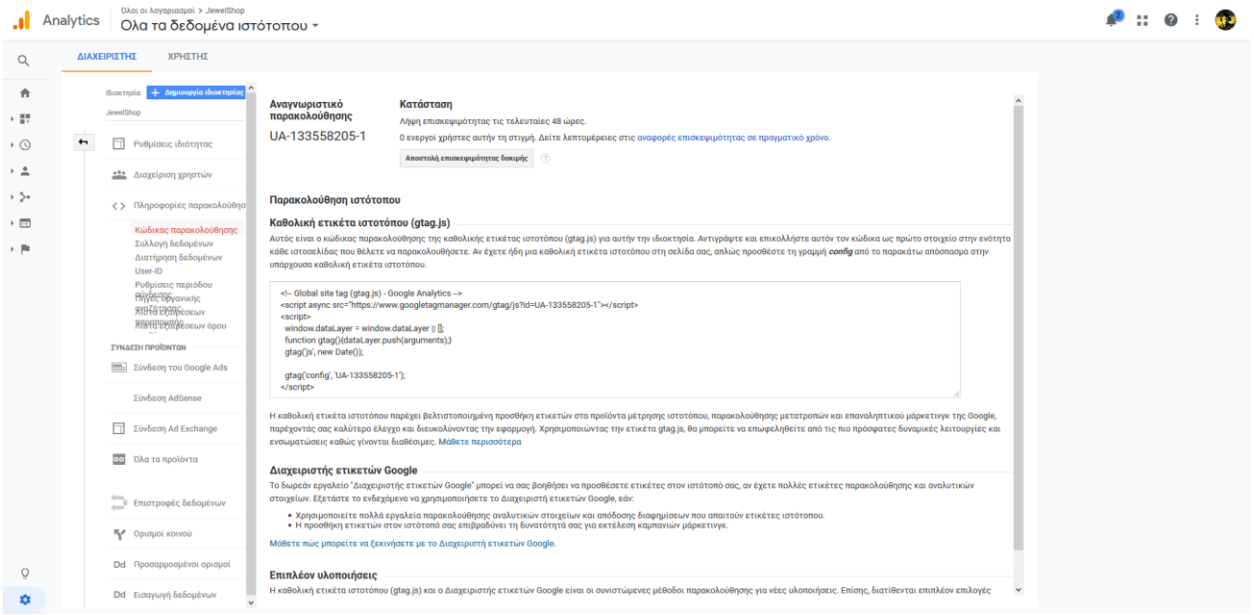
7.4 Χρήση Google Analytics και παραγωγή key

Για να δημιουργήσετε ένα νέο Google Analytic λογαριασμό θα πρέπει να πάμε στις ρυθμίσεις / Διαχειριστής και να πατήσουμε το κουμπί δημιουργία λογαριασμού πάνω αριστερά όπως βλέπουμε και στην παρακάτω εικόνα.



Εικόνα 78 (Google Analytics Data)

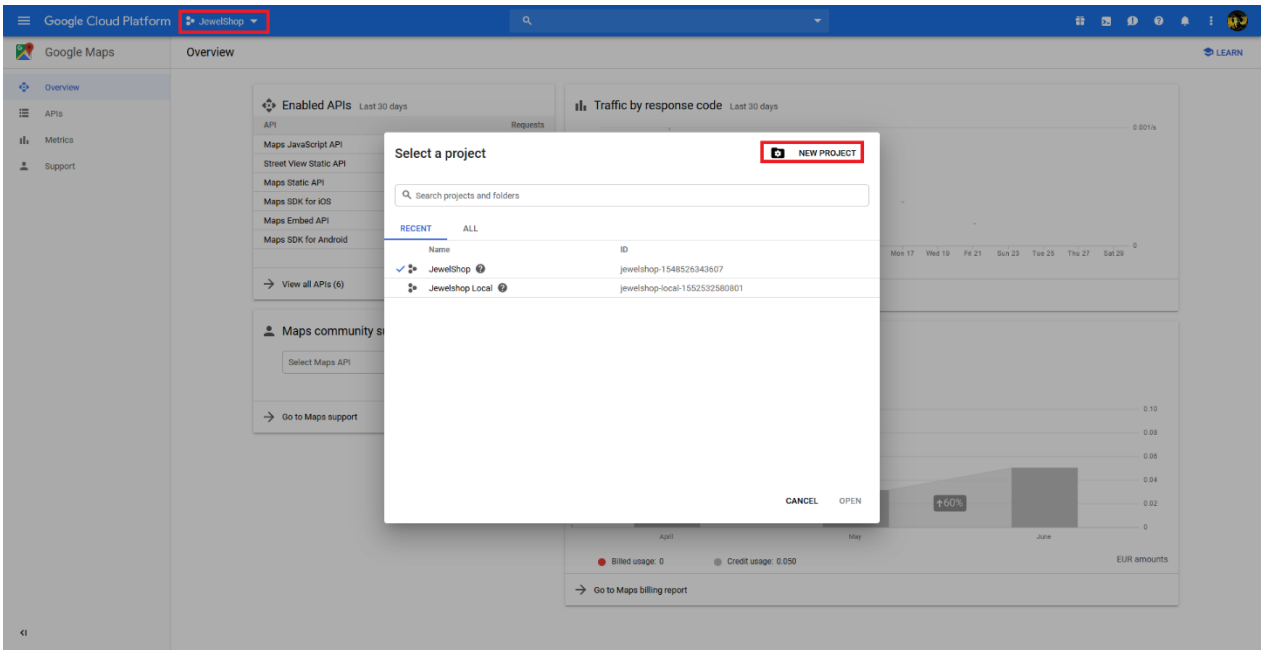
Μετά στο μεσαίο tab <> πληροφορίες παρακολούθησης πηγαίνουμε και περνούμε το κλειδί και τον κώδικα ώστε να το προσθέσουμε στον κώδικα μας.



Εικόνα 79 (Google Analytics Track Code)

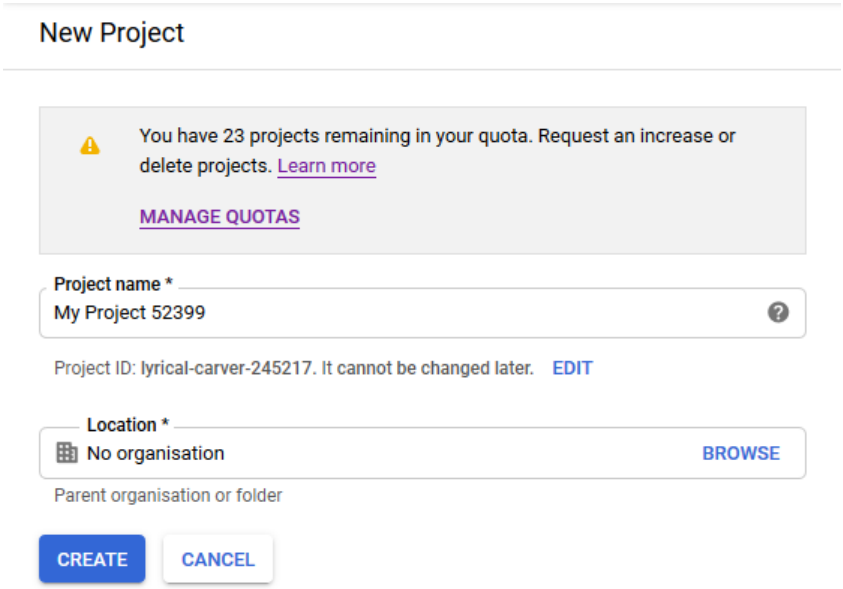
7.5 Χρήση Google Maps και παραγωγή key

Για να δημιουργήσω ένα νέο project ώστε να προσθέσω google map key πρέπει να κάνω την παρακάτω διαδικασία.



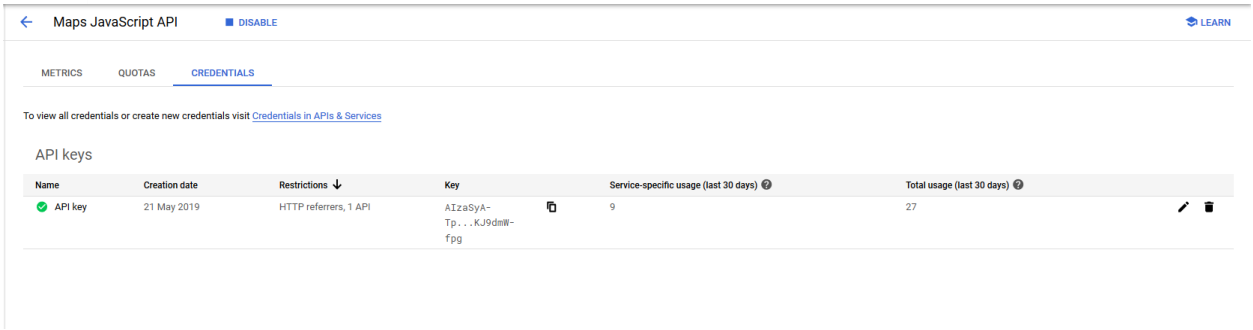
Εικόνα 80 (Google Maps New Project Create)

Παρακάτω δημιουργούμε το όνομα του project και σε ποιον οργανισμό βρίσκετε.



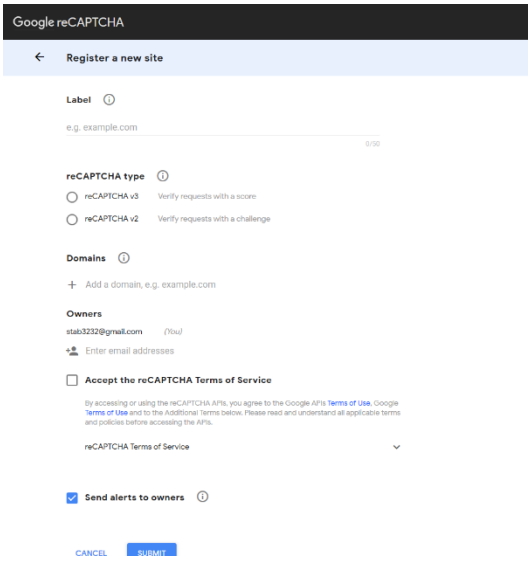
Εικόνα 81 (Google Maps New Project Name)

Μετά από αυτήν την διαδικασία πρέπει να ενεργοποιήσουμε και να εγκαταστήσουμε το JavaScript map API και τέλος να πάρουμε το κλειδί.



Εικόνα 82 (Google Maps API Key)

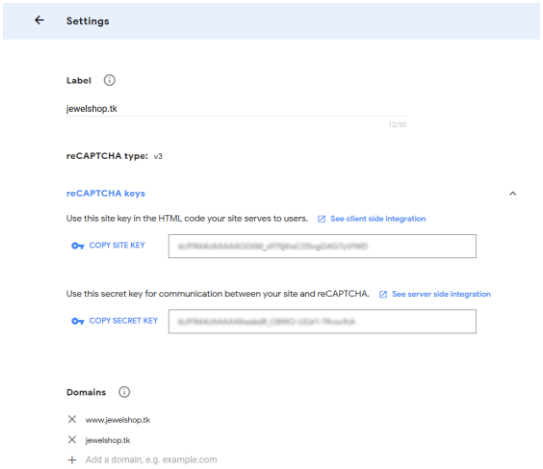
7.6 Χρήση Google re-Captcha και παραγωγή key



Εικόνα 83 (Google re-CAPTCHA Register new site)

Εδώ βλέπουμε το πως να δημιουργήσουμε ένα νέο κλειδί για το Google re-CAPTCHA και μπορούμε να επιλέξουμε την έκδοση v2 η v3 για να προχωρήσουμε για την ιστοσελίδα μας.

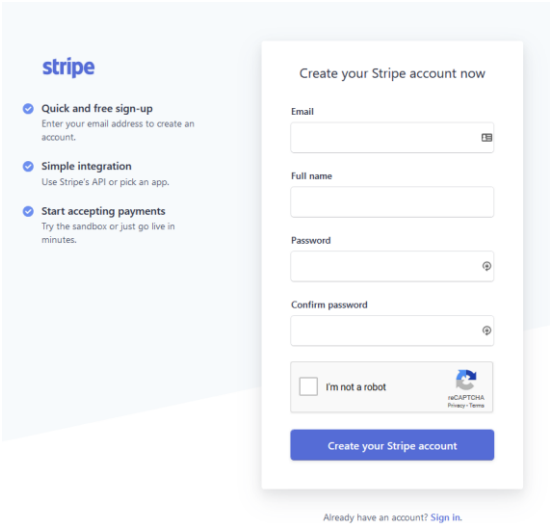
Σε αυτήν την εικόνα μπορούμε να δούμε τα δυο κλειδιά που θα μας χρειαστούν για να χρησιμοποιήσουμε το captcha τα οποία θα τα βρούμε στις ρυθμίσεις όταν περάσουμε το από πάνω βήμα και δημιουργήσουμε την ιστοσελίδα μας.



Εικόνα 84 (Google re-CAPTCHA Keys)

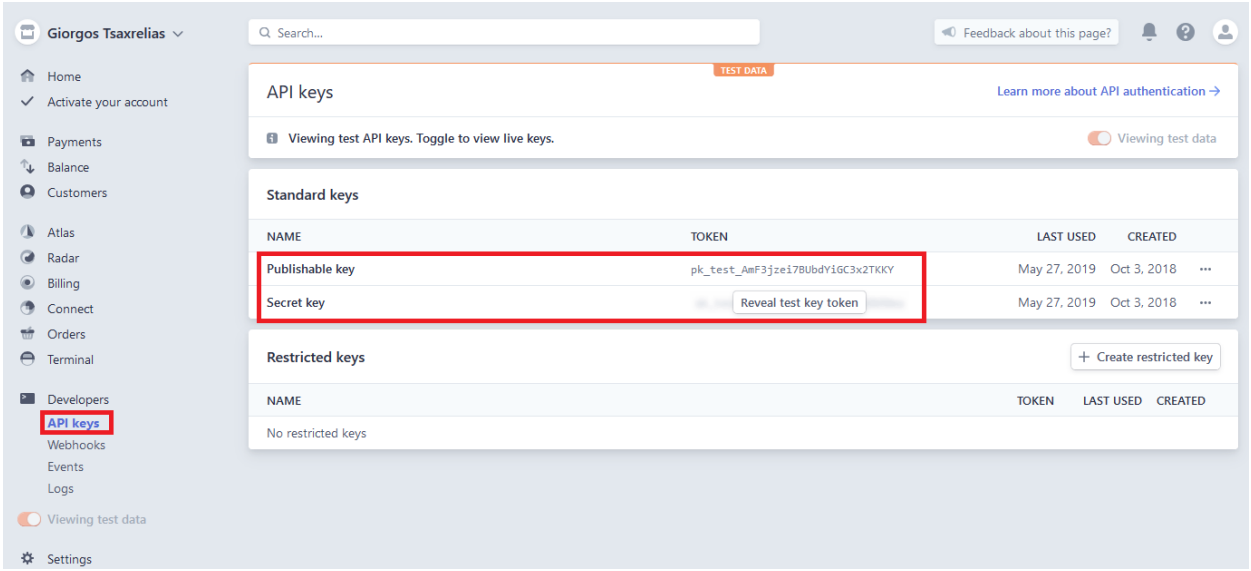
7.7 Εγγραφή υπηρεσίας Stripe για ηλεκτρονική πληρωμή

Για αρχή όπως σε κάθε ιστοσελίδα πρέπει να εγγράφουμε για να μπορέσουμε να πάρουμε δοκιμαστικό λογαριασμό.



Εικόνα 85 (Stripe Create Account)

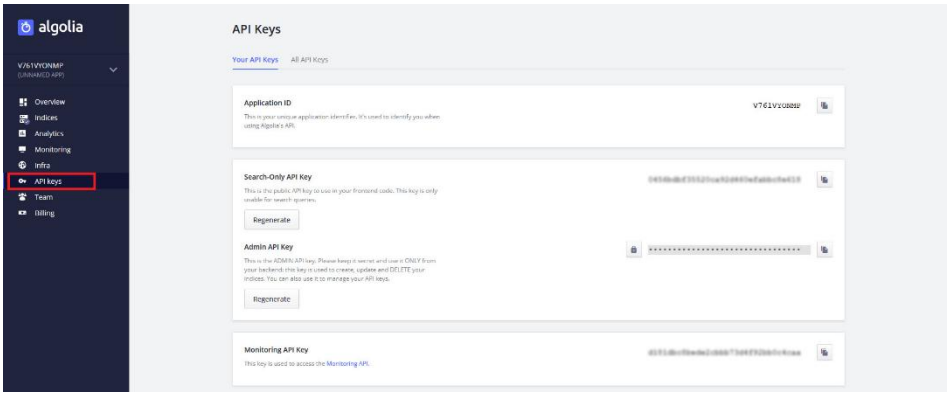
Και παρακάτω πρέπει να πάμε στην καρτέλα για προγραμματιστές και να πατήσουμε στο API keys όπως θα δούμε παρακάτω.



Εικόνα 86 (Stripe API Keys)

7.8 Χρήση του Algolia Search Engine και παραγωγή key

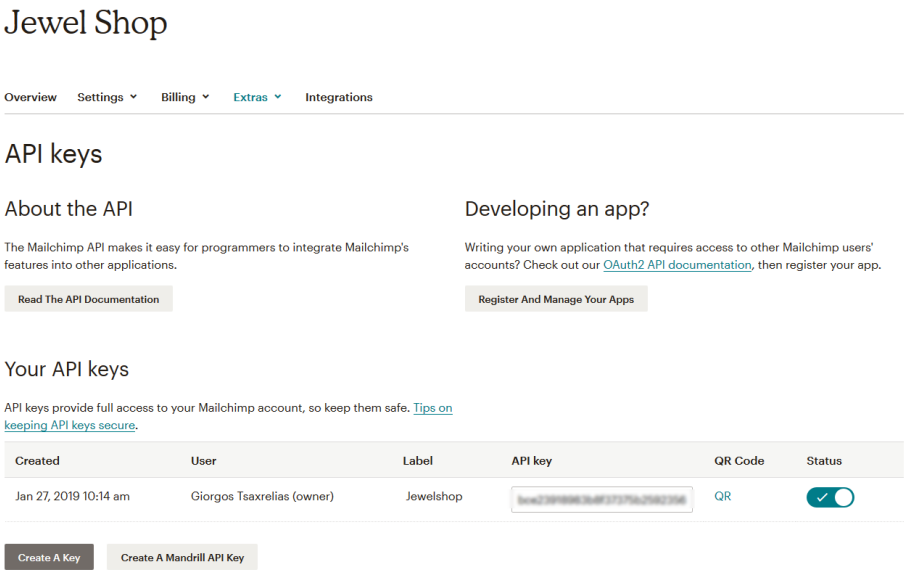
Θα χρειαστεί να εγγραφούμε και σε αυτήν την ιστοσελίδα και να πάρουμε πάλι τα κλειδιά ώστε να μπορούμε να χρησιμοποιήσουμε το πακέτο για την αναζήτηση προϊόντων. Παρακάτω θα δούμε σε ποια καρτέλα θα μπορέσουμε να πάρουμε τα κλειδιά ώστε ένα να βάλουμε στο backend και ένα στο frontend για να μπορέσουμε να δουλέψουμε με το search.



Εικόνα 87 (Algolia API Keys)

7.9 Εγγραφή MailChimp για newsletters

Όπως κάναμε σε όλα τα υπόλοιπα πακέτα έτσι και εδώ θα χρειαστούμε μια εγγραφή στην ιστοσελίδα και μετά πρέπει να πάμε Extras / API keys για να πάρουμε το κλειδί ώστε να το χρησιμοποιήσουμε στο πακέτο που έχω βάλει για να μπορούμε να κρατάμε την λίστα των πελατών.



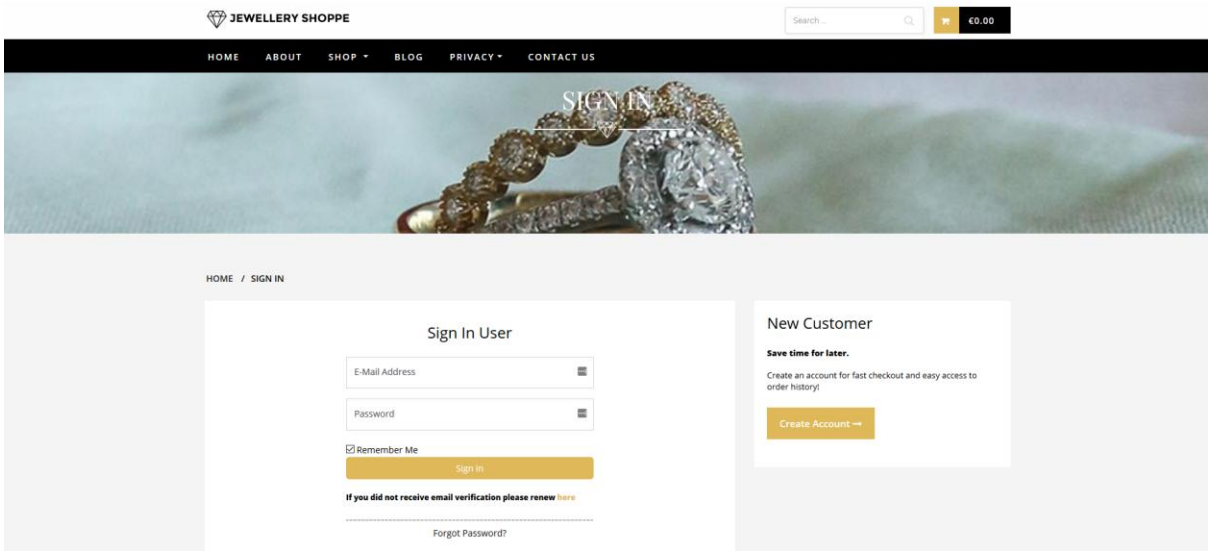
Εικόνα 88 (MailChimp API Keys)

Κεφάλαιο 8: Προεπισκόπηση Ιστοσελίδας

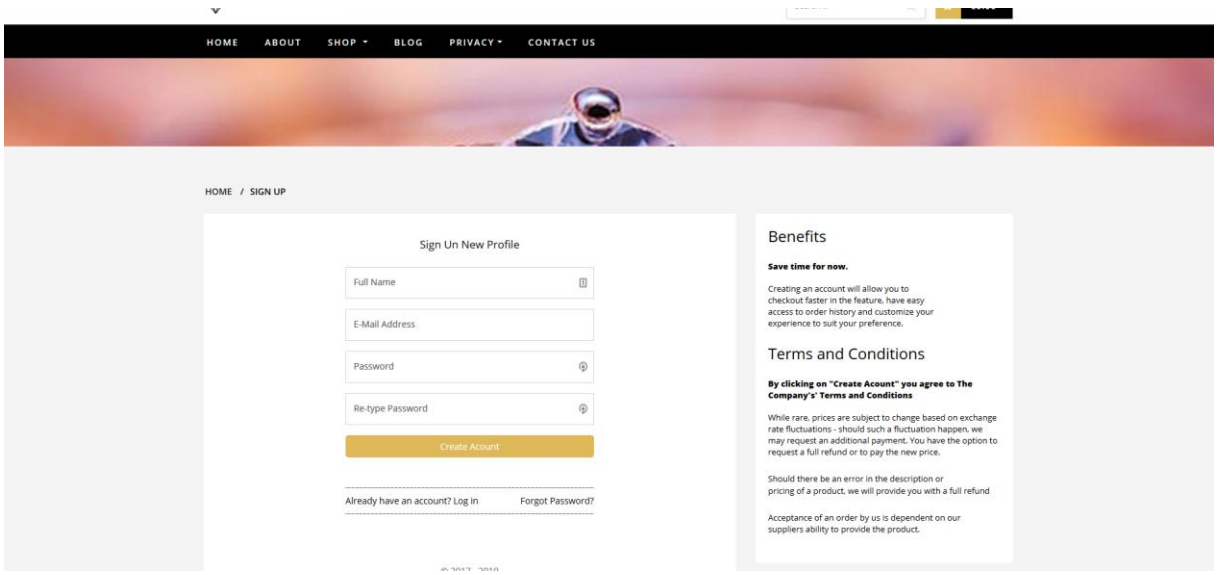
8.1 Κύρια Σημεία Προσομοίωση Επίσκεψης

8.1.1 Περιβάλλον Χρήστη

Παρακάτω θα σας δείξω το γραφικό περιβάλλον για την σύνδεση ενός χρήστη και την εγγραφή του στο κατάστημα.

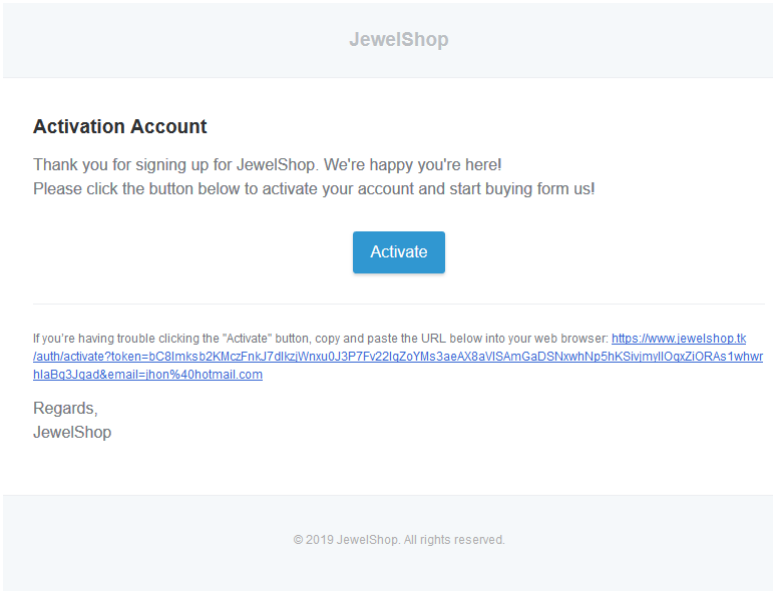


Εικόνα 89 (Login Page Site)



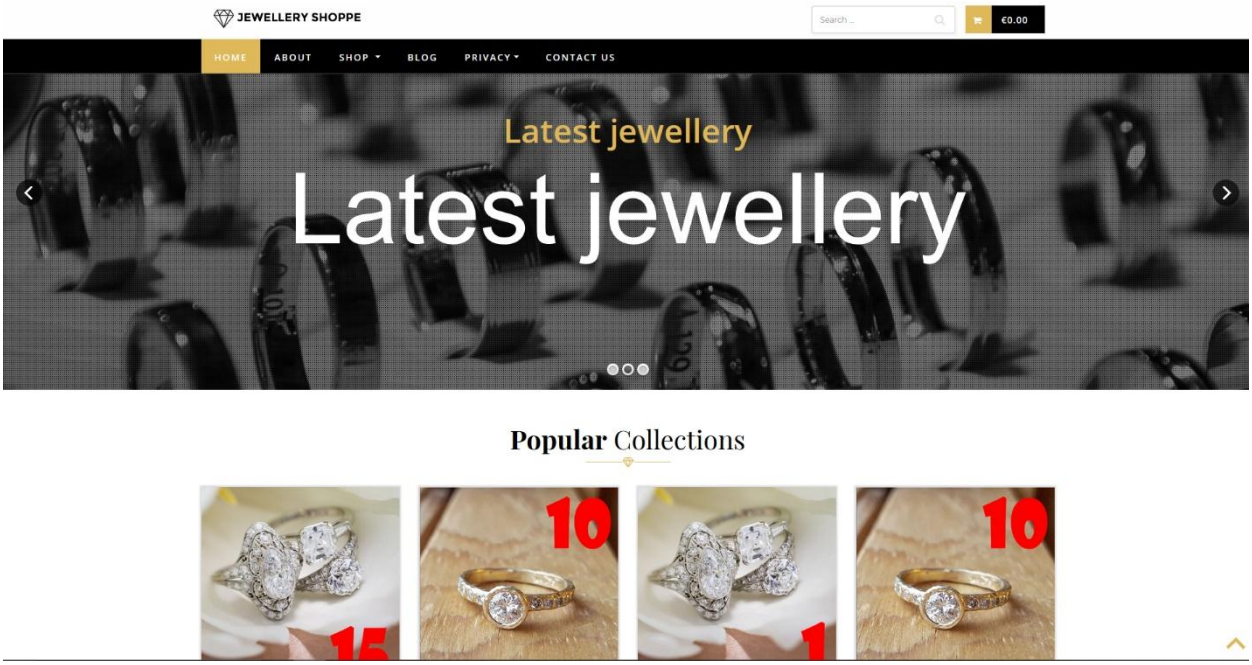
Εικόνα 90 (Register Page Site)

Όταν γίνετε εγγραφή ενός νέου χρήστη απαιτείται να γίνει επιβεβαίωση με το email που έχει δώσει ο χρήστης και παρακάτω θα δείξω πως φαίνεται το verify στο email.



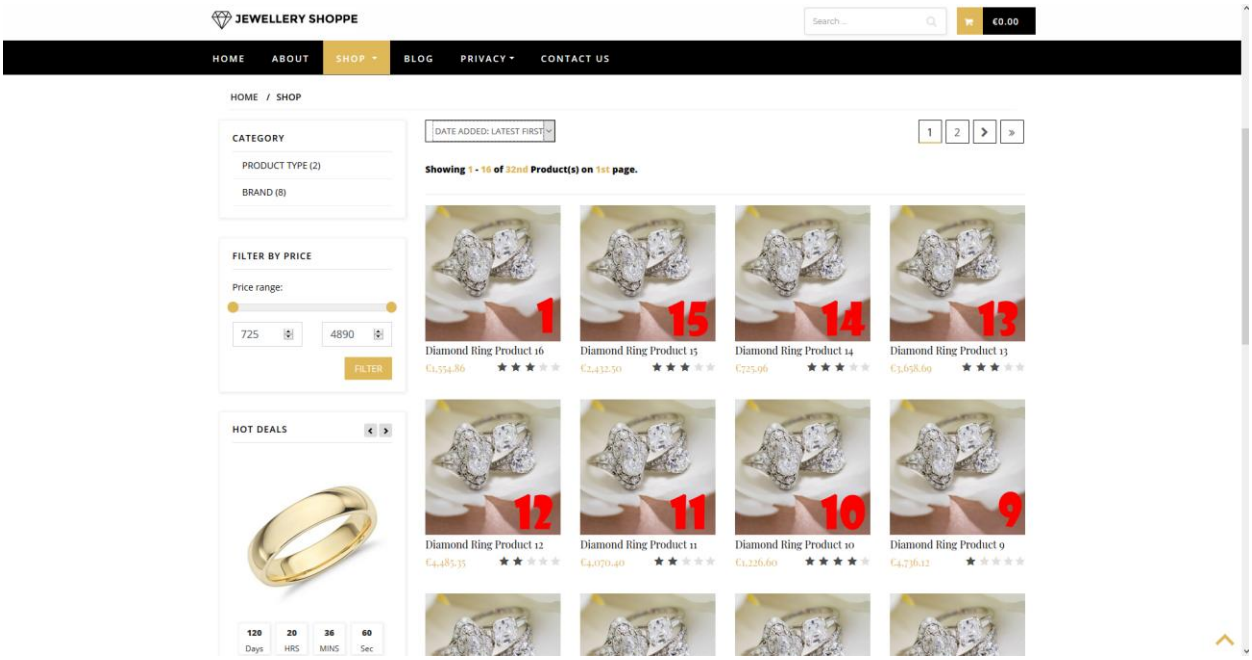
Εικόνα 91 (Email Activation User)

Παρακάτω θα δούμε την αρχική σελίδα η οποία περιλαμβάνει ένα carousel στην πάνω μεριά και πιο κάτω θα δούμε τα πιο δημοφιλές προϊόντα, τα καλύτερα προϊόντα του καταστήματος, τις κατηγορίες, τα τελευταία 8 προϊόντα που προσδέθηκαν από τον διαχειριστή και τέλος τα brand που έχει το κατάστημα.



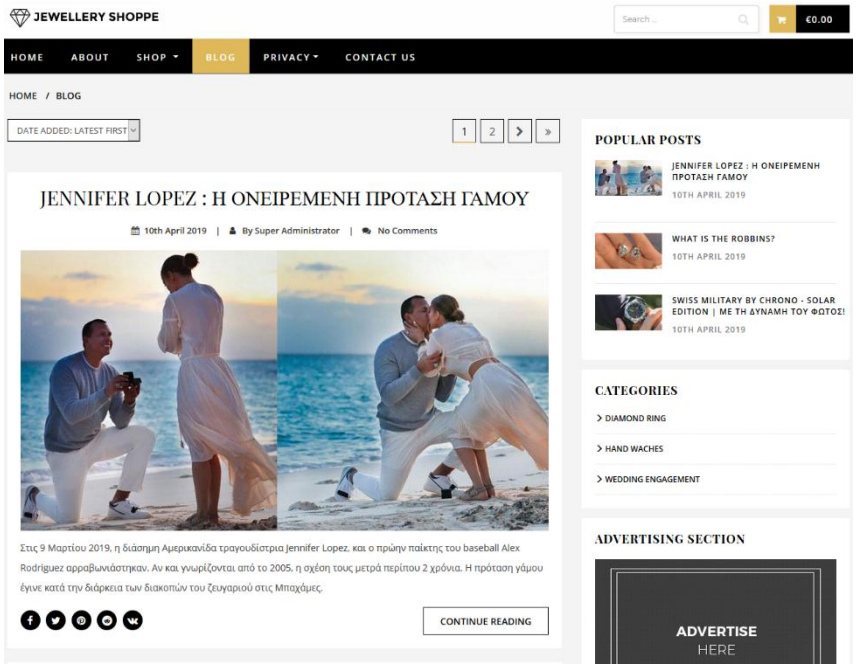
Εικόνα 92 (Home Page Site)

Στην συνέχεια θα δούμε την σελίδα με τα προϊόντα η οποία περιλαμβάνει πολλαπλά φίλτρα με αλφαβητική σειρά με τιμή από μικρότερη προς μεγαλύτερη και το αντίθετο και με ημέρα προσθήκης. Ακόμα στα αριστερά θα δούμε φίλτρα όπως με βάση το brand του προϊόντος, με βάση την κατηγορία του η την υποκατηγορία του και τέλος ένα slider για να μπορούμε να βρούμε ένα συγκεκριμένο εύρος τιμών.



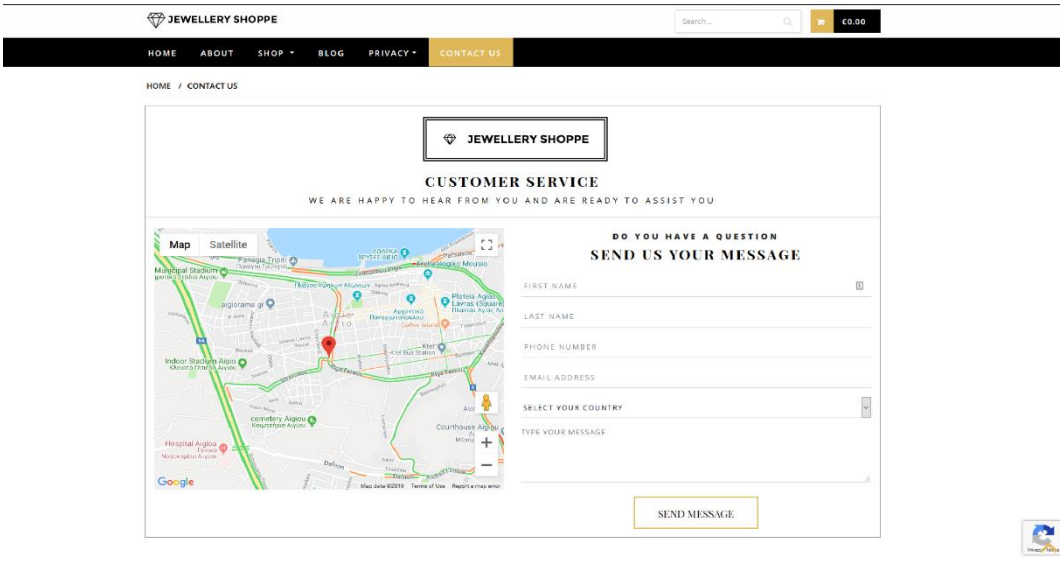
Εικόνα 93 (Shop Page Site)

Παρακάτω θα δούμε το blog τις ιστοσελίδας μας το οποίο θα έχει στην δεξιά μεριά τα πιο δημοφιλείς blog posts με βάση πόσοι χρήστες το έχουν ανοίξει και τέλος θα έχει δυο πάνελ το ένα για διαφημίσεις και το άλλο που θα έχει τα tags για τα blog posts.



Εικόνα 94 (Blog Page Site)

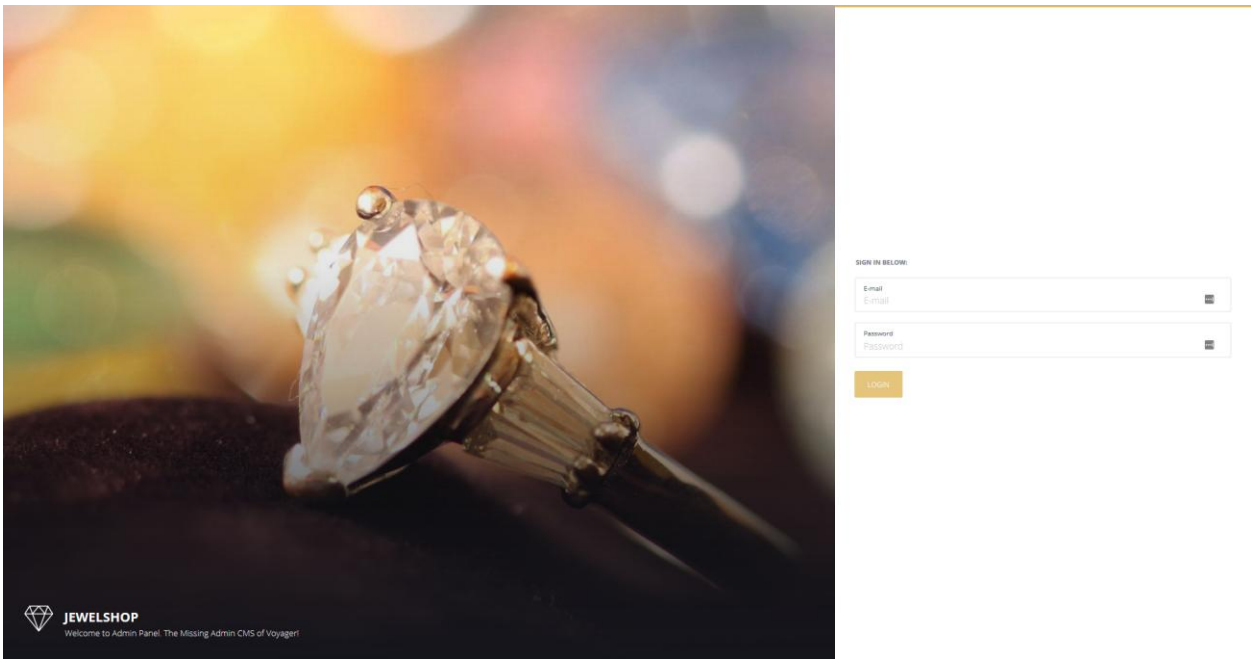
Και τέλος έχουμε την φόρμα επικοινωνίας οπού χρησιμοποιεί και captcha για να αναγνωρίζει πότε είναι άνθρωπος η ρομπότ και τέλος έχει και το Google Map για την τοποθεσία του καταστήματος.



Εικόνα 95 (Contact Page Site)

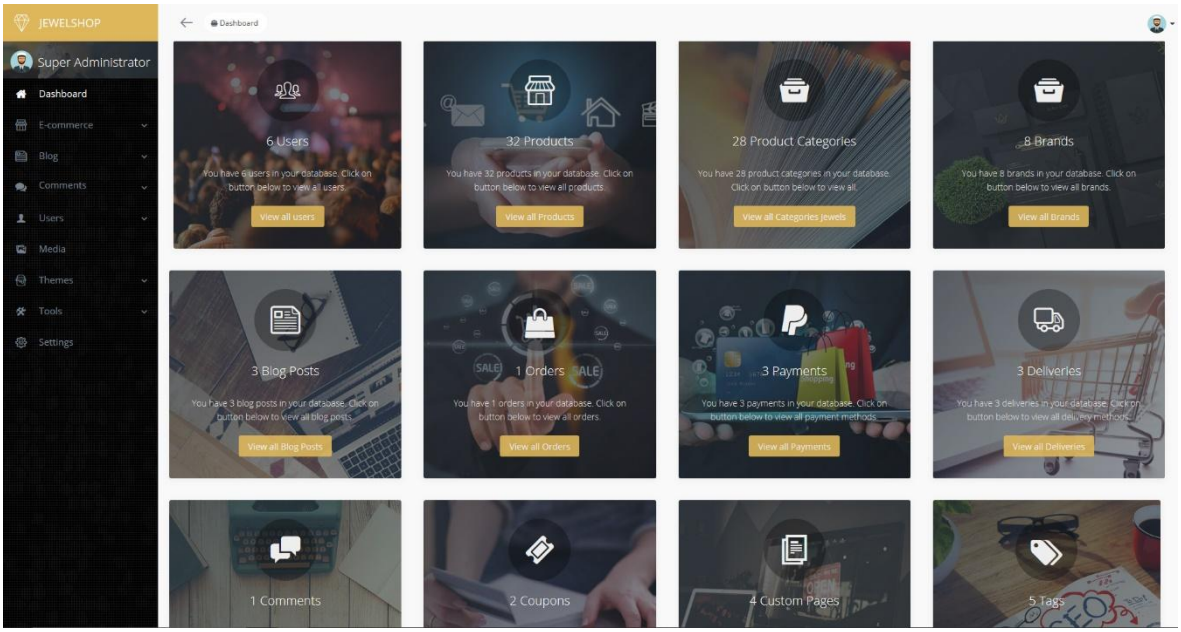
8.1.2 Περιβάλλον Διαχειριστή

Στην παρακάτω εικόνα θα σας δείξω τη σελίδα login για τους άλλους τύπους χρηστών όπως συγγράφεις, αρθρογράφους, διαχειριστές και Σούπερ Διαχειριστές.

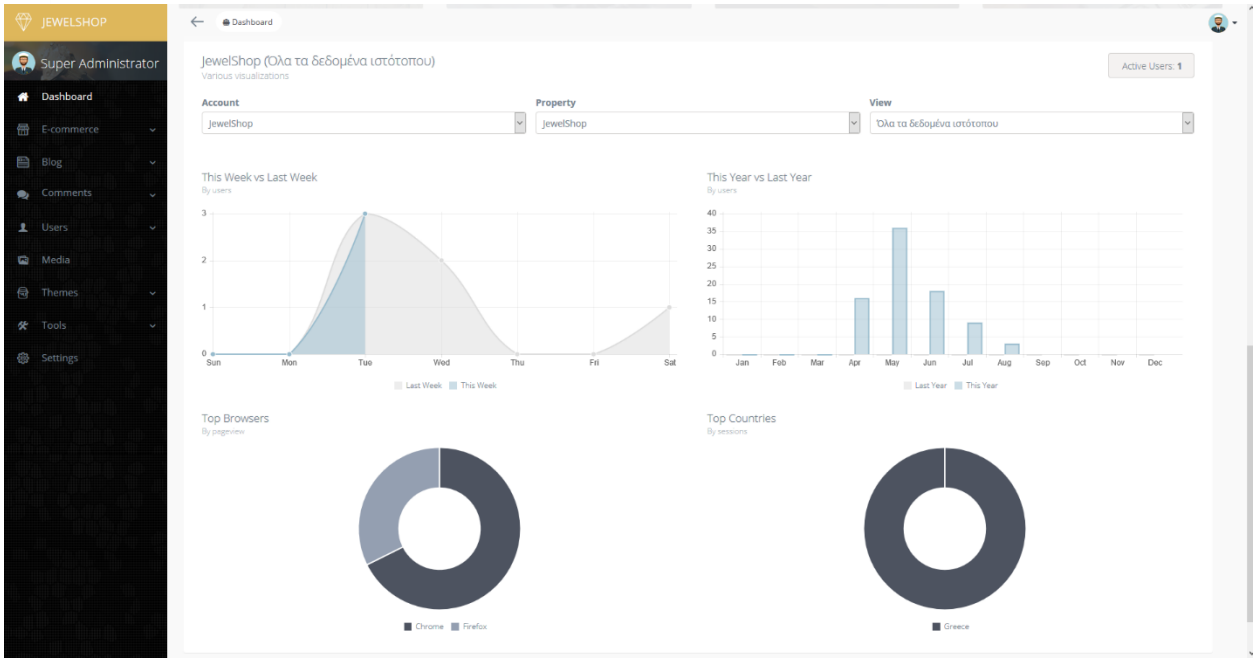


Εικόνα 96 (Login Page Admin)

Εδώ θα δούμε την αρχική σελίδα του διαχειριστή το οποίο είναι το dashboard και έχει όλα τα widgets με το τι υπάρχει στο διαχειριστικό πάνελ και πιο κάτω υπάρχουν τα Google Analytics τις ιστοσελίδας μας.

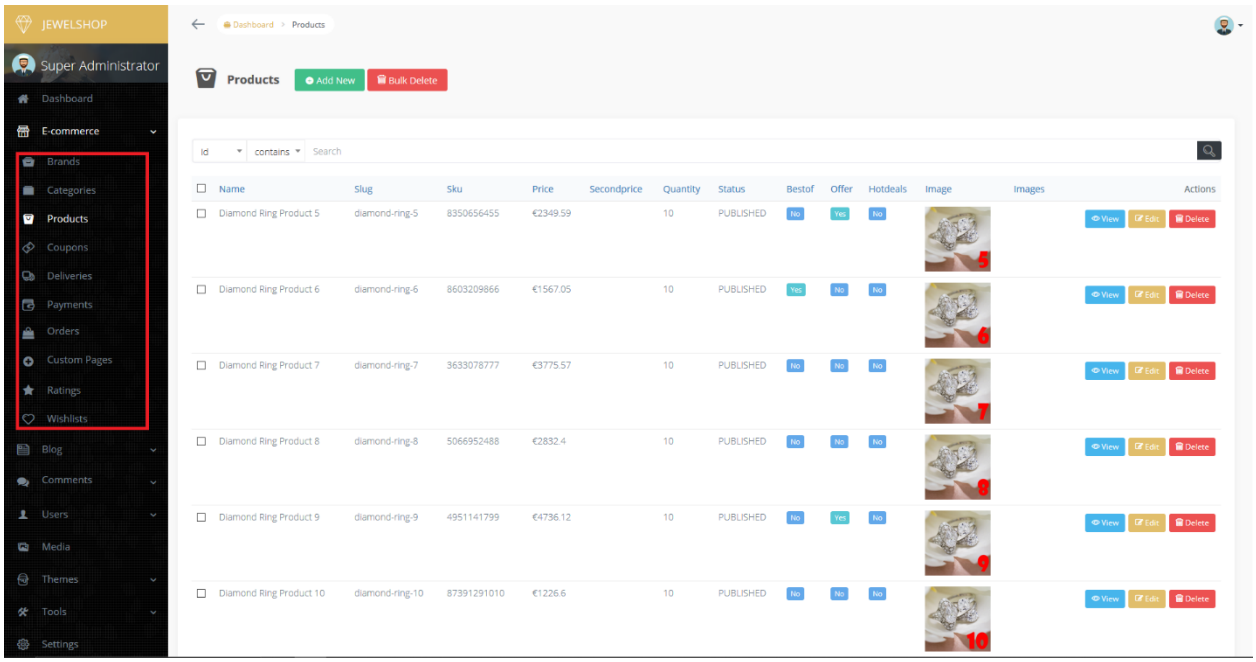


Εικόνα 97 (Admin Dashboard)



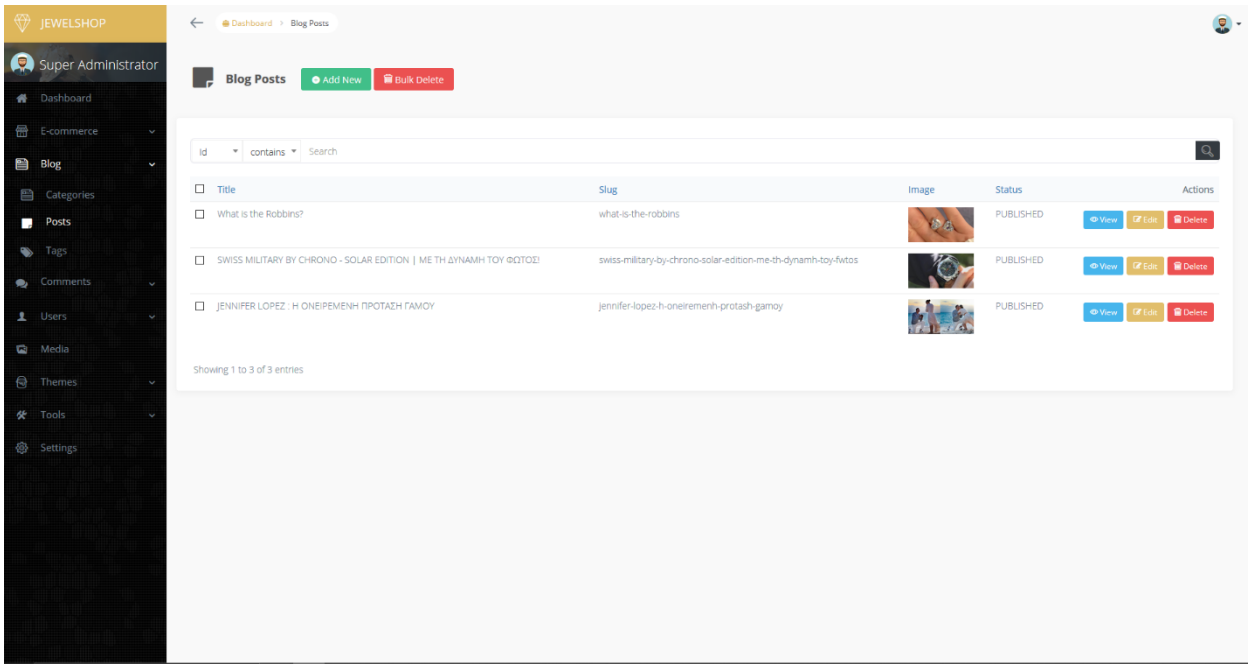
Εικόνα 98 (Admin Google Analytics)

Παρακάτω θα δείξω πως φαίνεται η σελίδα με τα περασμένα προϊόντα τα οποία μπορούμε να επεξεργαστούμε, να σβήσουμε, να προσθέσουμε νέα και να δούμε ένα - ένα σε ξεχωριστή σελίδα. Όπως θα δούμε παρακάτω για τα προϊόντα έτσι μπορούμε να κάνουμε τα ιδιά ακριβώς πράγματα για τα brands, για τις κατηγορίες, για τα κουπόνια, για τις μεθόδους πληρωμής, για τις μεθόδους μεταφοράς, για τα αγαπημένα, για τα review κάθε προϊόντος, για τις παραγγελίες και για τις custom σελίδες που θέλουμε να προσθέσουμε στην ιστοσελίδα μας.



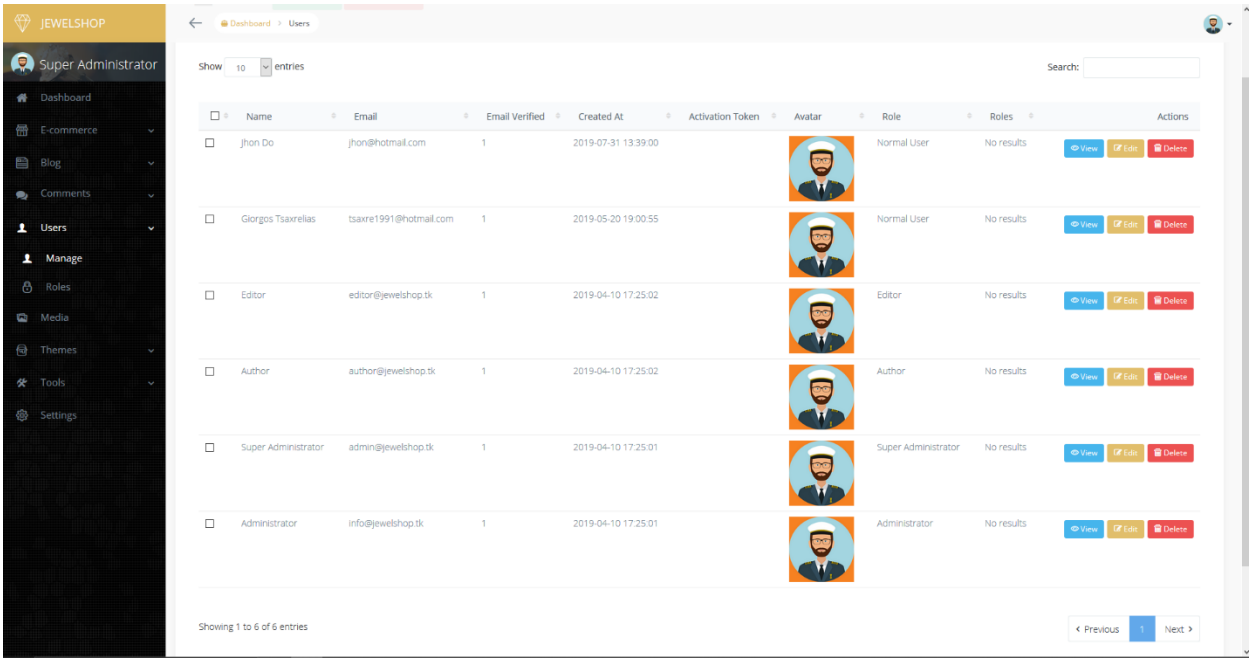
Εικόνα 99 (Admin Products View)

Όπως ακριβώς έχω αναφέρει και παραπάνω ο διαχειριστής μπορεί να κάνει τα εξής και για τα blog posts, τις κατηγορίες των blog, τα tags των blog τα comments των χρηστών και όλα τα replies που έχουν γίνει.

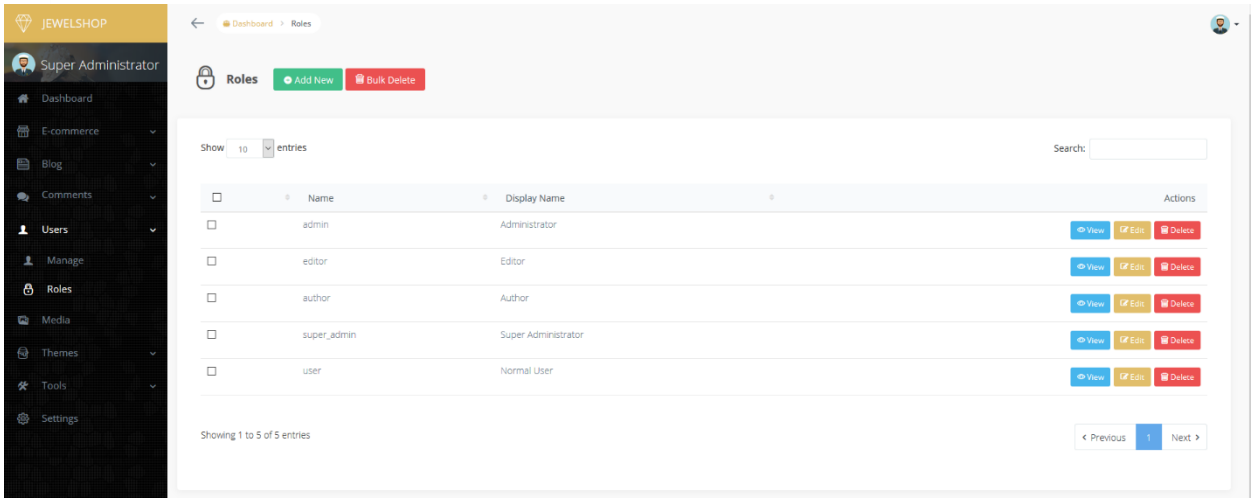


Εικόνα 100 (Blog Posts View)

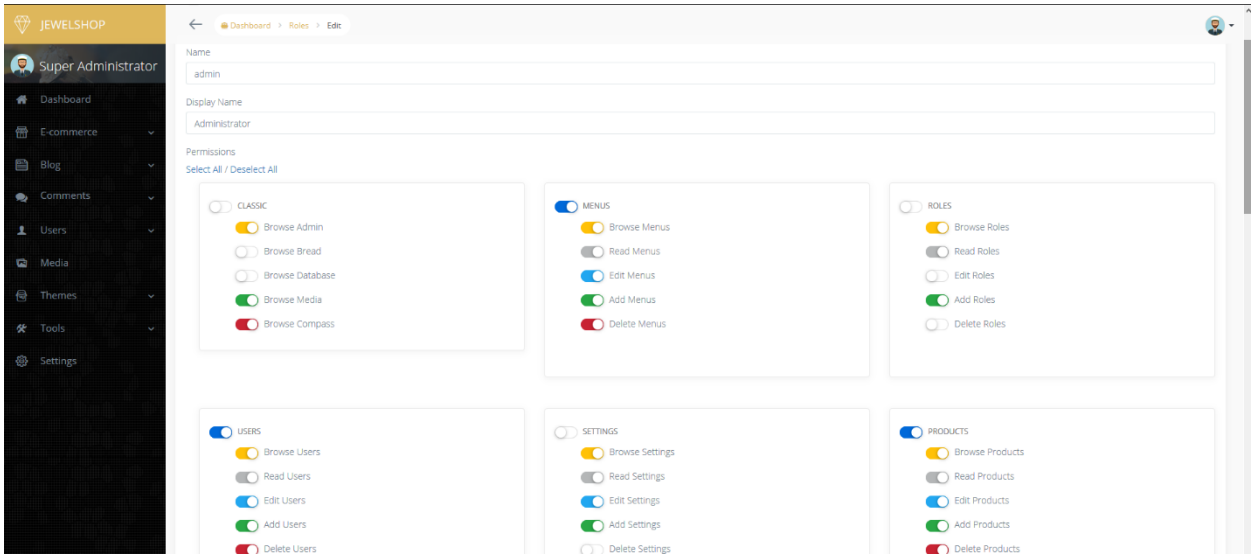
Παρακάτω θα δούμε μερικές φωτογραφίες σχετικά με τους χρήστες με τους ρόλους και με τα δικαιώματα του κάθε χρήστη στο τι μπορεί να δει στο σύστημα μας.



Εικόνα 101 (Manage Users)

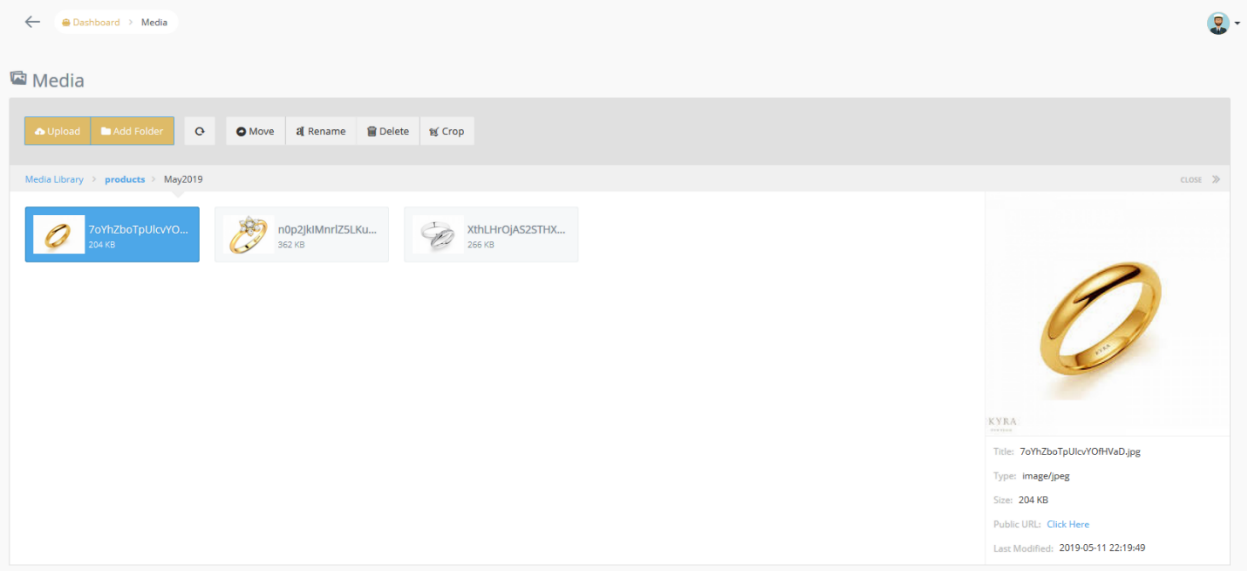


Εικόνα 102 (User Types)



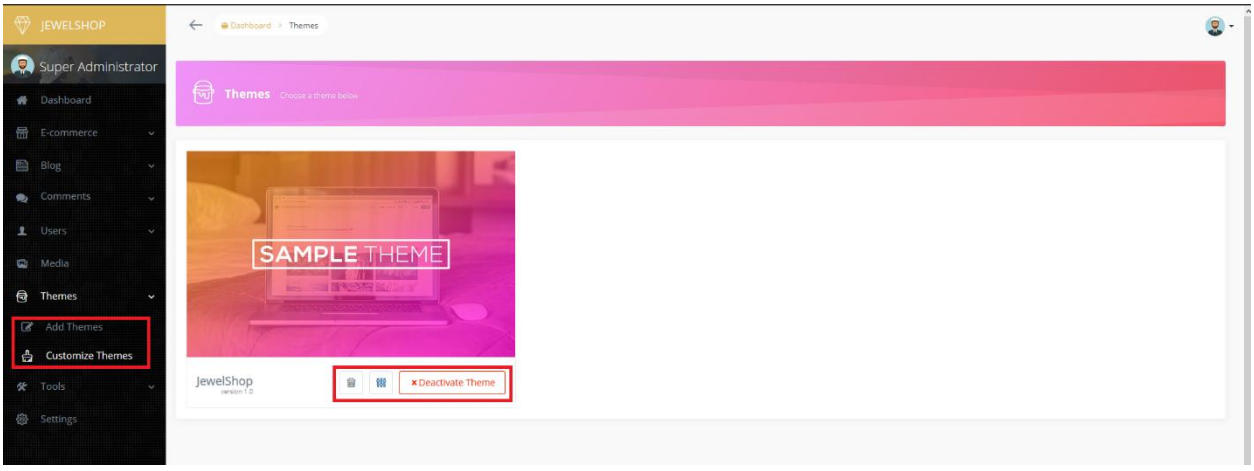
Εικόνα 103 (Admin Permissions)

Στην συνέχεια θα δούμε ότι το σύστημα μας έχει media manager που μπορούμε να διαχειριστούμε όλες τις εικόνες να αλλάξουμε όνομα να σβήσουμε εικόνα να ανεβάσουμε μια νέα ώστε να μπορέσουμε να χρησιμοποιήσουμε το URL το domain μας.

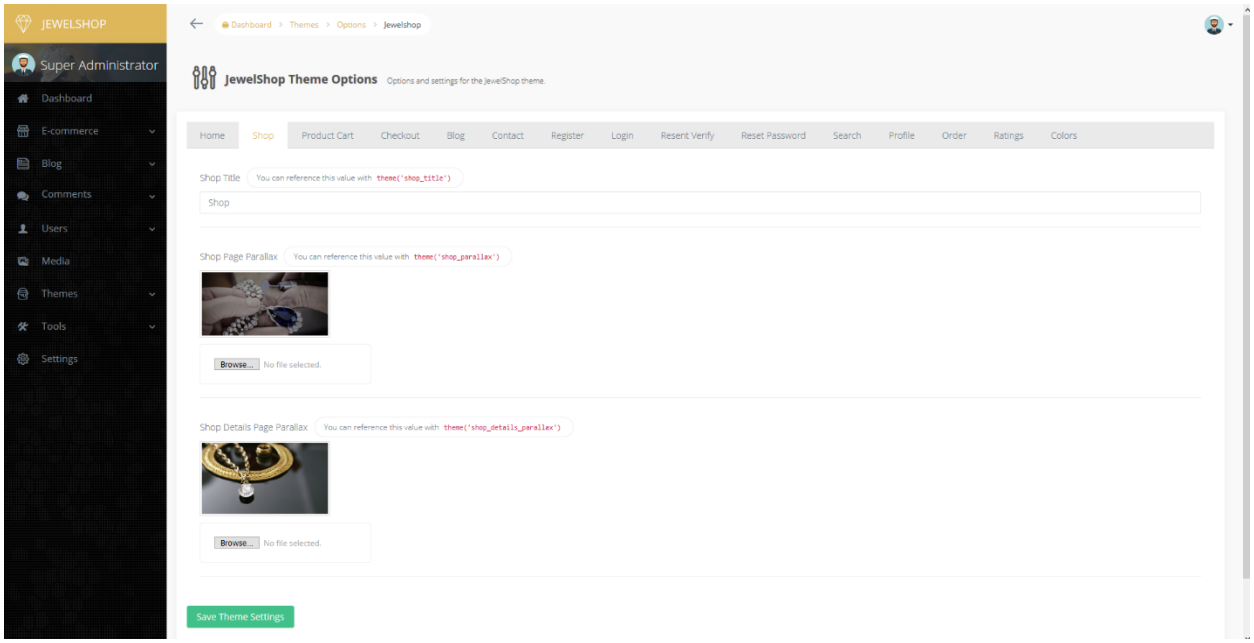


Εικόνα 104 (Admin Media Manager)

Παρακάτω έχω προσθέσει θέματα για τους τίτλους και τα χρώματα τις σελίδας τα οποία μπορούμε να τα χρησιμοποιήσουμε στην ιστοσελίδα μας. Μπορούμε να δημιουργήσομε νέα θέματα να τα έχουμε καταχωρημένα να μπορούμε να κάνουμε ενεργοποίηση όποιο από τα θέματα θέλουμε και ακόμα και να διαγράψουμε το θέμα αρκεί πάντα να υπάρχει ένα άλλο θέμα ώστε να επηρεάζει το περιεχόμενο τις ιστοσελίδας μας.

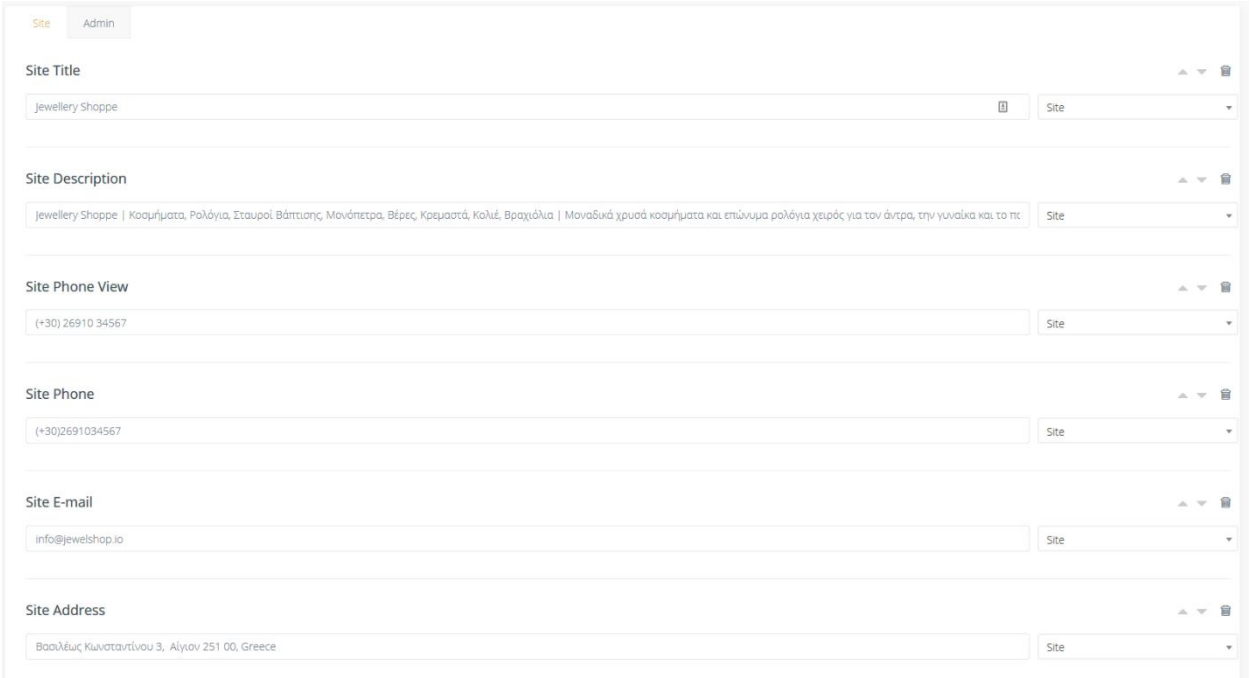


Εικόνα 105 (Customize Themes)

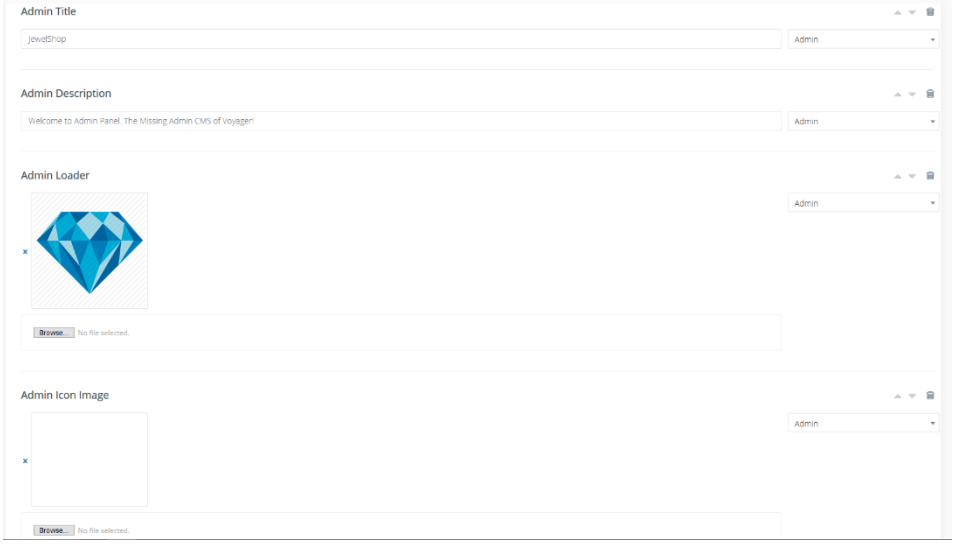


Εικόνα 106 (Change Theme Option)

Επίσης έχουμε την επιλογή να αλλάξουμε κάποιες ρυθμίσεις οι οποίες βρίσκονται στο footer τις ιστοσελίδας μας και κάποια favicon η logo του site καθώς και περίγραφες και εικόνες για το admin panel.



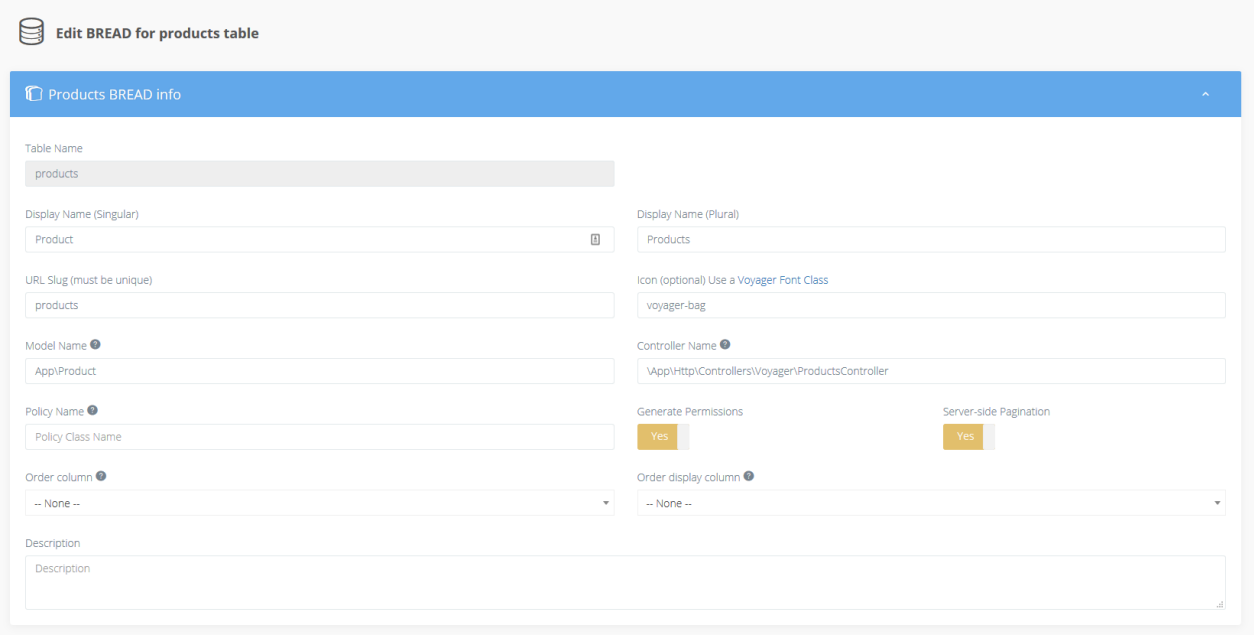
Εικόνα 107 (Site Settings)



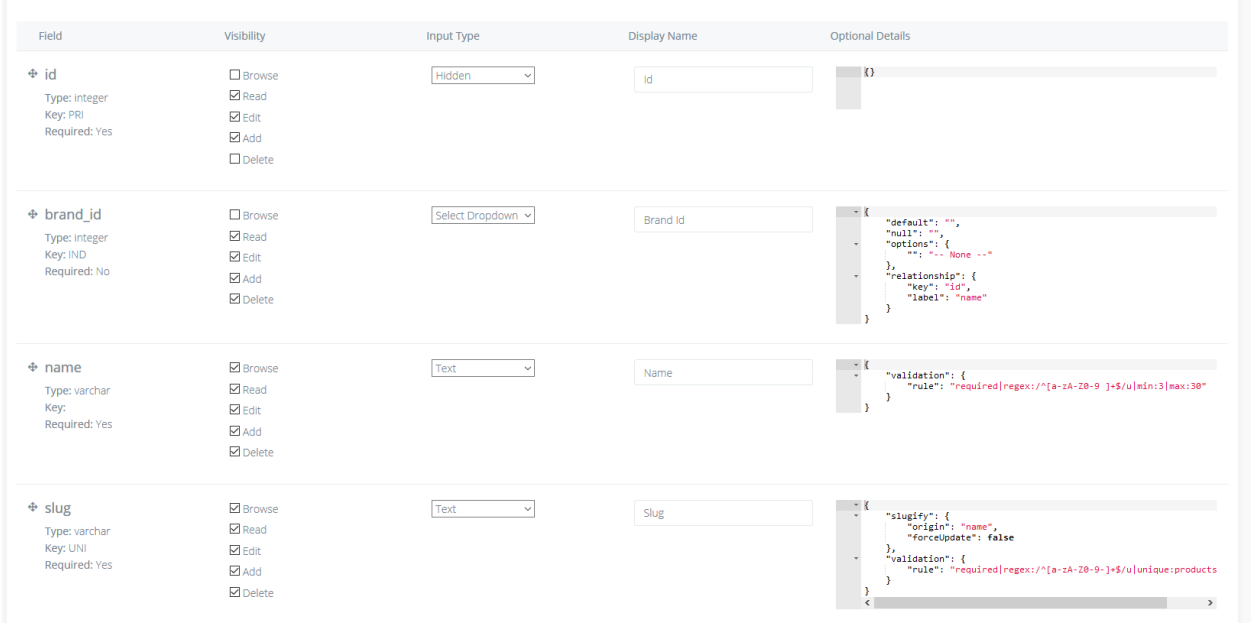
Εικόνα 108 (Admin Settings)

Τέλος έχουμε ένα dropdown ακόμα το όποιο έχει το menu builder και για τον admin και για το site που βλέπουν οι χρήστες η οι επισκέπτες τις σελίδας, ακόμα μπορούμε να πειράξουμε τα πεδία τις βάσης η να προσθέσουμε ένα νέο πίνακα και τέλος έχουμε τα BREADS (Browse,Read,Edit,Add,Delete) με τα οποία μπορούμε να πειράξουμε τα δικαιώματα ενός πίνακα με το τι θα φαίνεται και με το τι θα μπορεί να έχει δικαίωμα ο χρήστης να κάνει και τέλος μας δίνετε η δυνατότητα να βάλουμε σε μορφή JSON τα Validation του κάθε πεδίου ακόμα και να πειράξουμε τα μηνύματα λάθους που θα μας εμφανίζει.

Παρακάτω θα δούμε τα Breads για τα προϊόντα :

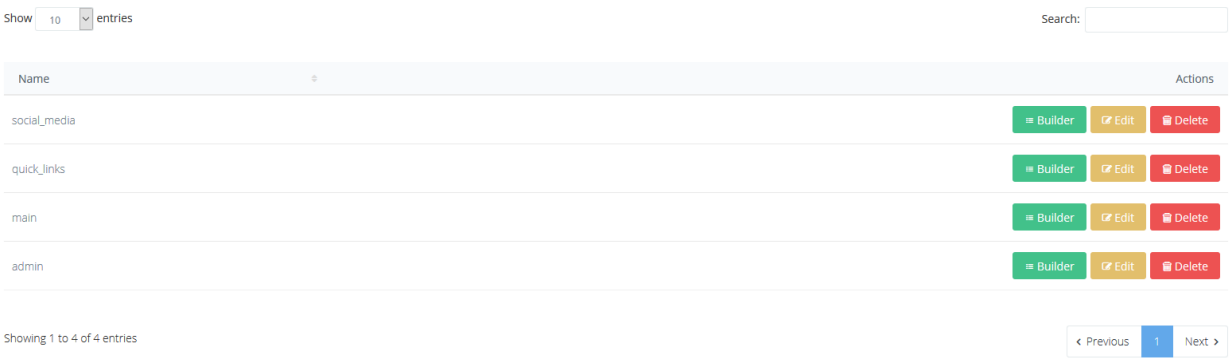


Εικόνα 109 (Products Bread Info)



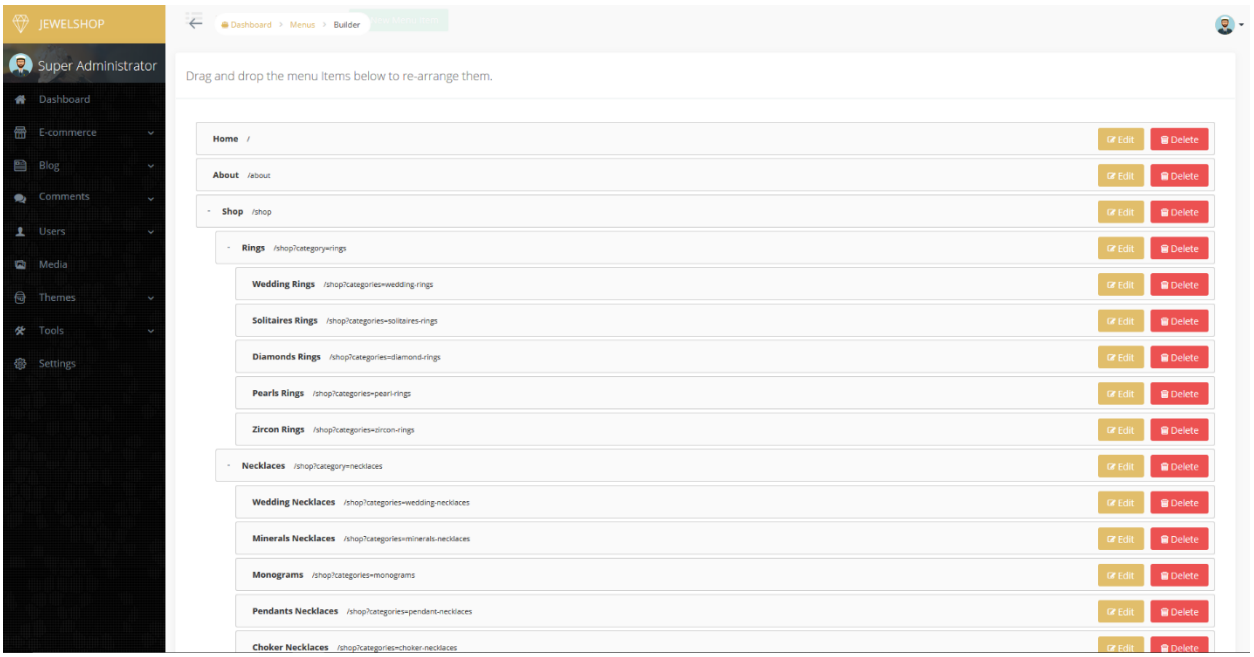
Εικόνα 110 (Products Bread Permissions)

Παρακάτω θα δούμε τον menu builder :

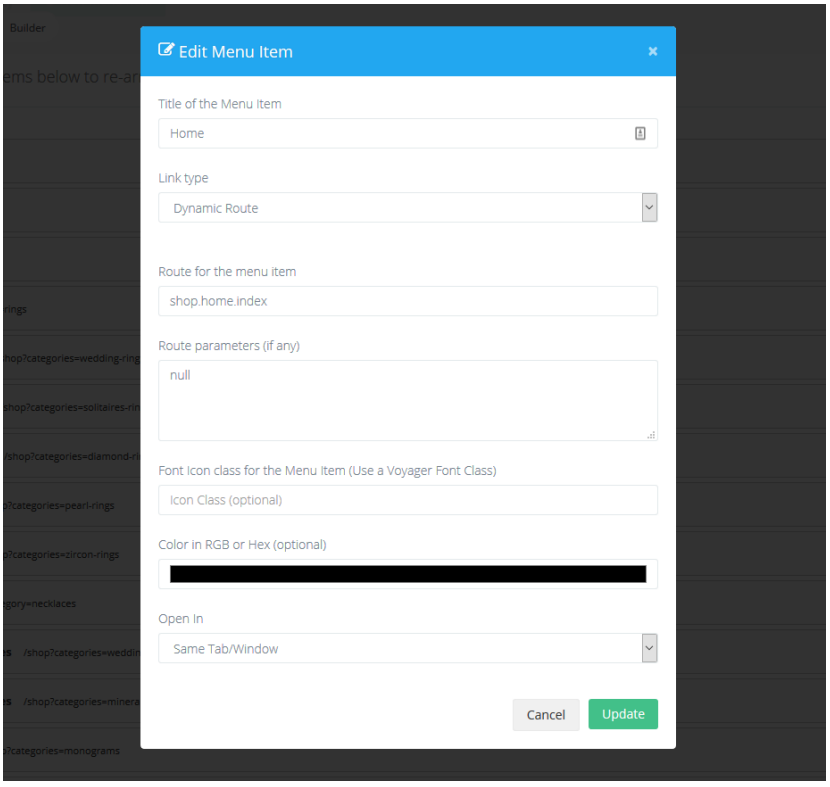


Εικόνα 111 (Menu Builder Browse)

Εδώ θα δούμε το menu του site το όποιο μπορούμε να αλλάξουμε σειρά σέρνοντας το menu και να κάνουμε και edit τα στοιχεία του όπως θα δούμε παρακάτω.

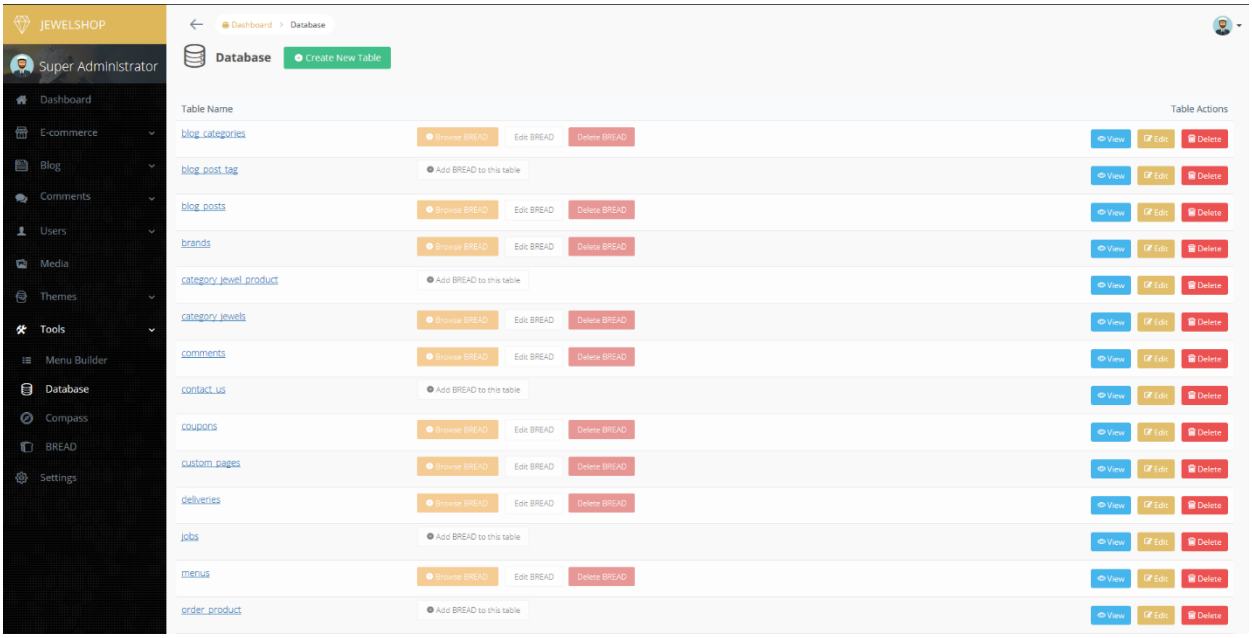


Εικόνα 112 (Draggable Site Menu)

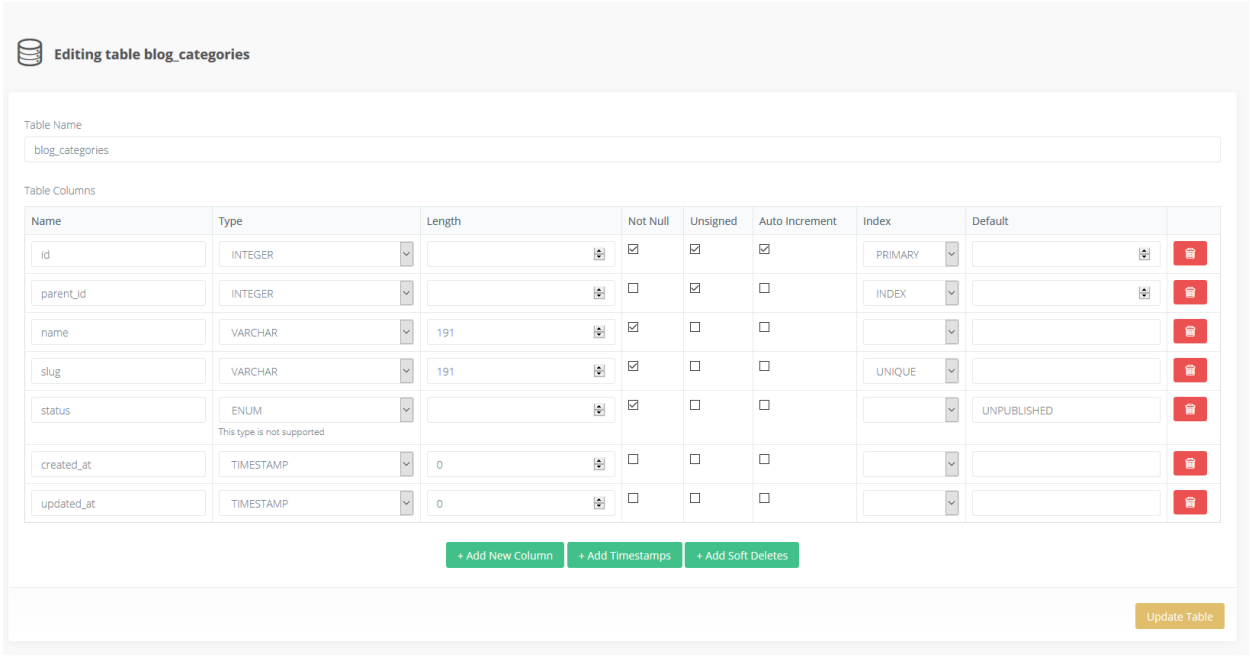


Εικόνα 113 (Edit Home Menu)

Και τέλος έχουμε την βάση δεδομένων όπου βλέπουμε όλους τους πίνακες και μπορούμε να κάνουμε edit/add/delete/browse σε οτιδήποτε έχουμε πρόσβαση λόγο δικαιωμάτων.



Εικόνα 114 (Database Tables)



Εικόνα 115 (Edit Table Blog Categories)

8.2 Κύρια Σημεία Κώδικα

8.2.1 Routes

8.2.1.1 Web Routes

Παρακάτω θα δούμε τρεις φωτογραφίες με όλα τα routes που θα χρειαστούμε για την ιστοσελίδα μας.

```php
Route::get('/','HomePageController@index')->name('shop.home.index');

Route::get('/shop','ShopController@index')->name('shop.products.index');
Route::get('/shop/{product}','ShopController@show')->name('shop.products.show');

Route::get('/cart','CartController@index')->name('shop.shopping-cart.index');
Route::post('/cart','CartController@store')->name('shop.shopping-cart.store');
Route::post('/shop','CartController@addToCart')->name('shop.shopping-cart.addToCart');
Route::delete('/cart/{product}','CartController@destroy')->name('shop.shopping-cart.destroy');
Route::patch('/cart/{product}','CartController@update')->name('shop.shopping-cart.update');
Route::post('/cart/saveForLater/{product}','CartController@saveForLater')->name('shop.shopping-cart.saveForLater');

Route::delete('/saveForLater/{product}','SaveForLaterController@destroy')->name('shop.shopping-cart.destroyForLater');
Route::post('/saveForLater/switchToCart/{product}','SaveForLaterController@switchToCart')->name('shop.shopping-cart.switchToCart');

Route::get('/checkout-user','CheckoutController@index')->name('shop.checkout.index')->middleware('auth');
Route::get('/checkout-guest','CheckoutController@guest')->name('shop.checkout.guest');
Route::post('/checkout','CheckoutController@store')->name('shop.checkout.store');
Route::get('/thankyou','ConfirmationController@index')->name('shop.checkout.confirm');

Route::post('/coupon','CouponsController@store')->name('shop.coupons.store');
Route::delete('/coupon','CouponsController@destroy')->name('shop.coupons.destroy');

Route::get('/order/{order}','OrderController@show')->name('shop.order.show');;

Route::get('/search','SearchController@show')->name('shop.search.show');

Route::get('/blog','BlogController@index')->name('shop.blog.index');
Route::get('/blog/{post}','BlogController@show')->name('shop.blog.show');
Route::post('/comments/{blog_post}','CommentController@store')->name('shop.comment.store');
Route::post('/reply/{comment_id}','CommentController@reply')->name('shop.comment.reply');
Route::delete('/delete/comment/{comment}','CommentController@destroy')->name('shop.comment.destroy');
Route::delete('/delete/reply/{reply}','CommentController@destroyReply')->name('shop.comment.destroyReply');


Route::get('/contact','ContactController@index')->name('shop.contact.index');
Route::post('/contact','ContactController@store')->name('shop.contact.store');

Route::post('/newsletter','NewsletterController@mailChimp')->name('shop.newsletter.mailChimp');
```

Εικόνα 116 (Routes 1)

```php
Route::middleware('auth')->group(function (){
    Route::post('/rating/{product}','RatingController@store')->name('shop.rating.store');

    Route::get('/customer-profile','ProfileController@index')->name('shop.profile.index');
    Route::patch('/update-password','ProfileController@updatePassword')->name('shop.profile.updatePassword');
    Route::patch('/update-details','ProfileController@updateDetails')->name('shop.profile.updateDetails');
    Route::patch('/update-addresses','ProfileController@updateAddresses')->name('shop.profile.updateAddresses');
    Route::get('/customer-order/{order}','OrderController@customerShow')->name('shop.order.customerShow');
    Route::get('/customer-rate/{rating}','RatingController@edit')->name('shop.rating.edit');
    Route::post('/customer-rate/{rating}','RatingController@update')->name('shop.rating.update');
    Route::post('/wishlist','WishlistController@store')->name('shop.wishlist.store');
    Route::delete('/wishlist/{product}','WishlistController@destroy')->name('shop.wishlist.destroy');
//    Route::get('/customer-rates','ProfileController@getRates')->name('shop.rates.getRates');
});

Route::get('empty', function(){
    Cart::destroy();
});

Route::get('emptySaves', function(){
    Cart::instance('saveForLater')->destroy();
});

//...
Route::group(['prefix' => 'admin'], function () {
    Voyager::routes();


    Route::group(['middleware' => 'admin.user'], function () {
        Route::get('themes', ['uses' => 'ThemesController@index', 'as' => 'theme.index']);
        Route::get('themes/activate/{theme}', ['uses' => 'ThemesController@activate', 'as' => 'theme.activate']);
        Route::get('themes/deactivate/{theme}', ['uses' => 'ThemesController@deactivate', 'as' => 'theme.deactivate']);
        Route::get('themes/options/{theme}', ['uses' => 'ThemesController@options', 'as' => 'theme.options']);
        Route::post('themes/options/{theme}', ['uses' => 'ThemesController@options_save', 'as' => 'theme.options.post']);
        Route::get('themes/options', function () {
            return redirect(route( name: 'theme.index'));
        });
        Route::get('themes/activate', function () {
            return redirect(route( name: 'theme.index'));
        });
        Route::get('themes/deactivate', function () {
            return redirect(route( name: 'theme.index'));
        });
        Route::delete('themes/delete', ['uses' => 'ThemesController@delete', 'as' => 'theme.delete']);
    });
```

Εικόνα 117 (Routes 2)

```
107
108
109         // Menu Routes
110         Route::group([
111             'middleware' => 'admin.user',
112             'as'         => 'voyager.menus.',
113             'prefix' => 'menus/{menu}',
114         ], function () {
115             Route::get('builder', ['uses' => 'Voyager\MenusController@builder',    'as' => 'builder']);
116         });
117
118     });
119
120     Auth::routes();
121
122     Route::get('auth/activate', 'Auth\ActivationController@activate')->name('auth.activate');
123     Route::get('auth/activate/resend', 'Auth\ActivationController@showResendForm')->name('auth.activate.showResendForm');
124     Route::post('auth/activate/resend', 'Auth\ActivationController@resend')->name('auth.activate.resend');
125
126     Route::get('{slug}', [
127         'uses' => 'PagesController@getPage'
128     ])->where('slug', '([A-Za-z0-9-]+)')->name('shop.pages.getPage');
```

Εικόνα 118 (Routes 3)

8.2.1.2 API Routes

```php
<?php

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/
$params = [
    'version' =>  env('API_VERSION'),
    'prefix' => 'api',
    'domain' => env('APP_DOMAIN'),
    'namespace' => 'App\\Http\\Controllers\\API\\v1',
];
$api = app('Dingo\Api\Routing\Router');
$api->group($params, function ($api) {
    $api->group(['prefix' => 'v1'], function ($api) {
        $api->post('login', 'AuthenticateController@login');
        $api->post('logout', 'AuthenticateController@logout');
        $api->get('token', 'AuthenticateController@getToken');
        $api->group(['middleware' => 'api.auth'], function ($api) {
            $api->get('user', 'AuthenticateController@authenticatedUser');
            $api->get('user/{user}/details', 'AuthenticateController@show');
            $api->patch('user/{user}/details/update',
'AuthenticateController@updateDetails');
            $api->patch('user/{user}/password/update',
'AuthenticateController@updatePassword');
            $api->patch('user/{user}/address/update',
'AuthenticateController@updateAddress');
            $api->get('home', 'HomePageController@index');
            $api->get('products', 'ShopController@index');
            $api->get('product/{product}', 'ShopController@show');
```

```
            $api->get('blog/{blog}', 'BlogController@show');
            $api->get('user/{user}/wishlists', 'WishlistController@index');
            $api->delete('user/{user}/wishlist', 'WishlistController@destroy');
            $api->post('user/wishlist/create', 'WishlistController@store');
            $api->get('user/{user}/orders', 'OrderController@index');
        });
    });
});
```

8.2.2 Model User

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Builder;
use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends \TCG\Voyager\Models\User
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password', 'email_verified', 'activation_token'
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    public function scopeByActivationColumns(Builder $builder, $email, $token) {
        return $builder->where('email', $email)->where('activation_token', $token);
    }

    public function orders(){
        return $this->hasMany('App\Order');
    }

    public function page()
    {
        return $this->hasOne('App\CustomPage');
```

```
    }

    public function userDetail()
    {
        return $this->hasOne('App\UserDetail');
    }

    public function ratings()
    {
        return $this->hasMany('App\Rating');
    }
    public function wishlist(){
        return $this->hasMany('App\Wishlist');
    }
}
```

Παραπάνω βλέπουμε όλα τα relationship που έχει ο user με άλλα μοντέλα όπως το Wish List, το Rating, το User Details, τα Pages και τα Orders. Καθώς βλέπουμε και πια πεδία μπορούν να γίνουν fill από το User και ποια πεδία θα είναι κρυφά για λόγους ασφάλειας.

8.2.3 Controllers

8.2.3.1 Auth Register Controller

```
<?php

namespace App\Http\Controllers\Auth;

use App\User;
use App\Http\Controllers\Controller;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Illuminate\Foundation\Auth\RegistersUsers;
use App\Events\Auth\UserActivationEmail;
use TimeHunter\LaravelGoogleReCaptchaV3\Facades\GoogleReCaptchaV3;

class RegisterController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Register Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles the registration of new users as well as their
    | validation and creation. By default this controller uses a trait to
    | provide this functionality without requiring any additional code.
    |
    */

    use RegistersUsers;
```

```php
    /**
     * Where to redirect users after registration.
     *
     * @var string
     */
    protected $redirectTo = '/';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6|confirmed',
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
            'email_verified' => false,
            'activation_token' => str_random(100)
        ]);
    }

    /**
     * Handle a registration request for the application.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
```

```
     */
    public function register(Request $request)
    {
        $this->validator($request->all())->validate();
        $captcha = GoogleReCaptchaV3::verifyResponse($request->input('g-recaptcha-
response'));

        if($captcha->isSuccess()) {
            if($captcha->getScore() >= 0.5) {
                event(new Registered($user = $this->create($request->all())));
                return $this->registered($request, $user) ?: redirect($this-
>redirectPath());
            } else {
                return redirect()->route('register')->withErrors('Robot verification
detect! Try later!');
            }

        } else {
            return redirect()->route('register')->withErrors('Robot verification
failed, please refresh and try again!');
        }
//        $this->guard()->login($user);
    }

    /**
     * The user has been registered.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  mixed  $user
     * @return mixed
     */
    protected function registered(Request $request, $user)
    {
        //send email
        event(new UserActivationEmail($user));

//        $this->guard()->logout();
        return redirect()->route('login')->with('success_message','Registered
successfully. Please check your email to activate your account!');
    }
}
```

8.2.3.2 Auth Login Controller

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;
use Illuminate\Http\Request;
use Illuminate\Validation\Rule;
```

```php
use function session;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
     *
     * @var string
     */
    protected $redirectTo = '/';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }

    /**
     * Show the application's login form.
     *
     * @return \Illuminate\Http\Response
     */
    public function showLoginForm()
    {
        session()->put('previousUrl', url()->previous());
        return view('auth.login');
    }

    public function redirectTo()
    {
        return str_replace(url('/'), '', session()->get('previousUrl', '/'));
    }

    /**
     * Validate the user login request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return void
```

```
     */
    protected function validateLogin(Request $request)
    {
        $this->validate($request, [
            $this->username() => [
                'required','string',
                Rule::exists('users')->where(function($query) {
                    $query->where('email_verified',true);
                })
            ],
            'password' => 'required|string',
        ], $this->validationError());
    }

    /**
     * Get the validation error for login
     *
     * @return array
     */
    public function validationError() {
        return [
            $this->username() . '.exists' => 'The selected E-mail is invalid or you
need to activate your account!'
        ];
    }
}
```

8.2.3.3 Auth Activation Controller

```
<?php

namespace App\Http\Controllers\Auth;


use App\User;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Events\Auth\UserActivationEmail;
use Auth;

class ActivationController extends Controller
{
    public function activate(Request $request) {
        $user = User::byActivationColumns($request->email, $request->token)-
>firstOrFail();
        $user->update([
            'email_verified' => true,
            'activation_token' => null
        ]);

        Auth::loginUsingId($user->id);

        return redirect()->route('shop.home.index')-
```

```
>with('success_message','Successfully activated! You are sign in!');
    }

    public function showResendForm() {
        return view('auth.passwords.resend');
    }

    public function resend(Request $request) {
        $this->validateResendRequest($request);

        $user = User::where('email', $request->email)->first();

        if($user->activation_token == null && $user->email_verified == true) {
            return redirect()->route('login')->with('success_message','That account
is activated!');
        }

        event(new UserActivationEmail($user));

        return redirect()->route('login')->with('success_message','Email activation
has been resend!');
    }

    protected function validateResendRequest(Request $request) {
        $this->validate($request , [
           'email' => 'required|email|exists:users,email',
        ], [
            'email.exists' => 'This email is not exists. Please check your email'
        ]);
    }
}
```

8.2.3.4 Shop Controller

```
<?php

namespace App\Http\Controllers;

use App\Brand;
use App\CategoryJewel;
use App\Product;
use App\Rating;
use App\User;
use Illuminate\Support\Facades\Auth;
use function request;

class ShopController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
```

```
    public function index()
    {
        $specialOffers = Product::where('status', 'like', 'PUBLISHED')-
>where('offer', true)->inRandomOrder()->get();
        $hotDeals = Product::where('status', 'like', 'PUBLISHED')->where('hotdeals',
true)->inRandomOrder()->get();
        $allBrands = Brand::where('status', 'like', 'PUBLISHED')-
>orderBy('name','ASC')->with('products')->whereHas('products', function ($query) {
            $query->where('status', 'like', 'PUBLISHED');
        })->get();

        $allCategories = CategoryJewel::where('status', 'like', 'PUBLISHED')-
>orderBy('name','ASC')->with('products')->whereHas('products', function ($query) {
            $query->where('status', 'like', 'PUBLISHED');
        })->get();


        if (request()->category) {

            // show product by category
            $category = CategoryJewel::where('slug',request()->category)-
>firstOrFail();
            $id = $category->id;
            $products = Product::with('categoriesJewels')-
>whereHas('categoriesJewels', function($query) use($id) {
                $query->where('parent_id', $id);
            });

            $maxPrice = $products->max('price');
            $minPrice = $products->min('price');
            $productsPagination = $products->count();

        } else {
            $products = Product::where('status', 'like', 'PUBLISHED')->take(16);
            $maxPrice = $products->max('price');
            $minPrice = $products->min('price');
            $productsPagination = $products->count();
        }

        if(request()->filter == "range"){
            $products = $products->where('price', '>=', request()->min * 100)
                ->where('price', '<', request()->max * 100);
        }

        if(request()->brands && request()->brands != ""){
            $products = Product::where('status', 'like', 'PUBLISHED')->with('brand')-
>whereHas('brand', function ($query) {
                $query->where('status', 'like', 'PUBLISHED')-
>whereIn('slug',explode(' ', request()->brands));
            });
            $maxPrice = $products->max('price');
            $minPrice = $products->min('price');

            //add request brands to popular by visit slug
            $popularBrands = Brand::where(function($query) {
```

```
                $query->whereIn('slug',explode(' ', request()->brands));
            })->get();
            foreach ($popularBrands as $popularBrand) {
                $popularBrand->visit();
            }
        }
        if(request()->categories && request()->categories != ""){
            $products = Product::where('status', 'like', 'PUBLISHED')-
>with('categoriesJewels')->whereHas('categoriesJewels', function ($query) {
                $query->where('status', 'like', 'PUBLISHED')-
>whereIn('slug',explode(' ', request()->categories));
            });
            $maxPrice = $products->max('price');
            $minPrice = $products->min('price');
        }
        if(request()->brands && request()->brands != "" && request()->categories &&
request()->categories != ""){
            $products = Product::where('status', 'like', 'PUBLISHED')-
>with('categoriesJewels')->whereHas('categoriesJewels', function ($query) {
                $query->whereIn('slug',explode(' ', request()->categories));
            })->with('brand')->whereHas('brand', function ($query) {
                $query->whereIn('slug',explode(' ', request()->brands));
            });

            //add request brands to popular by visit slug
            $popularBrands = Brand::where(function($query) {
                $query->whereIn('slug',explode(' ', request()->brands));
            })->get();
            foreach ($popularBrands as $popularBrand) {
                $popularBrand->visit();
            }
        }

        if(request()->sort == "low_high"){
            $products = $products->orderBy('price')->paginate(Product::PAGINATION);
        } else if(request()->sort == "high_low") {
            $products = $products->orderBy('price', 'desc')-
>paginate(Product::PAGINATION);
        } else if(request()->sort == "new") {
            $products = $products->orderBy('id', 'desc')-
>paginate(Product::PAGINATION);
        } else if(request()->sort == "a_z") {
            $products = $products->orderBy('name')->paginate(Product::PAGINATION);
        } else if(request()->sort == "z_a") {
            $products = $products->orderBy('name', 'desc')-
>paginate(Product::PAGINATION);
        } else {
            $products = $products->orderBy('id', 'desc')-
>paginate(Product::PAGINATION); //->onEachSide($onside)
        }


        return view('shop.products.main')->with([
            'products' => $products,
            'productsPagination' => $productsPagination,
```

```
            'specialOffers' => $specialOffers,
            'hotDeals' => $hotDeals,
            'maxPrice' => $maxPrice,
            'minPrice' => $minPrice,
            'allBrands' => $allBrands,
            'allCategories' => $allCategories,
        ]);
    }

    /**
     * Display the specified resource.
     *
     * @param  string  $slug
     * @return \Illuminate\Http\Response
     */
    public function show($slug)
    {
        $product = Product::where('status', 'like', 'PUBLISHED')-
>where('slug',$slug)->firstOrFail();
        $mightAlsoLike = Product::where('status', 'like', 'PUBLISHED')-
>where('slug','!=',$slug)->inRandomOrder()->take(20)->get();

        if(Auth::user()){
            $user = Auth::user()->id;
        } else {
            $user = 0;
        }
        $rate = Rating::where('user_id',$user)->where('product_id',$product->id)-
>first();

        //appear badge for quantity of product
        $stockLevel = getStockLevel($product->quantity);

        // add in visit table for products
        $product->visit();

        return view('shop.products.details')->with([
            'product' => $product,
            'mightAlsoLike' => $mightAlsoLike,
            'stockLevel' => $stockLevel,
            'rate' => $rate,
        ]);
    }
}
```

8.2.3.5 Cart Controller

```
<?php

namespace App\Http\Controllers;

use App\Product;
use Gloudemans\Shoppingcart\Facades\Cart;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
```

```
class CartController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $mightAlsoLike = Product::where('status', 'like', 'PUBLISHED')-
>inRandomOrder()->take(4)->get();
        return view('shop.shopping-cart.main')->with([
            'mightAlsoLike' => $mightAlsoLike,
             'discount' => getNumbers()->get('discount'),
            'newSubTotal' => getNumbers()->get('newSubTotal'),
            'newTax' => getNumbers()->get('newTax'),
            'newTotal' => getNumbers()->get('newTotal'),
        ]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $duplicates = Cart::search(function($cartItem,$rowId) use ($request){
            return $cartItem->id === $request->id;
        });
        if ($duplicates->isNotEmpty()){
            return redirect()->route('shop.shopping-cart.index')-
>with('warning_message','Item is already in your cart!!');
        }
        Cart::add($request->id, $request->name, 1, $request->price)
            ->associate('App\Product');
        return redirect()->route('shop.shopping-cart.index')-
>with('success_message','Item was added to your shopping cart!');
    }

    /**
     * Store a newly created resource in storage.
     *
```

```
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function addToCart(Request $request)
    {
        $duplicates = Cart::search(function($cartItem,$rowId) use ($request){
            return $cartItem->id === $request->id;
        });
        if ($duplicates->isNotEmpty()){
            return redirect()->route('shop.products.index')-
>with('warning_message','Item is already in your cart!!');
        }
        Cart::add($request->id, $request->name, 1, $request->price)
            ->associate('App\Product');
        return redirect()->route('shop.products.index')->with('success_message','Item
was added to your shopping cart!');
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     */
    public function show($id)
    {
        //
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     */
    public function edit($id)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        $validator = Validator::make($request->all(),[
            'quantity' => 'required|numeric|gt:0'
        ]);
        if($validator->fails()){
            session()->flash('errors', collect(['This quantity must be greater than
one!']));
            return response()->json(['success' => false], 400);
        }
```

```
        if($request->productQuantity){
            if($request->quantity > $request->productQuantity){
                session()->flash('errors', collect(['We dont have enough items to
add! Please contact with us.']));
                return response()->json(['success' => false], 400);
            }
        }

        Cart::update($id, $request->quantity);

        session()->flash('success_message', 'Quantity was updated successfully!');
        return response()->json(['success' => true]);
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        Cart::remove($id);
        return back()->with('success_message','Product has been removed!!');
    }

    /**
     * Shopping cart save for later
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function saveForLater($id)
    {
        $item = Cart::instance('default')->get($id);


        $duplicates = Cart::instance('saveForLater')->search(function($cartItem,
$rowId) use ($id){
            return $rowId === $id;
        });
        if ($duplicates->isNotEmpty()){
            return redirect()->route('shop.shopping-cart.index')-
>with('warning_message','Item is already Saved For Later!!');
        }else {

            Cart::instance('default')->remove($id);
            Cart::instance('saveForLater')->add($item->id, $item->name, 1, $item-
>price)
                ->associate('App\Product');

            return redirect()->route('shop.shopping-cart.index')-
>with('success_message', 'Item has moved to Saved for Later!!');
        }
```

```
    }
}
```

8.2.3.6 Checkout Controller

```php
<?php

namespace App\Http\Controllers;

use App\Delivery;
use App\Events\Order\UserOrderPlace;
use App\Http\Requests\CheckoutRequest;
use App\Order;
use App\OrderProduct;
use App\Payment;
use App\Product;
use Cartalyst\Stripe\Exception\CardErrorException;
use Cartalyst\Stripe\Laravel\Facades\Stripe;
use Gloudemans\Shoppingcart\Facades\Cart;
use Illuminate\Support\Str;
use function request;
use function sha1;


class CheckoutController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $deliveries = Delivery::where('status', 'like', 'PUBLISHED')->get();
        $payments = Payment::where('status', 'like', 'PUBLISHED')->get();

        if(Cart::instance('default')->count() == 0) {
            return redirect()->route('shop.products.index');
        }

        return view('shop.checkout.main')->with([
            'discount' => getNumbers()->get('discount'),
            'newSubTotal' => getNumbers()->get('newSubTotal'),
            'newTax' => getNumbers()->get('newTax'),
            'newTotal' => getNumbers()->get('newTotal'),
            'deliveries' => $deliveries,
            'payments' => $payments,
        ]);
    }

    /**
     * Display a listing of the resource.
     *
```

```
     * @return \Illuminate\Http\Response
     */
    public function guest()
    {
        $deliveries = Delivery::where('status', 'like', 'PUBLISHED')->get();
        $payments = Payment::where('status', 'like', 'PUBLISHED')->get();

        if(Cart::instance('default')->count() == 0) {
            return redirect()->route('shop.products.index');
        }

        if(auth()->user() && request()->is('checkout-guest')) {
            return redirect()->route('shop.checkout.index');
        }

        return view('shop.checkout.main')->with([
            'discount' => getNumbers()->get('discount'),
            'newSubTotal' => getNumbers()->get('newSubTotal'),
            'newTax' => getNumbers()->get('newTax'),
            'newTotal' => getNumbers()->get('newTotal'),
            'deliveries' => $deliveries,
            'payments' => $payments,
        ]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \App\Http\Requests\CheckoutRequest $request
     *
     * @return \Illuminate\Http\Response
     */
    public function store(CheckoutRequest $request)
    {
        //method to check before purchase process start if we have multi user and one
of them is faster than other
        if($this->productsAreNoLongerAvailable()){
            return back()->withErrors($request->errors());
        }

        $content = Cart::content()->map(function($item){
            return 'Product Name: '.$item->model->slug.', Quantity: '.$item->qty;
        })->values()->toJson();
        $discount = [
            'Name' => session()->get('coupon')['name'],
            'Discount' =>  number_format(session()->get('coupon')['discount'] /
100,2,'.',',')
        ];
        if(!empty($request->holder_name)) {
            try {
                $charge = Stripe::charges()->create([
                    'amount' => getNumbers()->get('newTotal') / 100,
                    'currency' => 'EUR',
                    'source' => $request->stripeToken,
                    'description' => 'Order',
```

```
                    'receipt_email' => $request->email,
                    'metadata' => [
                        'contents' => $content,
                        'discount' => collect($discount)->toJson(),
                        'quantity' => Cart::instance('default')->count(),
                    ],
                ]);

                $this->addToOrdersTables($request,null);

                //decrease quantity from all items in cart
                $this->decreaseQuantities();

                //SUCCESSFUL
                Cart::instance('default')->destroy();
                session()->forget('coupon');

                return redirect()->route('shop.checkout.confirm')-
>with('success_message','Thank you! Your payment has been successfully accepted!');
            } catch (CardErrorException $e) {
                $this->addToOrdersTables($request, $e->getMessage());
                return back()->withErrors('Error! ' . $e->getMessage());
            }
        } else {
            $this->addToOrdersTables($request,null);

            //decrease quantity from all items in cart
            $this->decreaseQuantities();

            //SUCCESSFUL
            Cart::instance('default')->destroy();
            session()->forget('coupon');

            return redirect()->route('shop.checkout.confirm')-
>with('success_message','Thank you! Your payment has been successfully accepted!');
        }
    }

    protected function addToOrdersTables($request, $error){
        $random_id = str::random(60);
        $random_id .= microtime();
        $hash_id = sha1($random_id);

        //INSERT INTO ORDERS TABLE
        $order = Order::create([
            'user_id' => auth()->user() ? auth()->user()->id : null,
            'unique_id' => $hash_id,

            'billing_fname' => $request->fname,
            'billing_lname' => $request->lname,
            'billing_address' => $request->address,
            'billing_housenumber' => $request->hnumber,
            'billing_locality' => $request->locality,
            'billing_email' => $request->email,
            'billing_city' => $request->city,
```

```
            'billing_phone' => $request->phone,
            'billing_country' => $request->country,
            'billing_postalcode' => $request->zip_code,

            'different_shipping_address' => !empty($request->dif_shipping) ? true :
false,

            'second_billing_fname' => !empty($request->fname_sec) ? $request-
>fname_sec : null,
            'second_billing_lname' => !empty($request->lname_sec) ? $request-
>lname_sec : null,
            'second_billing_address' => !empty($request->lname_sec) ? $request-
>lname_sec : null,
            'second_billing_housenumber' => !empty($request->hnumber_sec) ? $request-
>hnumber_sec : null,
            'second_billing_locality' => !empty($request->locality_sec) ? $request-
>locality_sec : null,
            'second_billing_email' => !empty($request->email_sec) ? $request-
>email_sec : null,
            'second_billing_city' => !empty($request->city_sec) ? $request->city_sec
: null,
            'second_billing_phone' => !empty($request->phone_sec) ? $request-
>phone_sec : null,
            'second_billing_country' => !empty($request->country_sec) ? $request-
>country_sec : null,
            'second_billing_postalcode' => !empty($request->zipcode_sec) ? $request-
>zipcode_sec : null,

            'billing_discount' => getNumbers()->get('discount'),
            'billing_discount_code' => getNumbers()->get('code'),

            'billing_subtotal' => getNumbers()->get('newSubTotal'),
            'billing_tax' => getNumbers()->get('newTax'),
            'billing_total' => getNumbers()->get('newTotal'),

            'delivery_gateway' => $request->delivery,
            'payment_gateway' => $request->card,
            'name_on_card' => !empty($request->holder_name) ? $request->holder_name :
null,
            'instructions' => !empty($request->instructions) ? $request->instructions
: null,

            'error' => $error
        ]);

        //INSERT INTO ORDER_PRODUCT TABLE
        foreach(Cart::content() as $item){
            OrderProduct::create([
                'order_id' => $order->id,
                'product_id' => $item->model->id,
                'quantity' => $item->qty
            ]);
        }

        event(new UserOrderPlace($order));
```

```
    }

    protected function decreaseQuantities() {
        foreach (Cart::content() as $item){
            $product = Product::find($item->model->id);
            $product->update(['quantity' => $product->quantity - $item->qty]);
        }
    }

    protected function productsAreNoLongerAvailable() {
        foreach (Cart::content() as $item){
            $product = Product::find($item->model->id);
            if($product->quantity < $item->qty){
                return true;
            }
        }
        return false;
    }
}
```

8.2.3.7 Contact Controller

```
<?php

namespace App\Http\Controllers;

use App\ContactUs;
use App\Events\Contact\ContactEvent;
use App\Http\Requests\ContactRequest;
use Mapper;

class ContactController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        Mapper::map(38.24808000000, 22.08052000000, ['markers' => ['title' => 'Jewel
Shop Aigio', 'animation' => 'DROP'], 'overlay' => 'TRAFFIC']);
        return view('shop.contact.main');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Requests\ContactRequest $request
     * @return \Illuminate\Http\Response
     */
    public function store(ContactRequest $request)
    {
```

```
        $contact = ContactUs::create([
            'fname' => $request->fname,
            'lname' => $request->lname,
            'phone' => $request->phone,
            'email' => $request->email,
            'country' => $request->country,
            'message' => $request->message,
        ]);

        event(new ContactEvent($contact));

        return redirect()->route('shop.contact.index')->with('success_message','Thank
you! Your message has been send successfully! <br>We will contact as soon as
possible!');
    }
}
```

8.2.3.8 Profile Controller

```
<?php

namespace App\Http\Controllers;

use App\User;
use App\UserDetail;
use App\Wishlist;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use App\Http\Requests\UpdateProfileAddressesRequest;
use App\Http\Requests\UpdateProfileDetailsRequest;
use App\Http\Requests\UpdateProfilePasswordRequest;

class ProfileController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $user = User::where('id',auth()->id())->with('orders')->with('userDetail')-
>first();
        $rates = $user->ratings()->with('product')->latest()->paginate(5);
        $wishlists = Wishlist::where("user_id", "=", $user->id)->with('product')-
>orderby('id', 'desc')->paginate(5);

        return view('shop.profile.main')->with([
            'user' => $user,
            'rates' => $rates,
            'wishlists' => $wishlists,
        ]);
    }
```

```
    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \App\Http\Requests\UpdateProfilePasswordRequest  $request
     * @return \Illuminate\Http\Response
     */
    public function updatePassword(UpdateProfilePasswordRequest $request)
    {
            $user = User::where('id',auth()->id())->first();

            if(Hash::check($request->old_password,$user->password)){
                $user->password = Hash::make($request->password);
                $user->save();

                //logout other user with old password after change
//                $user->logoutOtherDevices($user->password);
                Auth::login($user);
                return back()->with('success_message','Password Update
Successfully!');
            } else {
                return back()->withErrors('Old Password is Wrong!');
            }
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \App\Http\Requests\UpdateProfileDetailsRequest  $request
     * @return \Illuminate\Http\Response
     */
    public function updateDetails(UpdateProfileDetailsRequest $request)
    {
        $user = User::where('id',auth()->id())->first();

        $user->name = $request->name ? $request->name : $user->name;
        $user->email = $request->email ? $request->email : $user->email;
        $user->save();

        if($user->userDetail()->first()) {
            $details = new userDetail;

            $details->phone = $request->phone ? $request->phone : $details->phone;
```

```
            $details->company = $request->company ? $request->company : $details->company;

            $user->userDetail()->update([
                'phone' => $details->phone,
                'company' => $details->company
            ]);
        } else {
            $details = new userDetail;

            $details->phone = $request->phone ? $request->phone : $details->phone;
            $details->company = $request->company ? $request->company : $details->company;
            $user->userDetail()->save($details);
        }
        return back()->with('success_message','Profile Details Update Successfully!');

    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \App\Http\Requests\UpdateProfileAddressesRequest  $request
     * @return \Illuminate\Http\Response
     */
    public function updateAddresses(UpdateProfileAddressesRequest $request)
    {
        $user = User::where('id',auth()->id())->first();
        $details = $this->updateDetailsUser($request);

        if($user->userDetail()->first()){
            $user->userDetail()->update([
                'city' => $details->city,
                'country' => $details->country,
                'address' => $details->address,
                'house_number' => $details->house_number,
                'postal_code' => $details->postal_code,
                'locality' => $details->locality,
            ]);
        } else {
            $user->userDetail()->save($details);
        }

        return redirect()->route('shop.profile.index','#addresses-tab')->with('success_message','Profile Addresses Update Successfully!');

    }

    protected function updateDetailsUser($request){
        $details = new userDetail;
        $details->phone = $request->phone;
        $details->company = $request->company;
        $details->city = $request->city;
        $details->country = $request->country;
```

```
        $details->address = $request->address;
        $details->house_number = $request->house_number;
        $details->postal_code = $request->postal_code;
        $details->locality = $request->locality;
        return $details;
    }

}
```

8.2.3.9 Comment Controller

```
<?php

namespace App\Http\Controllers;

use App\BlogPost;
use App\Comment;
use App\Http\Requests\CommentRequest;
use App\Reply;
use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class CommentController extends Controller
{
    /**
     * Store a newly created resource in storage.
     *
     * @param  \App\Http\Requests\CommentRequest  $request
     * @param  \App\BlogPost  $id
     * @return \Illuminate\Http\Response
     */
    public function store(CommentRequest $request, $id)
    {
        $user = Auth::user();
        if(!$user){
            $post = BlogPost::find($id);

            $comment = new Comment();
            $comment->name = $request->name;
            $comment->email = $request->email;
            $comment->comment = $request->comment;
            $comment->post()->associate($post);
            $comment->save();

            return back()->with('success_message','Comment successfully added!');
        } else {
            $post = BlogPost::find($id);

            $comment = new Comment();
            $comment->user()->associate($user);
            $comment->comment = $request->comment;
```

```
            $comment->post()->associate($post);
            $comment->save();

            return back()->with('success_message','Comment successfully added!');
        }
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \App\Http\Requests\CommentRequest  $request
     * @param  \App\Comment  $comment_id
     * @return \Illuminate\Http\Response
     */
    public function reply(CommentRequest $request, $comment_id)
    {
        $user = Auth::user();
        $comment = Comment::find($comment_id);
        if(!$user){
            $reply = new Reply();
            $reply->comment()->associate($comment);
            $reply->name = $request->name;
            $reply->email = $request->email;
            $reply->comment = $request->comment;
            $reply->save();

            return back()->with('success_message','Comment successfully added!');
        } else {
            $reply = new Reply();
            $reply->comment()->associate($comment);
            $reply->user()->associate($user);
            $reply->comment = $request->comment;
            $reply->save();

            return back()->with('success_message','Comment successfully added!');
        }

    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\Comment $comment
     * @return \Illuminate\Http\Response
     */
    public function destroy(Comment $comment)
    {
        if(auth()->id() == $comment->user_id) {
            $reply = Reply::where(['comment_id'=>$comment->id]);
            $com = Comment::where(['user_id'=>auth()->id(), 'id'=>$comment->id]);
            if ($reply->count() > 0 && $com->count() > 0) {
                $reply->delete();
                $com->delete();
                return back()->with('success_message','Comment with replies
successfully deleted!');
```

```
            } else if($com->count() > 0){
                $com->delete();
                return back()->with('success_message','Comment successfully
deleted!');
            } else{
                return back()->withErrors('Comment can deleted only from owner!');
            }
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\Reply  $reply
     * @return \Illuminate\Http\Response
     */
    public function destroyReply(Reply $reply)
    {
        if(auth()->id() == $reply->user_id) {
            $rep = Reply::where(['id'=>$reply->id,'user_id'=>auth()->id()]);
            $rep->delete();
            return back()->with('success_message','Comment successfully deleted!');
        } else{
            return back()->withErrors('Comment can deleted only from owner!');
        }
    }
}
```

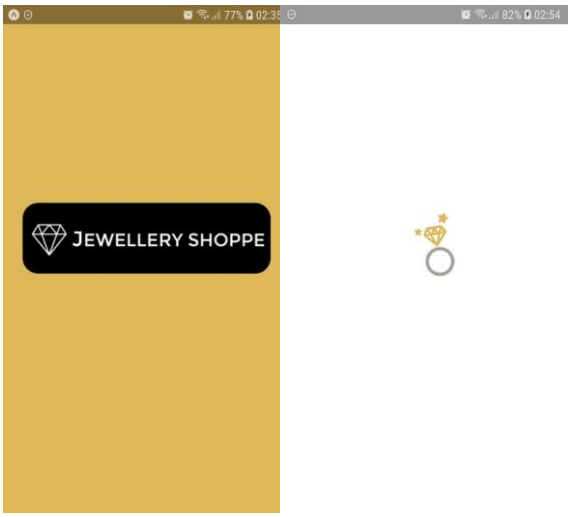
8.2.3.10 Newsletter Controller

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Newsletter;

class NewsletterController extends Controller
{
    public function mailChimp(Request $request)
    {
        if ( ! Newsletter::isSubscribed($request->email) ) {
            Newsletter::subscribe($request->email);
            return  back()->with('success_message', 'Thanks for subscribe your e-
mail!');
        }
        return back()->with( 'warning_message','Sorry! You have already subscribed
this e-mail!');
    }
}
```

Κεφάλαιο 9: Προεπισκόπηση Android Application

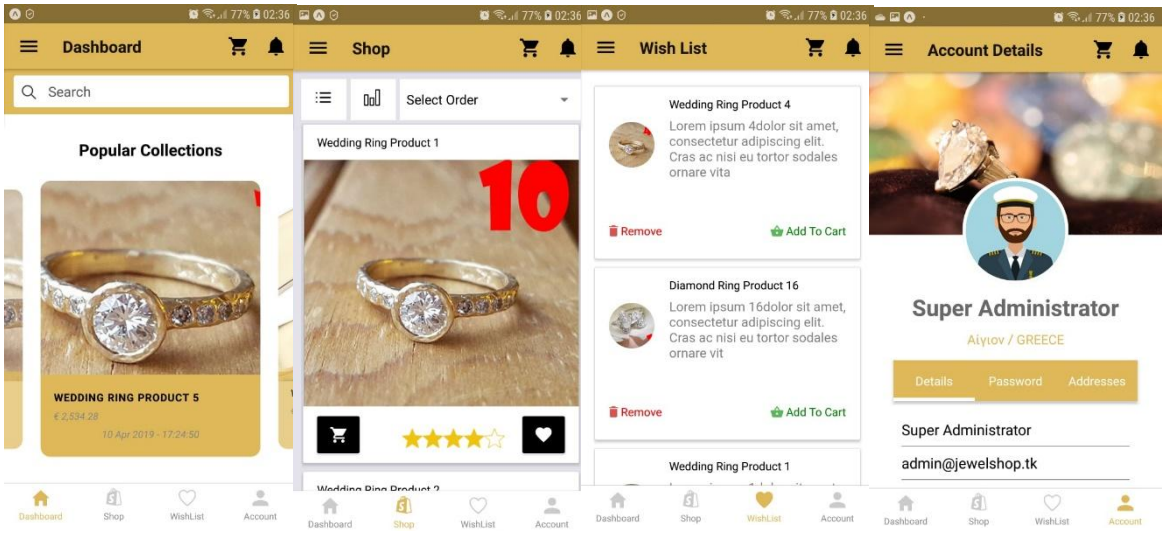
9.1 Κύρια Σημεία Προσομοίωση Επίσκεψης

Παρακάτω θα δούμε το αρχικό screen (splash) και τον loader τις εφαρμογής μου όταν φορτώνει τα δεδομένα για να ανοίξει.



Εικόνα 119 (Android Splash Screen + Loader Screen)

Στην συνέχεια θα δούμε τα τέσσερα βασικά bottom tabs τα οποία και θα αναλύσουμε παρακάτω στα κύρια σημεία του κώδικα κάποια βασικά σημεία.

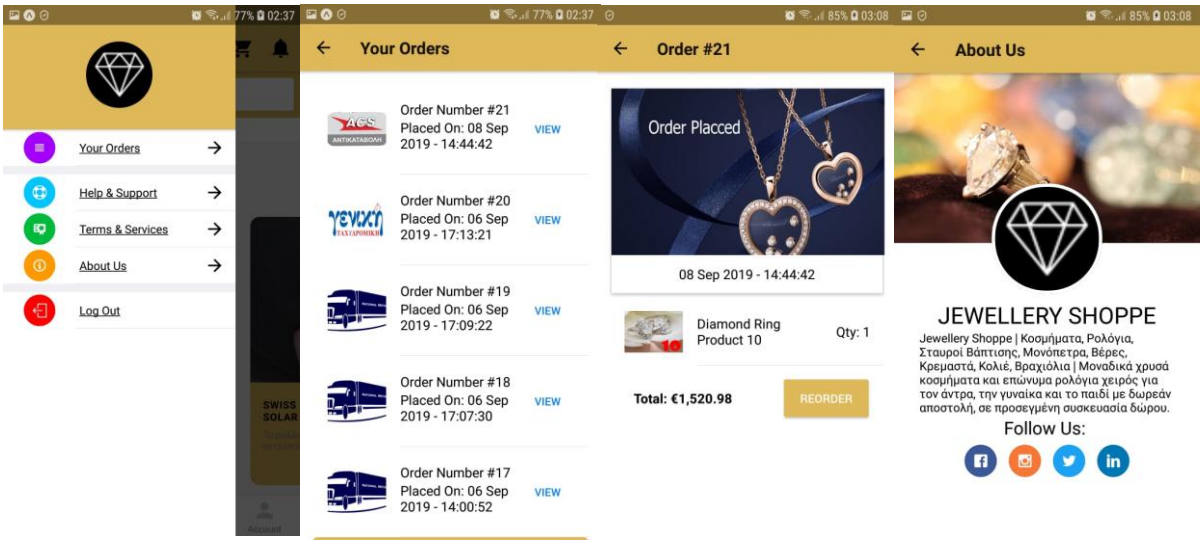


Εικόνα 120 (Dashboard, Shop, Wishlist, Account Screen)

Στην παρακάτω εικόνα θα δούμε το screen shop details σε περίπτωση που ο χρήστης πατήσει είτε στο carousel στο dashboard screen είτε στην shop screen είτε στην wish list screen στον τίτλο του προϊόντος.

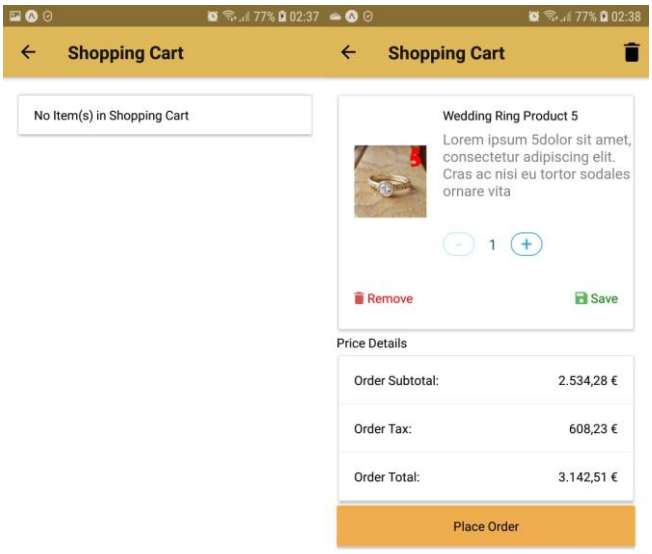
Εικόνα 121 (Shop Details Screen)

Παρακάτω θα δούμε το πλαϊνό menu τις εφαρμογής και μέσα θα δούμε τις παραγγελίες όλες του χρήστη, την παραγγελία αναλυτικά και το screen για το about us.



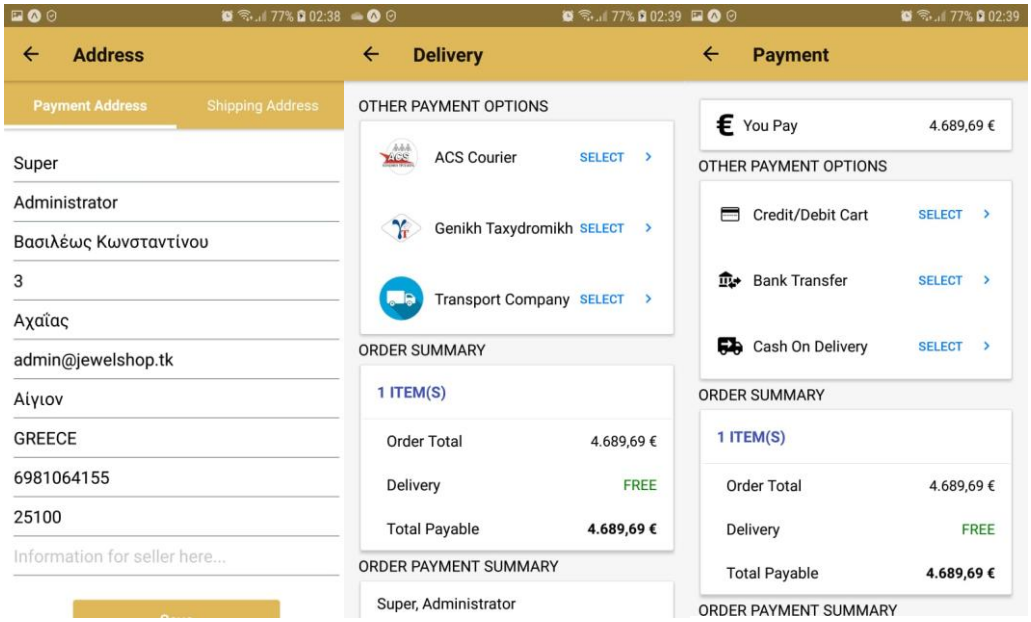
Εικόνα 122 (Drawer, Order, Order Details, About Us Screen)

Στις από κάτω δυο φωτογραφίες θα δούμε πως είναι το screen για το καλάθι όταν είναι άδειο και όταν έχει έστω ένα προϊόν μέσα.

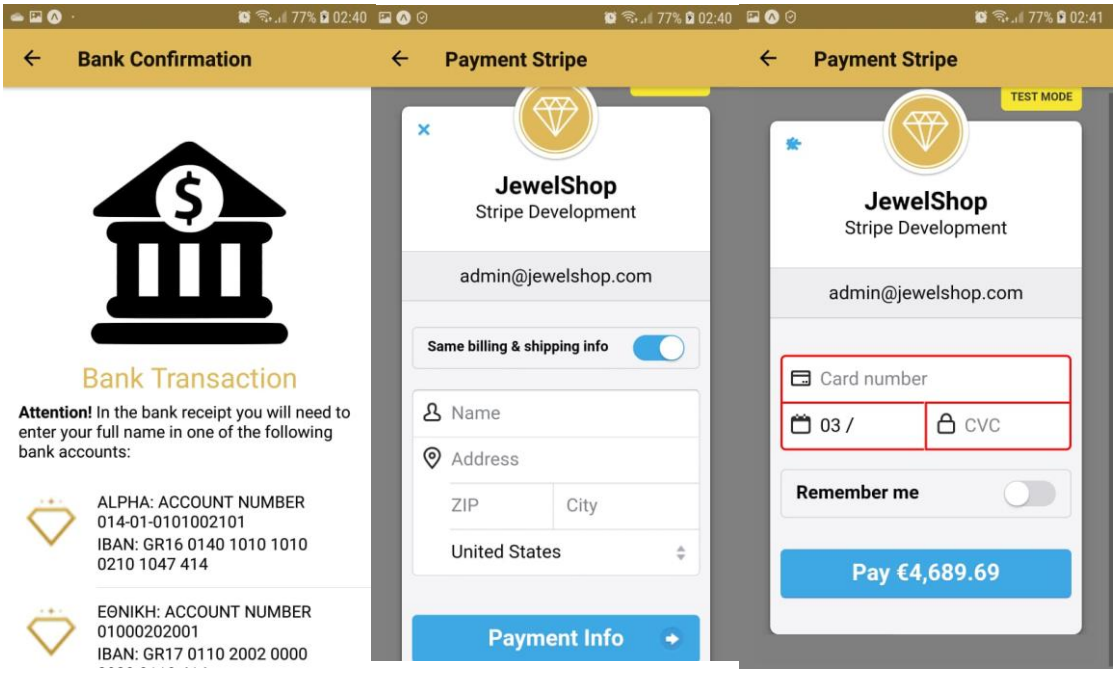


Εικόνα 123 (Shopping Cart Screen)

Και τέλος θα δούμε την ολοκλήρωση τις παραγγελίας με τις διευθύνσεις που μπορεί ο χρήστης να προσθέσει με την μέθοδο μεταφοράς τις παραγγελίας του και με την μέθοδο πληρωμής τις παραγγελίας του.



Εικόνα 124 (Addresses, Delivery, Payment Screen)



Εικόνα 125 (Bank Transfer, Stripe Credit Cart Screen)

9.2 Κύρια Σημεία Κώδικα

9.2.1 Sign In

```
import React from 'react';

import {StyleSheet, Image, View, AsyncStorage} from 'react-native';
import {Input, Button} from 'react-native-elements';

import Colors from "../../constants/Colors";
import {validate, validateErrorMessages} from '../../utils/validation';

import {connect} from 'react-redux'
import {authentication} from "./store/index";

class AuthenticationScreen extends React.Component {
    constructor(props) {
        super(props);
    }

    static navigationOptions = {
        title: "Login",
    };

    state = {
        form: {
            email: {
                value: "",
```

```
                valid: false,
                validationRules: {
                    minLength: 5,
                    isEmail: true,
                    notEmpty: true
                },
                validationErrorMessage: '',
                touched: false,
            },
            password: {
                value: "",
                valid: false,
                validationRules: {
                    minLength: 4,
                    notEmpty: true
                },
            },
            validationErrorMessage: '',
            touched: false,
        },
    };

    updateInputState = (key, value) => {
        let connectedValue = {};
        if (this.state.form[key].validationRules.equalTo) {
            const equalValidate = this.state.form[key].validationRules.equalTo;
            const equalValue = this.state.form[equalValidate].value;
            connectedValue = {
                ...connectedValue,
                equalTo: equalValue
            };
        }
        if (key === "password") {
            connectedValue = {
                ...connectedValue,
                equalTo: value
            };
        }
        this.setState(prevState => {
            return {
                ...prevState,
                form: {
                    ...prevState.form,
                    [key]: {
                        ...prevState.form[key],
                        value: value,
                        valid: validate(
                            value,
                            prevState.form[key].validationRules,
                            connectedValue
                        ),
                        validationErrorMessage: validateErrorMessages(value,
prevState.form[key].validationRules, connectedValue),
                        touched: true
                    }
```

```
                },
            };
        });
    };

    render() {
        return (
            <View style={styles.container}>
                <View style={styles.imageContainer}>
                    <Image style={styles.image}
                           source={require("../../assets/images/icon.png")}/>
                </View>
                <View style={{justifyContent: 'center', alignItems: 'center'}}>
                    <Input
                        value={this.state.form.email.value}
                        onChangeText={(val) => this.updateInputState("email", val)}
                        leftIcon={{ type: 'MaterialIcons', name: 'email' }}
                        keyboardType='email-address'
                        placeholder='E-mail'
                        errorStyle={{color: 'red'}}
                        containerStyle={{width: '90%', margin: 5}}

errorMessage={this.state.form["email"].validationErrorMessage}
                    />
                    <Input
                        value={this.state.form.password.value}
                        onChangeText={(val) => this.updateInputState("password",
val)}
                        leftIcon={{ type: 'AntDesign', name: 'lock' }}
                        shake={true}
                        placeholder='Password'
                        secureTextEntry={true}
                        errorStyle={{color: 'red'}}
                        containerStyle={{width: '90%', margin: 5}}

errorMessage={this.state.form["password"].validationErrorMessage}
                    />
                    <Button
                        buttonStyle={{backgroundColor:Colors.tabIconSelected}}
                        title="Login"
                        containerStyle={{width: '80%', margin: 20}}
                        loading={this.props.loading}
                        disabled={!this.state.form.email.valid ||
!this.state.form.password.valid}
                        disabled={false}
                        onPress={() =>
this.props.onAuthenticate(this.state.form.email.value,
this.state.form.password.value)}
                    />
                </View>
            </View>
        );
    }
}
```

```
const styles = StyleSheet.create({
    container: {
        flex: 1,
        alignContent: 'center'
    },
    imageContainer: {
        justifyContent: 'center',
        alignItems: 'center',
        textAlign: 'center',
        margin: 10
    },
    image: {
        width: 100,
        height: 100
    },
});


const mapStateToProps = state => {
    return {
        loading: state.auth.loading
    }
};

const mapDispatchToProps = dispatch => {
    return {
        onAuthenticate: (email, password) => dispatch(authentication(email,
password)),
    }
};
export default connect(mapStateToProps, mapDispatchToProps)(AuthenticationScreen);
```

9.2.2 Dashboard

```
import React from 'react';
import {View, ScrollView, Text, SafeAreaView, RefreshControl, AsyncStorage, Animated}
from 'react-native';
import {Container, Content} from 'native-base'

import {MenuIcon, NotificationIcon} from "../../components/HeaderIcons";
import SearchBar from '../../components/SearchBar'
import Carousel from 'react-native-snap-carousel';
import { sliderWidth, itemWidth } from './style/Slider.style';
import Slider from './components/Slider';
import styles from './style/index.style';
import {onApiTokenValid} from '../../utils/api'
import colors from "../../constants/Colors";
import {connect} from "react-redux";

const SLIDER_INIT = 1;

class DashboardHome extends React.Component {
    static navigationOptions = ({navigation}) => ({
        title: 'Dashboard',
```

```
        headerLeft: (
            <MenuIcon openDrawer={navigation.openDrawer} />
        ),
        headerRight: (
            <NotificationIcon navigate={navigation.navigate}
countCart={navigation.getParam('countCart','0')}/>
        ),
    });

    constructor (props) {
        super(props);
        this.state = {
            ActiveSlide: SLIDER_INIT,
            popProducts: {},
            popBlogs: {},
            categories: {},
            refreshing: true,
            token: false,
        };
    }

    componentWillMount() {
        // const {setParams} = this.props.navigation;
        this.props.navigation.setParams({countCart: this.props.cart.length});

        this.didFocusCart = this.props.navigation.addListener('didFocus', () => {
            console.log('didFocus');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });

        this.willFocusCart = this.props.navigation.addListener('willFocus', () => {
            console.log('willFocus');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });

        this.didBlurCart = this.props.navigation.addListener('didBlur', () => {
            console.log('didBlur');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });

        this.willBlurCart = this.props.navigation.addListener('willBlur', () => {
            console.log('willBlur');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });
    }

    componentWillUnmount() {
        this.didFocusCart.remove();
        this.willFocusCart.remove();
        this.didBlurCart.remove();
        this.willBlurCart.remove();
    }

    componentDidMount(){
        this._onApiCall();
```

```
    }


    _renderItem ({item, index}) {
        return (
            <Slider
                key={index}
                data={item}
                navigation={ this.props.navigation }/>
        );
    }

    _renderItemWithParallax ({item, index}, parallaxProps) {
        return (
            <Slider
                key={index}
                data={item}
                parallax={true}
                parallaxProps={parallaxProps}
                navigation={ this.props.navigation }/>
        );
    }

    _onApiCall = () => {
        return new Promise((resolve, reject) => {
            const url = 'https://api.jewelshop.tk/api/v1/home';
            AsyncStorage.getItem("jewelshop:token", (error, token) => {
                console.log(token);
                setTimeout(()=>{
                    fetch(url, {
                        method: 'GET',
                        headers: {
                            Accept : 'application/json',
                            'Content-Type': 'application/json',
                            'Authorization': "Bearer " + token
                        },
                    })
                    .then(response => response.json())
                    .then((responseJson)=> {
                        if(responseJson["status_code"] === 401 ||
responseJson["status_code"] === 500){
                            onApiTokenValid()
                                .then((done)=> {
                                    if(done){
                                        this.setState(prevState => ({
                                            ...prevState,
                                            refreshing: false,
                                        }));
                                    }
                                    resolve(true);
                                })
                                .catch(() => {
                                    console.log(error)
                                    reject(false);
                                });
```

```
                    } else {
                        this.setState(prevState => ({
                            ...prevState,
                            refreshing: false,
                            popProducts: responseJson["products"][0],
                            popBlogs: responseJson["posts"][0],
                            categories: responseJson["categories"][0]
                        }));
                        resolve(true);
                    }
                })
                .catch(error => {
                    console.log(error);
                    reject(false);
                }) //to catch the errors if any
            },500)
        });
    });
};

_onRefresh = () => {
    this.setState(prevState => ({
        ...prevState,
        refreshing: true,
    }));
    this._onApiCall()
        .then(() => {
            this.setState(prevState => ({
                ...prevState,
                refreshing: false,
            }));
        })
        .catch((error) => {
            console.error(error);
        });
};

sliderMain (title,data) {
    return (
        <View style={styles.exampleContainer}>
            <Text style={styles.title}>{title}</Text>
            <Carousel
                data={data}
                index={(index) => index}
                key={(key) => key}
                renderItem={this._renderItemWithParallax.bind(this)}
                sliderWidth={sliderWidth}
                itemWidth={itemWidth}
                hasParallaxImages={true}
                firstItem={SLIDER_INIT}
                inactiveSlideScale={0.94}
                inactiveSlideOpacity={0.7}
                containerCustomStyle={styles.slider}
                contentContainerCustomStyle={styles.sliderContentContainer}
                loop={true}
```

```
                    loopClonesPerSide={2}
                    autoplay={true}
                    autoplayDelay={500}
                    autoplayInterval={3000}
                    onSnapToItem={(index) => this.setState({ ActiveSlide: index }) }
                />
            </View>
        );
    }

    momentum (title,data) {
        return (
            <View style={styles.exampleContainer}>
                <Text style={styles.title}>{title}</Text>
                <Carousel
                    data={data}
                    index={(index) => index}
                    key={(key) => key}
                    renderItem={this._renderItem.bind(this)}
                    sliderWidth={sliderWidth}
                    itemWidth={itemWidth}
                    inactiveSlideScale={0.95}
                    inactiveSlideOpacity={1}
                    enableMomentum={true}
                    activeSlideAlignment={'start'}
                    containerCustomStyle={styles.slider}
                    contentContainerCustomStyle={styles.sliderContentContainer}
                    activeAnimationType={'spring'}
                    activeAnimationOptions={{
                        friction: 4,
                        tension: 40
                    }}
                />
            </View>
        );
    }

    render() {
        let products = null;
        let blog_posts = null;
        let categories = null;
        if(!this.state.refreshing){
            products =  this.sliderMain('Popular Collections',
this.state.popProducts.data);
            blog_posts = this.momentum('Popular Blog Posts',
this.state.popBlogs.data);
            categories = this.sliderMain('Product
Categories',this.state.categories.data);
        }

        return (
            <Container>
                <SearchBar />
                {!this.state.refreshing ?
                    <ScrollView
```

```
                        refreshControl={
                            <RefreshControl
                                refreshing={this.state.refreshing}
                                onRefresh={this._onRefresh}
                                enabled={true}
                                tintColor="#fff" titleColor="#fff"
colors={[colors.tabIconSelected]}
                            />
                        }>
                        <Content>
                            <SafeAreaView  style={styles.safeArea}>
                                <View style={styles.container}>
                                    <ScrollView
                                        style={styles.scrollView}
                                        scrollEventThrottle={200}
                                        directionalLockEnabled={true}
                                    >
                                        { products }

                                    </ScrollView>
                                </View>
                                <View style={styles.containerBackground}>
                                    <ScrollView
                                        style={styles.scrollView}
                                        scrollEventThrottle={200}
                                        directionalLockEnabled={true}
                                    >
                                        { blog_posts }
                                    </ScrollView>
                                </View>
                                <View style={styles.container}>
                                    <ScrollView
                                        style={styles.scrollView}
                                        scrollEventThrottle={200}
                                        directionalLockEnabled={true}
                                    >
                                        { categories }
                                    </ScrollView>
                                </View>
                            </SafeAreaView>
                        </Content>
                    </ScrollView>
                    : <ScrollView
                        refreshControl={
                            <RefreshControl
                                refreshing={this.state.refreshing}
                                onRefresh={this._onRefresh}
                                enabled={true}
                                tintColor="#fff" titleColor="#fff"
colors={[colors.tabIconSelected]}
                            />
                        }>
                    </ScrollView> }
            </Container>
        );
```

```
    }
}

const mapStateToProps = state => {
    return {
        cart: state.cart.products
    }
};

export default connect(mapStateToProps)(DashboardHome);
```

9.2.3 Bottom Tabs

```
import React from 'react';
import { Platform } from 'react-native';
import { createStackNavigator, createBottomTabNavigator } from 'react-navigation';
import {Icon} from "react-native-elements";
import Colors from '../constants/Colors';

import TabBarIcon from '../components/TabBarIcon';
import DashboardScreen from "../screens/dashboard/home";
import ShopScreen from "../screens/shop/home";
import WishlistScreen from "../screens/wishlist/home";
import AccountScreen from "../screens/account/home";

const DashboardStack = createStackNavigator({
    Home: DashboardScreen,
},{
    defaultNavigationOptions : {
        headerStyle:{
            backgroundColor: Colors.tabIconSelected
        }
    }
});

DashboardStack.navigationOptions = {
  tabBarLabel: 'Dashboard',
  tabBarIcon: ({ focused }) => (
    <TabBarIcon
      focused={focused}
      name={
        Platform.OS === 'ios'
          ? `ios-home${focused ? '' : '-outline'}`
          : 'md-home'
      }
    />
  ),
};

const ShopStack = createStackNavigator({
    Shop: ShopScreen,
},{
    defaultNavigationOptions : {
        headerStyle:{
```

```
            backgroundColor: Colors.tabIconSelected
        }
    }
});

ShopStack.navigationOptions = {
    tabBarLabel: 'Shop',
    tabBarIcon: ({ focused }) => (
        <Icon
            name='shopify'
            type='material-community'
            color={focused ? Colors.tabIconSelected : Colors.tabIconDefault}
            size={26}/>
    ),
};


const WishlistStack = createStackNavigator({
    Wishlist: WishlistScreen,
},{
    defaultNavigationOptions : {
        headerStyle:{
            backgroundColor: Colors.tabIconSelected
        }
    }
});

WishlistStack.navigationOptions = {
  tabBarLabel: 'WishList',
  tabBarIcon: ({ focused }) => (
    <TabBarIcon
      focused={focused}
      name={Platform.OS === 'ios' ? `md-heart${focused ? "" : "-empty"}` : `ios-
heart${focused ? "" : "-empty"}`}
    />
  ),
};


const AccountStack = createStackNavigator({
    Account: AccountScreen,
},{
    defaultNavigationOptions : {
        headerStyle:{
            backgroundColor: Colors.tabIconSelected
        }
    }
});

AccountStack.navigationOptions = {
    tabBarLabel: 'Account',
    tabBarIcon: ({ focused }) => (
        <TabBarIcon
            focused={focused}
            name={Platform.OS === 'ios' ? 'ios-person' : 'md-person'}
```

```
        />
    ),
};

export default createBottomTabNavigator({
    DashboardStack,
    ShopStack,
    WishlistStack,
    AccountStack,
},{
    tabBarOptions: {
        activeTintColor: Colors.tabIconSelected,
    }
});
```

9.2.4 Profile

```
import React from 'react';

import {View, Text, StyleSheet, Image, KeyboardAvoidingView, ScrollView, Dimensions,
AsyncStorage, RefreshControl} from 'react-native';
import {Button, Input} from "react-native-elements";
import {Tabs, Tab, Container} from 'native-base'

import {MenuIcon, NotificationIcon} from "../../components/HeaderIcons";
import colors from "../../constants/Colors";

import {onApiTokenValid} from "../../utils/api";
import {validate, validateErrorMessages} from "../../utils/validation";
import {connect} from "react-redux";
import {toastr} from "../../components/Toast";
const {width} = Dimensions.get('window');

class AccountHome extends React.Component {

    static navigationOptions = ({navigation}) => ({
        title: 'Account Details',
        headerLeft: (
            <MenuIcon openDrawer={navigation.openDrawer} />
        ),
        headerRight: (
            <NotificationIcon navigate={navigation.navigate}
countCart={navigation.getParam('countCart','0')}/>
        ),
    });

    constructor (props) {
        super(props);
    }

    componentDidMount(){
        AsyncStorage.getItem("jewelshop:token", (error, exists) => {
            this.setState(prevState => ({
                ...prevState,
```

```
                token: exists
            }));
            this._onApiCall();
        });
    }

    componentWillMount() {
        // const {setParams} = this.props.navigation;
        this.props.navigation.setParams({countCart: this.props.cart.length});

        this.didFocusCart = this.props.navigation.addListener('didFocus', () => {
            console.log('didFocus');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });

        this.willFocusCart = this.props.navigation.addListener('willFocus', () => {
            console.log('willFocus');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });

        this.didBlurCart = this.props.navigation.addListener('didBlur', () => {
            console.log('didBlur');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });

        this.willBlurCart = this.props.navigation.addListener('willBlur', () => {
            console.log('willBlur');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });
    }

    componentWillUnmount() {
        this.didFocusCart.remove();
        this.willFocusCart.remove();
        this.didBlurCart.remove();
        this.willBlurCart.remove();
    }

    state = {
        userDetails: {},
        refreshing: true,
        token: false,
        form: {
            name: {
                value: "",
                valid: false,
                validationRules: {
                    minLength: 10,
                    notEmpty: true
                },
                validationErrorMessage: '',
                touched: false,
            },
            email: {
                value: "",
```

```
            valid: false,
            validationRules: {
                minLength: 5,
                isEmail: true,
                notEmpty: true
            },
            validationErrorMessage: '',
            touched: false,
        },
        phone: {
            value: "",
            valid: false,
            validationRules: {
                minLength: 10,
                notEmpty: true
            },
            validationErrorMessage: '',
            touched: false,
        },
        company: {
            value: "",
            valid: false,
            validationRules: {
                minLength: 3,
                notEmpty: true
            },
            validationErrorMessage: '',
            touched: false,
        },
        old_password: {
            value: "",
            valid: false,
            validationRules: {
                minLength: 8,
                notEmpty: true
            },
        },
        password: {
            value: "",
            valid: false,
            validationRules: {
                minLength: 8,
                notEmpty: true
            },
        },
        password_confirmation: {
            value: "",
            valid: false,
            validationRules: {
                minLength: 8,
                notEmpty: true
            },
        },
        city: {
            value: "",
```

```
                valid: false,
                validationRules: {
                    minLength: 3,
                    notEmpty: true
                },
            },
            country: {
                value: "",
                valid: false,
                validationRules: {
                    minLength: 5,
                    notEmpty: true
                },
            },
            address: {
                value: "",
                valid: false,
                validationRules: {
                    minLength: 6,
                    notEmpty: true
                },
            },
            house_number: {
                value: "",
                valid: false,
                validationRules: {
                    minLength: 1,
                    notEmpty: true
                },
            },
            zip: {
                value: "",
                valid: false,
                validationRules: {
                    minLength: 5,
                    notEmpty: true
                },
            },
            locality: {
                value: "",
                valid: false,
                validationRules: {
                    minLength: 5,
                    notEmpty: true
                },
            },
            validationErrorMessage: '',
            touched: false,
        },
    };

    updateInputState = (key, value) => {
        let connectedValue = {};
        if (this.state.form[key].validationRules.equalTo) {
            const equalValidate = this.state.form[key].validationRules.equalTo;
```

```
            const equalValue = this.state.form[equalValidate].value;
            connectedValue = {
                ...connectedValue,
                equalTo: equalValue
            };
        }
        if (key === "password") {
            connectedValue = {
                ...connectedValue,
                equalTo: value
            };
        }
        this.setState(prevState => {
            return {
                ...prevState,
                form: {
                    ...prevState.form,
                    [key]: {
                        ...prevState.form[key],
                        value: value,
                        valid: validate(
                            value,
                            prevState.form[key].validationRules,
                            connectedValue
                        ),
                        validationErrorMessage: validateErrorMessages(value,
prevState.form[key].validationRules, connectedValue),
                        touched: true
                    }
                },
            };
        });
    };

    _onRefresh = () => {
        this.setState(prevState => ({
            ...prevState,
            refreshing: true,
        }));
        this._onApiCall()
            .then(() => {
                this.setState(prevState => ({
                    ...prevState,
                    refreshing: false,
                }));
            })
            .catch((error) => {
                console.error(error);
            });
    };

    _onUpdateDetails = (name, email, phone, company) => {
        let userId = this.props.id;
        const url = 'https://api.jewelshop.tk/api/v1/user/'+userId+'/details/update';
        fetch(url, {
```

```
            method: 'PATCH',
            headers: {
                Accept : 'application/json',
                'Content-Type': 'application/json',
                'Authorization': "Bearer " + this.state.token,
            },
            body: JSON.stringify({
                name: name,
                email: email,
                phone: phone,
                company: company,
            }),
        })
        .then(response => response.json())
        .then((responseJson)=> {
            if(!responseJson["error"]){
                this._onRefresh();
                toastr.showToast('User Details
Saved','success','bottom','center',null,2500);
            } else {
                toastr.showToast('Something went
Wrong!','warning','bottom','center',null,2500);
            }
        })
        .catch(error => {
            toastr.showToast('No Internet Connection, Please connect to internet and
try again!','warning','bottom','left','Okey',2500);
            console.log(error);
        })
    };

    _onUpdatePassword = (old_password, password, password_confirmation) => {
        let userId = this.props.id;
        const url =
'https://api.jewelshop.tk/api/v1/user/'+userId+'/password/update';
        fetch(url, {
            method: 'PATCH',
            headers: {
                Accept : 'application/json',
                'Content-Type': 'application/json',
                'Authorization': "Bearer " + this.state.token,
            },
            body: JSON.stringify({
                old_password: old_password,
                password: password,
                password_confirmation: password_confirmation,
            }),
        })
            .then(response => response.json())
            .then((responseJson)=> {
                console.log(responseJson);
                if(!responseJson["errors"]){
                    if(responseJson["status_code"] === 204){

toastr.showToast(responseJson["message"],'warning','bottom','center',null,2500);
```

```
                    } else {
                        this.setState(prevState => ({
                            ...prevState,
                            form: {
                                ...prevState.form,
                                old_password: {
                                    ...prevState.form.old_password,
                                    value: ""
                                },
                                password: {
                                    ...prevState.form.password,
                                    value: ""
                                },
                                password_confirmation: {
                                    ...prevState.form.password_confirmation,
                                    value: ""
                                }
                            }

                        }));
                        this._onRefresh();
                        toastr.showToast('User Password
Changed','success','bottom','center',null,2500);
                    }
                } else {

toastr.showToast(responseJson["errors"]["password"],'warning','bottom','center',null,
2500);
                }
            })
            .catch(error => {
                toastr.showToast('No Internet Connection, Please connect to internet
and try again!','warning','bottom','left','Okey',2500);
                console.log(error);
            })
    };

    _onUpdateAddress = (city, country, address, house_number, zip, locality) => {
        let userId = this.props.id;
        const url = 'https://api.jewelshop.tk/api/v1/user/'+userId+'/address/update';
        fetch(url, {
            method: 'PATCH',
            headers: {
                Accept : 'application/json',
                'Content-Type': 'application/json',
                'Authorization': "Bearer " + this.state.token,
            },
            body: JSON.stringify({
                city: city,
                country: country,
                address: address,
                house_number: house_number,
                postal_code: zip,
                locality: locality,
            }),
```

```
        })
            .then(response => response.json())
            .then((responseJson)=> {
                console.log(responseJson);
                if(!responseJson["errors"]){
                    this._onRefresh();
                    toastr.showToast('User Addresses
Saved','success','bottom','center',null,2500);
                } else {
                    if(responseJson["status_code"] === 422){
                        if(responseJson["errors"]["city"]){

toastr.showToast(responseJson["errors"]["city"],'warning','bottom','center',null,2500
);
                        } else if(responseJson["errors"]["country"]){

toastr.showToast(responseJson["errors"]["country"],'warning','bottom','center',null,2
500);
                        } else if(responseJson["errors"]["address"]){

toastr.showToast(responseJson["errors"]["address"],'warning','bottom','center',null,2
500);
                        } else if(responseJson["errors"]["house_number"]){

toastr.showToast(responseJson["errors"]["house_number"],'warning','bottom','center',n
ull,2500);
                        } else if(responseJson["errors"]["postal_code"]){

toastr.showToast(responseJson["errors"]["postal_code"],'warning','bottom','center',nu
ll,2500);
                        } else if(responseJson["errors"]["locality"]){

toastr.showToast(responseJson["errors"]["locality"],'warning','bottom','center',null,
2500);
                        } else {

toastr.showToast(responseJson["message"],'warning','bottom','center',null,2500);
                        }
                    } else {
                        toastr.showToast('Something went
Wrong!','warning','bottom','center',null,2500);
                    }
                }
            })
            .catch(error => {
                toastr.showToast('No Internet Connection, Please connect to internet
and try again!','warning','bottom','left','Okey',2500);
                console.log(error);
            })
    };

    _onApiCall = () => {
        return new Promise((resolve, reject) => {
            let userId = this.props.uid;
            const url = 'https://api.jewelshop.tk/api/v1/user/'+userId+'/details';
```

```
            fetch(url, {
                method: 'GET',
                headers: {
                    Accept : 'application/json',
                    'Content-Type': 'application/json',
                    'Authorization': "Bearer " + this.state.token,
                },
            })
                .then(response => response.json())
                .then((responseJson)=> {
                    if(responseJson["status_code"] === 401 ||
responseJson["status_code"] === 500){
                        onApiTokenValid()
                            .then((done)=> {
                                if(done){
                                    this.setState(prevState => ({
                                        ...prevState,
                                        refreshing: false,
                                    }));
                                    resolve(true);
                                }
                            })
                            .catch(() => {
                                console.log(error)
                                reject(false);
                            });
                    } else {
                        this.setState(prevState => ({
                            ...prevState,
                            refreshing: false,
                            userDetails: responseJson["data"],
                            form: {
                                ...prevState.form,
                                name: {
                                    ...prevState.form.name,
                                    value: responseJson["data"]["name"]
                                },
                                email: {
                                    ...prevState.form.email,
                                    value: responseJson["data"]["email"]
                                },
                                phone: {
                                    ...prevState.form.phone,
                                    value:
responseJson["data"]["details"]["data"]["phone"]
                                },
                                company: {
                                    ...prevState.form.company,
                                    value:
responseJson["data"]["details"]["data"]["company"]
                                },
                                city: {
                                    ...prevState.form.city,
                                    value:
responseJson["data"]["details"]["data"]["city"]
```

```
                                },
                                country: {
                                    ...prevState.form.country,
                                    value:
responseJson["data"]["details"]["data"]["country"]
                                },
                                address: {
                                    ...prevState.form.address,
                                    value:
responseJson["data"]["details"]["data"]["address"]
                                },
                                house_number: {
                                    ...prevState.form.house_number,
                                    value:
responseJson["data"]["details"]["data"]["house_number"]
                                },
                                zip: {
                                    ...prevState.form.zip,
                                    value:
responseJson["data"]["details"]["data"]["zip"]
                                },
                                locality: {
                                    ...prevState.form.locality,
                                    value:
responseJson["data"]["details"]["data"]["locality"]
                                },
                            }

                        }));
                        // console.log(responseJson["data"]);
                        resolve(true);
                    }
                })
                .catch(error => {
                    console.log(error);
                    reject(false);
                }) //to catch the errors if any
        })
    };

    render() {
        return (
            <Container>
                {!this.state.refreshing ?
                    <ScrollView
                        refreshControl={
                            <RefreshControl
                                refreshing={this.state.refreshing}
                                onRefresh={this._onRefresh}
                                enabled={true}
                                tintColor="#fff" titleColor="#fff"
colors={[colors.tabIconSelected]}
                            />
                        }>
                        <View style={styles.container}>
```

```
                            <View style={styles.header}>
                                <Image style={styles.image}
source={require('../../assets/images/admin_background.jpg')} />
                            </View>
                            <Image style={styles.avatar} source={{uri:
this.state.userDetails.avatar}}/>
                            <View style={styles.body}>
                                <View style={styles.bodyContent}>
                                    <Text
style={styles.name}>{this.state.userDetails.name}</Text>
                                    <Text style={styles.info}>
                                        {this.state.userDetails.details.data.city}
                                        {this.state.userDetails.details.data.city !==
null ? ' / ' : ' '}
                                        {this.state.userDetails.details.data.country}
                                    </Text>

                                    <Tabs style={{paddingTop: 15}}>
                                        <Tab heading="Details"
tabStyle={{backgroundColor: colors.tabIconSelected}} textStyle={{color: '#fff'}}
activeTabStyle={{backgroundColor: colors.tabIconSelected}} activeTextStyle={{color:
'#fff', fontWeight: 'normal'}}>
                                            {/*<Details
userDetails={this.state.userDetails}/>*/}
                                            <View style={{justifyContent: 'center',
alignItems: 'center',marginTop: 15}}>
                                                <Input

value={this.state.form.name.value}

                                                    onChangeText={(val) =>
this.updateInputState("name", val)}

                                                    placeholder='Full Name'
                                                    errorStyle={{color: 'red'}}

errorMessage={this.state.form["name"].validationErrorMessage}
                                                />
                                                <Input

value={this.state.form.email.value}

                                                    onChangeText={(val) =>
this.updateInputState("email", val)}

                                                    keyboardType='email-address'
                                                    placeholder='E-mail'
                                                    errorStyle={{color: 'red'}}

errorMessage={this.state.form["email"].validationErrorMessage}
                                                />
                                                <Input
                                                    keyboardType="numeric"

value={this.state.form.phone.value}

                                                    onChangeText={(val) =>
this.updateInputState("phone", val)}

                                                    placeholder='Phone'
                                                    errorStyle={{color: 'red'}}
```

```
errorMessage={this.state.form["phone"].validationErrorMessage}
                                                    />
                                                    <Input

value={this.state.form.company.value}
                                                        onChangeText={(val) =>
this.updateInputState("company", val)}
                                                        placeholder='Company'
                                                        errorStyle={{color: 'red'}}

errorMessage={this.state.form["company"].validationErrorMessage}
                                                    />
                                                    <Button

buttonStyle={{backgroundColor:colors.tabIconSelected,marginTop: 20}}
                                                        title="Save"
                                                        containerStyle={{width: '30%',
justifyContent: 'center'}}
                                                        disabled={false}
                                                        onPress={() =>
this._onUpdateDetails(this.state.form.name.value,this.state.form.email.value,this.sta
te.form.phone.value,this.state.form.company.value)}
                                                    />
                                                </View>
                                            </Tab>
                                            <Tab heading="Password"
tabStyle={{backgroundColor: colors.tabIconSelected}} textStyle={{color: '#fff'}}
activeTabStyle={{backgroundColor: colors.tabIconSelected}} activeTextStyle={{color:
'#fff', fontWeight: 'normal'}}>
                                                {/*<Password />*/}
                                                <View style={{justifyContent: 'center',
alignItems: 'center',marginTop: 15}}>
                                                    <Input

value={this.state.form.old_password.value}
                                                        onChangeText={(val) =>
this.updateInputState("old_password", val)}
                                                        secureTextEntry={true}
                                                        placeholder='Old Password'
                                                        errorStyle={{color: 'red'}}

errorMessage={this.state.form["old_password"].validationErrorMessage}
                                                    />
                                                    <Input

value={this.state.form.password.value}
                                                        onChangeText={(val) =>
this.updateInputState("password", val)}
                                                        secureTextEntry={true}
                                                        placeholder='New Password'
                                                        errorStyle={{color: 'red'}}

errorMessage={this.state.form["password"].validationErrorMessage}
                                                    />
```

```
                                            <Input

value={this.state.form.password_confirmation.value}
                                                onChangeText={(val) =>
this.updateInputState("password_confirmation", val)}
                                                secureTextEntry={true}
                                                placeholder='Retype New Password'
                                                errorStyle={{color: 'red'}}

errorMessage={this.state.form["password_confirmation"].validationErrorMessage}
                                            />
                                            <Button

buttonStyle={{backgroundColor:colors.tabIconSelected,marginTop: 20}}
                                                title="Save"
                                                containerStyle={{width: '30%',
justifyContent: 'center'}}

disabled={!this.state.form.old_password.valid || !this.state.form.password.valid ||
!this.state.form.password_confirmation.valid}
                                                onPress={() =>
this._onUpdatePassword(this.state.form.old_password.value,this.state.form.password.va
lue,this.state.form.password_confirmation.value)}
                                            />
                                        </View>
                                    </Tab>
                                    <Tab heading="Addresses"
tabStyle={{backgroundColor: colors.tabIconSelected}} textStyle={{color: '#fff'}}
activeTabStyle={{backgroundColor: colors.tabIconSelected}} activeTextStyle={{color:
'#fff', fontWeight: 'normal'}}>
                                        {/*<Addresses />*/}
                                        <View style={{justifyContent: 'center',
alignItems: 'center',marginTop: 15}}>
                                            <Input

value={this.state.form.city.value}

                                                onChangeText={(val) =>
this.updateInputState("city", val)}
                                                placeholder='City'
                                                errorStyle={{color: 'red'}}

errorMessage={this.state.form["city"].validationErrorMessage}
                                            />
                                            <Input

value={this.state.form.country.value}

                                                onChangeText={(val) =>
this.updateInputState("country", val)}
                                                placeholder='Country'
                                                errorStyle={{color: 'red'}}

errorMessage={this.state.form["country"].validationErrorMessage}
                                            />
                                            <Input
```

```
value={this.state.form.address.value}
                                                    onChangeText={(val) =>
this.updateInputState("address", val)}
                                                    placeholder='Address'
                                                    errorStyle={{color: 'red'}}

errorMessage={this.state.form["address"].validationErrorMessage}
                                                />
                                                <Input
                                                    keyboardType="numeric"

value={this.state.form.house_number.value}
                                                    onChangeText={(val) =>
this.updateInputState("house_number", val)}
                                                    placeholder='House Number'
                                                    errorStyle={{color: 'red'}}

errorMessage={this.state.form["house_number"].validationErrorMessage}
                                                />
                                                <Input
                                                    keyboardType="numeric"
                                                    value={this.state.form.zip.value}
                                                    onChangeText={(val) =>
this.updateInputState("zip", val)}
                                                    placeholder='Postal Code'
                                                    errorStyle={{color: 'red'}}

errorMessage={this.state.form["zip"].validationErrorMessage}
                                                />
                                                <Input

value={this.state.form.locality.value}
                                                    onChangeText={(val) =>
this.updateInputState("locality", val)}
                                                    placeholder='Locality'
                                                    errorStyle={{color: 'red'}}

errorMessage={this.state.form["locality"].validationErrorMessage}
                                                />
                                                <Button

buttonStyle={{backgroundColor:colors.tabIconSelected,marginTop: 20}}
                                                    title="Save"
                                                    containerStyle={{width: '30%',
justifyContent: 'center'}}
                                                    disabled={false}
                                                    onPress={() =>
this._onUpdateAddress(this.state.form.city.value,this.state.form.country.value,this.s
tate.form.address.value,this.state.form.house_number.value,this.state.form.zip.value,
this.state.form.locality.value)}
                                                />
                                            </View>
                                        </Tab>
                                    </Tabs>
                                </View>
```

```
                        </View>
                    </View>
                </ScrollView>
            :
                <ScrollView
                    refreshControl={
                        <RefreshControl
                            refreshing={this.state.refreshing}
                            onRefresh={this._onRefresh}
                            enabled={true}
                            tintColor="#fff" titleColor="#fff"
colors={[colors.tabIconSelected]}
                        />
                    }>
                </ScrollView>
            }
        </Container>
    );
  }
}

const styles = StyleSheet.create({
    header:{
        flex: 1,
        justifyContent: 'center',
        backgroundColor: 'transparent',
        height:200,
    },
    image: {
        width: width,
        height: 200,
    },
    avatar: {
        width: 130,
        height: 130,
        borderRadius: 63,
        borderWidth: 4,
        borderColor: "white",
        marginBottom:10,
        alignSelf:'center',
        position: 'absolute',
        marginTop:130
    },
    body:{
        marginTop:40,
    },
    bodyContent: {
        flex: 1,
        alignItems: 'center',
        padding:30,
    },
    name:{
        fontSize:28,
        color: "#696969",
        fontWeight: "600"
```

```
    },
    info:{
        fontSize:16,
        color: colors.tabIconSelected,
        marginTop:10
    },
});

const mapStateToProps = state => {
    return {
        uid: state.auth.id,
        cart: state.cart.products
    }
};

export default connect(mapStateToProps, null)(AccountHome);
```

9.2.5 Shop

```
import React from 'react';
import {View, Text, FlatList, StyleSheet, ScrollView, TouchableOpacity, AsyncStorage,
RefreshControl, ActivityIndicator} from 'react-native';
import { HeaderBackButton } from 'react-navigation';
import {Container, Content, Button, Icon, Picker} from 'native-base';
import {onApiTokenValid} from "../../utils/api";
import ProductList from "./components/ProductList";
import colors from '../../constants/Colors';
import {MenuIcon, NotificationIcon} from "../../components/HeaderIcons";
import {connect} from "react-redux";

class ShopHome extends React.Component {

    static navigationOptions = ({navigation}) => ({
        title: 'Shop',
        headerLeft: (
            <MenuIcon openDrawer={navigation.openDrawer} />
        ),
        headerRight: (
            <NotificationIcon navigate={navigation.navigate}
countCart={navigation.getParam('countCart','0')} />
        ),
    });

    constructor (props) {
        super(props);
        this.state = {
            selected: undefined,
            GridColumnsValue: false,
            ButtonType: 'Feather',
            ButtonName: 'list',
            products: [],
            meta: [],
            token: false,
            refreshing: true,
```

```
            fetching_from_server: false
        };
        this.offset = 1;
    }

    componentDidMount(){
        AsyncStorage.getItem("jewelshop:token", (error, exists) => {
            this.setState(prevState => ({
                ...prevState,
                token: exists
            }));
            this._onApiCall();
        });
    }

    componentWillMount() {
        // const {setParams} = this.props.navigation;
        this.props.navigation.setParams({countCart: this.props.cart.length});

        this.didFocusCart = this.props.navigation.addListener('didFocus', () => {
            console.log('didFocus');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });

        this.willFocusCart = this.props.navigation.addListener('willFocus', () => {
            console.log('willFocus');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });

        this.didBlurCart = this.props.navigation.addListener('didBlur', () => {
            console.log('didBlur');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });

        this.willBlurCart = this.props.navigation.addListener('willBlur', () => {
            console.log('willBlur');
            this.props.navigation.setParams({countCart: this.props.cart.length});
        });
    }

    componentWillUnmount() {
        this.didFocusCart.remove();
        this.willFocusCart.remove();
        this.didBlurCart.remove();
        this.willBlurCart.remove();
    }

    _onRefresh = () => {
        this.setState(prevState => ({
            ...prevState,
            refreshing: true,
        }));
        this.offset = 1;
        this._onApiCall()
            .then(() => {
```

```
                this.setState(prevState => ({
                    ...prevState,
                    refreshing: false,
                }));
            })
            .catch((error) => {
                console.error(error);
            });
    };

    _onApiCall = () => {
        return new Promise((resolve, reject) => {
            const url = 'https://api.jewelshop.tk/api/v1/products?page='+this.offset;
            fetch(url, {
                method: 'GET',
                headers: {
                    'Authorization': "Bearer " + this.state.token,
                    Accept : 'application/json',
                    'Content-Type': 'application/json'
                },
            })
            .then(response => response.json())
            .then((responseJson)=> {
                if(responseJson["status_code"] === 401){
                    onApiTokenValid().then(()=>{
                        this.setState(prevState => ({
                            ...prevState,
                            refreshing: false,
                            fetching_from_server: false,
                        }))
                    });
                } else {
                    this.setState(prevState => ({
                        ...prevState,
                        refreshing: false,
                        fetching_from_server: false,
                        products: responseJson["data"],
                        meta: responseJson["meta"],
                    }));
                    this.offset = this.offset + 1;
                    resolve(true);
                }
            })
            .catch(error => {
                console.log(error);
                reject(false);
            }) //to catch the errors if any
        })
    };

    loadMoreData = () => {
        //On click of Load More button We will call the web API again
        this.setState({ fetching_from_server: true }, () => {
            const url = 'https://api.jewelshop.tk/api/v1/products?page='+this.offset;
            fetch(url, {
```

```
                method: 'GET',
                headers: {
                    'Authorization': "Bearer " + this.state.token,
                    Accept : 'application/json',
                    'Content-Type': 'application/json'
                },
            })
                .then(response => response.json())
                .then((responseJson)=> {
                    if(responseJson["status_code"] === 401){
                        onApiTokenValid().then(()=>{
                            this.setState(prevState => ({
                                ...prevState,
                                refreshing: false,
                                fetching_from_server: false,
                            }))
                        });
                    } else {
                        this.offset = this.offset + 1;
                        this.setState(prevState => ({
                            ...prevState,
                            refreshing: false,
                            fetching_from_server: false,
                            products: [...this.state.products,
...responseJson["data"]],
                        }))
                    }
                })
                .catch(error => {
                    console.log(error);
                }) //to catch the errors if any
        });
    };
    onValueChange(value) {
        this.setState({
            selected: value
        });
    }

    ChangeGridValueFunction = () => {
        if(this.state.GridColumnsValue === true){
            this.setState(prevState => ({
                ...prevState,
                GridColumnsValue: false,
                ButtonType: 'Feather',
                ButtonName: 'list'
            }))
        } else {
            this.setState(prevState => ({
                ...prevState,
                GridColumnsValue: true,
                ButtonType: 'Feather',
                ButtonName: 'hash'
            }))
        }
```

```
    };

    _renderItem ({item, index}) {
        return (<ProductList
                    key={index}
                    data={item}
                    navigation={ this.props.navigation }
                />);
    }
    renderFooter() {
        let per_page = this.state.meta["pagination"]["per_page"];
        let total = this.state.meta["pagination"]["total"];
        let total_pages = this.state.meta["pagination"]["total_pages"];
        return (
            //Footer View with Load More button
            total > per_page  && this.offset <= total_pages ? (
            <View style={styles.footer}>
                <TouchableOpacity
                    activeOpacity={0.9}
                    onPress={this.loadMoreData}
                    //On Click of button calling loadMoreData function to load more
data
                    style={styles.loadMoreBtn}>
                    <Text style={styles.btnText}>Load More</Text>
                    {this.state.fetching_from_server ? (
                        <ActivityIndicator color="white" style={{ marginLeft: 8 }} />
                    ) : null}
                </TouchableOpacity>
            </View>) : null
        );
    }

    productFunc (data) {
        return (
            <FlatList
                data={data}
                renderItem={this._renderItem.bind(this)}
                ListFooterComponent={this.renderFooter.bind(this)}
                keyExtractor={(item, index) => index.toString()}
            />
        );
    }
    render() {
        const all_products =  this.productFunc(this.state.products);
        return (
            <Container>
                {!this.state.refreshing ? (
                    <ScrollView
                        refreshControl={
                            <RefreshControl
                                refreshing={this.state.refreshing}
                                onRefresh={this._onRefresh}
                                enabled={true}
                                tintColor="#fff" titleColor="#fff"
colors={[colors.tabIconSelected]}
```

```
                            />
                        }>
                    <Content style={{backgroundColor: '#e4e3eb'}}>
                        <ScrollView style={ styles.containerScrollView }>
                            <View style={{
                                flex: 1,
                                flexDirection: 'row',
                                justifyContent: 'space-between',
                            }}>
                                <View style={{backgroundColor: "#fff",width: 55,
height: 50}}>
                                    <TouchableOpacity
                                        onPress={() => this.ChangeGridValueFunction}>
                                        <Button transparent>
                                            <Icon
                                                active
                                                style={{fontSize: 22, color:
'black'}}
                                                type={this.state.ButtonType}
                                                name={this.state.ButtonName} />
                                        </Button>
                                    </TouchableOpacity>
                                </View>
                                <View style={{backgroundColor: "#fff",width: 55,
height: 50}}>
                                    <Button transparent>
                                        <Icon
                                            active
                                            style={{fontSize: 20, color: 'black'}}
                                            type="SimpleLineIcons"
                                            name="chart" />
                                    </Button>
                                </View>
                                <View style={{backgroundColor: "#fff",width: 220,
height: 50}}>
                                    <Picker
                                        mode="dropdown"
                                        iosIcon={<Icon name="arrow-down" />}
                                        selectedValue={this.state.selected}
                                        onValueChange={this.onValueChange.bind(this)}
                                    >
                                        <Picker.Item label="Select Order"
value={null} />
                                        <Picker.Item label="Date Added: Latest First"
value="newest" />
                                        <Picker.Item label="Price: Lower to Higher"
value="lower_price" />
                                        <Picker.Item label="Price: Higher to Lower"
value="higher_price" />
                                        <Picker.Item label="Product Name: A to Z"
value="AtoZ" />
                                        <Picker.Item label="Product Name: Z to A"
value="ZtoA" />
                                    </Picker>
                                </View>
```

```
                            </View>
                            { all_products }
                        </ScrollView>
                    </Content>
                    </ScrollView>
                    ) :
                    <ScrollView
                        refreshControl={
                            <RefreshControl
                                refreshing={this.state.refreshing}
                                enabled={true}
                                tintColor="#fff" titleColor="#fff"
colors={[colors.tabIconSelected]}
                            />
                        }>
                    </ScrollView>}
            </Container>
        );
    }
}

const styles = StyleSheet.create({
    containerScrollView: {
        flex:1,
        paddingLeft: 10,
        paddingRight: 10,
        paddingTop: 10
    },
    containerActions: {
        flex: 1, flexDirection: 'row', justifyContent: 'flex-end'
    },
    loadMoreBtn: {
        padding: 10,
        backgroundColor: colors.tabIconSelected,
        borderRadius: 6,
        flexDirection: 'row',
        justifyContent: 'center',
        alignItems: 'center',
        marginBottom: 10
    },
    btnText: {
        color: 'white',
        fontSize: 15,
        textAlign: 'center',
    },
});

const mapStateToProps = state => {
    return {
        cart: state.cart.products
    }
};

export default connect(mapStateToProps)(ShopHome);
```

9.2.6 Shopping Cart

```
import React, {Component} from 'react';
import {
    View,
    Text,
    StyleSheet,
    ScrollView,
    Platform,
    FlatList, Alert,
} from 'react-native';
import {Content, Container, CardItem, Button, Icon, Card} from 'native-base'
import DropDownItem from 'react-native-drop-down-item';
import CartList from "./components/CartList";
import {connect} from "react-redux";
import {HeaderBackButton} from "react-navigation";
import {numberWithCommas} from "../../utils/helpers";
import {CartIconTrash} from "../../components/HeaderIcons";
import {emptyCart} from "../shop/store";
import {validate, validateErrorMessages} from "../../utils/validation";
import {Input} from "react-native-elements";
import colors from "../../constants/Colors";


class CartHome extends Component {

    static navigationOptions = ({navigation}) => ({
        title: 'Shopping Cart',
        headerLeft: <HeaderBackButton onPress={() => navigation.goBack(null)}/>,
        headerRight: <CartIconTrash
                        emptyCart={navigation.getParam('emptyCart',() =>
console.log('error parse emptyCart function'))}
                        countCart={navigation.getParam('countCart','0')}/>,
    });

    constructor (props) {
        super(props);
        this.state = {
            contents: [
                {
                    title: 'Title 1',
                    body: 'Hi. I love this component. What do you think?',
                }
            ],

            form: {
                coupon: {
                    value: "",
                    valid: false,
                    validationRules: {
                        minLength: 4,
                        notEmpty: true
                    },
                    validationErrorMessage: '',
                    touched: false,
```

```
                }
            }
        };
    }

    componentWillMount() {
        this.props.navigation.setParams({emptyCart: this.props.onEmptyCart});
        this.props.navigation.setParams({countCart: this.props.cart.length});
    }

    updateInputState = (key, value) => {
        let connectedValue = {};
        if (this.state.form[key].validationRules.equalTo) {
            const equalValidate = this.state.form[key].validationRules.equalTo;
            const equalValue = this.state.form[equalValidate].value;
            connectedValue = {
                ...connectedValue,
                equalTo: equalValue
            };
        }
        this.setState(prevState => {
            return {
                ...prevState,
                form: {
                    ...prevState.form,
                    [key]: {
                        ...prevState.form[key],
                        value: value,
                        valid: validate(
                            value,
                            prevState.form[key].validationRules,
                            connectedValue
                        ),
                        validationErrorMessage: validateErrorMessages(value,
prevState.form[key].validationRules, connectedValue),
                        touched: true
                    }
                },
            };
        });
    };

    _renderItem ({item, index}) {
        return (
            <CartList
                key={index}
                data={item}
                navigation={ this.props.navigation }
            />
        );
    }

    shoppingCart (data) {
        return (
            this.props.cart.length > 0 ?
```

```
                <FlatList
                    data={data}
                    renderItem={this._renderItem.bind(this)}
                    keyExtractor={(item, index) => index.toString()}
                />
                :
                <Card>
                    <CardItem bordered>
                        <Text>No Item(s) in Shopping Cart</Text>
                    </CardItem>
                </Card>
        );
    }

    render() {
        return (
            <Container>
                <Content>
                    <ScrollView style={ styles.containerScrollView }>

                        {this.shoppingCart(this.props.cart)}

                        {this.props.cart.length > 0 ?
                            <Text>Price Details </Text>
                        : null}
                        {this.props.cart.length > 0 ?
                            <Card>
                                <CardItem header bordered>
                                    <View style={styles.spaceBetween}>
                                        <Text>Order Subtotal:</Text>

<Text>{numberWithCommas(this.props.cartSubTotal)}</Text>
                                    </View>
                                </CardItem>
                                <CardItem header bordered>
                                    <View style={styles.spaceBetween}>
                                        <Text>Order Tax:</Text>

<Text>{numberWithCommas(this.props.cartTax)}</Text>
                                    </View>
                                </CardItem>
                                <CardItem header bordered>
                                    <View style={styles.spaceBetween}>
                                        <Text>Order Total:</Text>

<Text>{numberWithCommas(this.props.cartTotal)}</Text>
                                    </View>
                                </CardItem>
                            </Card>
                        : null}
                        {this.props.cart.length > 0 ?
                            <Button
                                block
                                warning
                                style={{marginBottom: 20}}
```

```
                                onPress={() => {
                                    Alert.alert(
                                        'Checkout',
                                        'Are you sure you want checkout your
products?',
                                        [
                                            {
                                                text: 'Cancel',
                                                onPress: () => console.log('Cancel
Pressed'),
                                                style: 'cancel',
                                            },
                                            {
                                                text: 'OK',
                                                onPress: () => {

this.props.navigation.navigate("Address");
                                                }
                                            },
                                        ],
                                        {cancelable: false},
                                    );
                                }}>
                                <Text>Place Order</Text>
                            </Button>
                        : null}
                    </ScrollView>
                </Content>
            </Container>
        );
    }
}

const styles = StyleSheet.create({
    containerScrollView: {
        flex:1,
        paddingLeft: 15,
        paddingRight: 15,
        paddingTop: 15
    },
    spaceBetween: {
        flex: 1, flexDirection: 'row', justifyContent: 'space-between'
    },
    header: {
        width: '100%',
        paddingVertical: 8,
        paddingHorizontal: 12,
        flexWrap: 'wrap',
        flexDirection: 'row',
        alignItems: 'center',
    },
    headerTxt: {
        fontSize: 12,
        color: 'rgb(74,74,74)',
        marginRight: 60,
```

```
        flexWrap: 'wrap',
    },
    txt: {
        fontSize: 14,
    },
});

const mapStateToProps = state => {
    return {
        cart: state.cart.products,
        cartSubTotal: state.cart.subTotal,
        cartTax: state.cart.tax,
        cartTotal: state.cart.total
    }
};

const mapDispatchToProps = dispatch => {
    return {
        onEmptyCart: ()  => dispatch(emptyCart()),
    }
};

export default connect(mapStateToProps,mapDispatchToProps)(CartHome);
```

9.2.7 Stripe Checkout

```
import React, { Component } from 'react';
import {HeaderBackButton} from "react-navigation";
import {connect} from "react-redux";
import StripeCheckout from "./components/StripeCheckout"
import {toastr} from "../../components/Toast";
import {emptyCart} from "../shop/store";

class StripeHome extends Component {

    static navigationOptions = ({ navigation }) => ({
        title: "Payment Stripe",
        headerLeft: <HeaderBackButton onPress={() => navigation.goBack(null)}/>,
    });

    onPaymentSuccess = (token) => {
        console.log(token);
        console.log(this.props.cart);
        // send the stripe token to your backend!
        const url = 'https://api.jewelshop.tk/api/v1/charge';
        fetch(url, {
            method: 'POST',
            headers: {
                Accept : 'application/json',
                'Content-Type': 'application/json',
                'Authorization': "Bearer " + this.props.token,
            },
            body: JSON.stringify({
                products: this.props.cart,
```

```
                stripeToken: token,
                user_id: this.props.uid,
                subtotal: this.props.cartSubTotal,
                tax: this.props.cartTax,
                total: this.props.cartTotal,
                addresses: this.props.addresses,
                delivery: this.props.delivery,
                payment: 'Stripe'
            }),
        })
            .then(response => response.json())
            .then((responseJson)=> {
                console.log(responseJson);
                if(responseJson["status_code"] === 204 || responseJson["status_code"]
=== 404 || responseJson["status_code"] === 500){

toastr.showToast(responseJson["message"],'warning','bottom','center',null,2500);
                } else {
                    this.props.navigation.navigate("Confirm");

toastr.showToast(responseJson["message"],'success','bottom','center',null,2500);
                    this.props.onEmptyCart();
                }
            })
            .catch(error => {
                toastr.showToast('No Internet Connection, Please connect to internet
and try again!','warning','bottom','left','Okey',2500);
                console.log(error);
            })
    };

    onClose = () => {
        this.props.navigation.navigate("Payment")
    };

    render() {
        return (

            <StripeCheckout
                publicKey="pk_test_AmF3jzei7BUbdYiGC3x2TKKY"
                amount={this.props.cartTotal}

imageUrl="https://www.jewelshop.tk/storage/6nGs3UB4AjWftTbVcehjrZZakBSWl17xjjxYhuFw.p
ng"
                storeName="JewelShop"
                description="Stripe Development"
                currency="EUR"
                allowRememberMe={true}
                shippingAddress={true}
                prepopulatedEmail={this.props.addresses.email}
                style={{justifyContent: 'center'}}
                onClose={this.onClose}
                onPaymentSuccess={this.onPaymentSuccess}
            />
```

```
        );
    }
}

const mapStateToProps = state => {
    return {
        uid: state.auth.id,
        token: state.auth.token,
        cart: state.cart.products,
        cartSubTotal: state.cart.subTotal,
        cartTax: state.cart.tax,
        cartTotal: state.cart.total,
        addresses: state.address.address_user,
        delivery: state.delivery.delivery
    }
};

const mapDispatchToProps = dispatch => {
    return {
        onEmptyCart: ()  => dispatch(emptyCart()),
    }
};

export default connect(mapStateToProps,mapDispatchToProps)(StripeHome);
```

9.2.8 All Orders

```
import React, { Component } from 'react';
import {
    View,
    StyleSheet,
    AsyncStorage,
    FlatList,
    ScrollView,
    RefreshControl,
    TouchableOpacity,
    ActivityIndicator
} from 'react-native';
import {Container, Content, Text, Card, CardItem} from 'native-base';
import {HeaderBackButton} from "react-navigation";
import {onApiTokenValid} from "../../utils/api";
import OrderListItem from "../orders/components/OrderListItem";
import {connect} from "react-redux";
import colors from "../../constants/Colors";

class OrdersHome extends Component {

    static navigationOptions = ({ navigation }) => ({
        title: "Your Orders",
        headerLeft: <HeaderBackButton onPress={() => navigation.goBack(null)}/>,
    });

    constructor (props) {
        super(props);
```

```
        this.state = {
            orders: [],
            meta: [],
            refreshing: true,
            token: false,
            fetching_from_server: false
        };
        this.offset = 1;
    }

    componentDidMount(){
        AsyncStorage.getItem("jewelshop:token", (error, exists) => {
            this.setState(prevState => ({
                ...prevState,
                token: exists
            }));
            this._onApiCall();
        });
    }

    _onRefresh = () => {
        this.setState(prevState => ({
            ...prevState,
            refreshing: true,
        }));
        this.offset = 1;
        this._onApiCall()
            .then(() => {
                this.setState(prevState => ({
                    ...prevState,
                    refreshing: false,
                }));
            })
            .catch((error) => {
                console.error(error);
            });
    };

    _onApiCall = () => {
        let user_id = this.props.uid;
        return new Promise((resolve, reject) => {
            const url =
'https://api.jewelshop.tk/api/v1/user/'+user_id+'/orders?page='+this.offset;
            fetch(url, {
                method: 'GET',
                headers: {
                    Accept : 'application/json',
                    'Content-Type': 'application/json',
                    'Authorization': "Bearer " + this.state.token,
                },
            })
                .then(response => response.json())
                .then((responseJson)=> {
                    if(responseJson["status_code"] === 401 ||
responseJson["status_code"] === 500){
```

```
                        onApiTokenValid()
                            .then((done)=> {
                                if(done){
                                    this.setState(prevState => ({
                                        ...prevState,
                                        refreshing: false,
                                        fetching_from_server: false,
                                    }));
                                    resolve(true);
                                }
                            })
                            .catch(() => {
                                console.log(error)
                                reject(false);
                            });
                    } else {
                        this.setState(prevState => ({
                            ...prevState,
                            refreshing: false,
                            fetching_from_server: false,
                            orders: responseJson["data"],
                            meta: responseJson["meta"],
                        }));
                        this.offset = this.offset + 1;
                        resolve(true);
                    }
                })
                .catch(error => {
                    console.log(error);
                    reject(false);
                }) //to catch the errors if any
        })
    };

    loadMoreData = () => {
        //On click of Load More button We will call the web API again
        let user_id = this.props.uid;
        this.setState({ fetching_from_server: true }, () => {
            const url =
'https://api.jewelshop.tk/api/v1/user/'+user_id+'/orders?page='+this.offset;
            fetch(url, {
                method: 'GET',
                headers: {
                    'Authorization': "Bearer " + this.state.token,
                    Accept : 'application/json',
                    'Content-Type': 'application/json'
                },
            })
                .then(response => response.json())
                .then((responseJson)=> {
                    if(responseJson["status_code"] === 401){
                        onApiTokenValid().then(()=>{
                            this.setState(prevState => ({
                                ...prevState,
                                refreshing: false,
```

```
                            fetching_from_server: false,
                        }))
                    });
                } else {
                    this.offset = this.offset + 1;
                    this.setState(prevState => ({
                        ...prevState,
                        refreshing: false,
                        fetching_from_server: false,
                        orders: [...this.state.orders, ...responseJson["data"]],
                    }))
                }
            })
            .catch(error => {
                console.log(error);
            }) //to catch the errors if any
        });
    };

    renderFooter() {
        let per_page = this.state.meta["pagination"]["per_page"];
        let total = this.state.meta["pagination"]["total"];
        let total_pages = this.state.meta["pagination"]["total_pages"];
        return (
            //Footer View with Load More button
            total > per_page  && this.offset <= total_pages ? (
            <View style={styles.footer}>
                <TouchableOpacity
                    activeOpacity={0.9}
                    onPress={this.loadMoreData}
                    //On Click of button calling loadMoreData function to load more
data
                    style={styles.loadMoreBtn}>
                    <Text style={styles.btnText}>Load More</Text>
                    {this.state.fetching_from_server ? (
                        <ActivityIndicator color="white" style={{ marginLeft: 8 }} />
                    ) : null}
                </TouchableOpacity>
            </View>) : null
        );
    }

    _renderItem ({item, index}) {
        return (
            <OrderListItem
                key={index}
                data={item}
                navigation={ this.props.navigation }
                _onRefresh={this._onRefresh}
            />
        );
    }

    Orders (data) {
        return (
```

```
            this.state.orders.length > 0 ?
                <FlatList
                    data={data}
                    renderItem={this._renderItem.bind(this)}
                    ListFooterComponent={this.renderFooter.bind(this)}
                    keyExtractor={(item, index) => index.toString()}
                />
                :
                <Card>
                    <CardItem bordered>
                        <Text>No Item(s) in Orders</Text>
                    </CardItem>
                </Card>
        );
    }

    render() {
        const orders = this.Orders(this.state.orders);
        return (
            <Container>
                {!this.state.refreshing ? (
                        <ScrollView
                            refreshControl={
                                <RefreshControl
                                    refreshing={this.state.refreshing}
                                    onRefresh={this._onRefresh}
                                    enabled={true}
                                    tintColor="#fff" titleColor="#fff"
colors={[colors.tabIconSelected]}
                                />
                            }>
                            <Content>
                                <ScrollView style={ styles.containerScrollView }>
                                    {orders}
                                </ScrollView>
                            </Content>
                        </ScrollView>
                    ):
                    <ScrollView
                        refreshControl={
                            <RefreshControl
                                refreshing={this.state.refreshing}
                                enabled={true}
                                tintColor="#fff" titleColor="#fff"
colors={[colors.tabIconSelected]}
                            />
                        }>
                    </ScrollView>
                }
            </Container>
        );
    }
}

const styles = StyleSheet.create({
```

```
    containerScrollView: {
        flex:1,
        paddingLeft: 15,
        paddingRight: 15,
        paddingTop: 15
    },
    loadMoreBtn: {
        padding: 10,
        backgroundColor: colors.tabIconSelected,
        borderRadius: 6,
        flexDirection: 'row',
        justifyContent: 'center',
        alignItems: 'center',
        marginBottom: 10
    },
    btnText: {
        color: 'white',
        fontSize: 15,
        textAlign: 'center',
    },
});

const mapStateToProps = state => {
    return {
        uid: state.auth.id
    }
};

export default connect(mapStateToProps, null)(OrdersHome);
```

Βιβλιογραφία

- ✓ O'Brien, J A. (2003). Introduction to information systems: essentials for the e-business enterprise. McGraw-Hill, Boston, MA
- ✓ Jon Duckett (2014). Web Design with HTML, CSS, JavaScript and jQuery Set.
- ✓ Jennifer Robbins. (2018). Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics.
- ✓ Matt Stauffer. (2019). Laravel: Up & Running: A Framework for Building Modern PHP Apps.
- ✓ Josh Lockhart. (2015). Modern PHP: New Features and Good Practices.
- ✓ Karl Wiegers, Joy Beatty. (2013). Software Requirements (3rd Edition) (Developer Best Practices).
- ✓ Fernando Monteiro. (2016). Hands-On Full Stack Web Development with Angular 6 and Laravel 5: Become fluent in both frontend and backend web development with Docker, Angular and Laravel.
- ✓ Adel F (2016). Architecture of complex web applications: With examples in Laravel(PHP).
- ✓ Phil Sturgeon and Laura Bohill. (2015). Build APIs You Won't Hate: Everyone and their dog wants an API, so you should probably learn how to build them.
- ✓ Robin Nixon. (2018). Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning PHP, MYSQL, Javascript, CSS & HTML5).
- ✓ Jamie Chan. (2018). SQL: Learn SQL (using MySQL) in One Day and Learn It Well. SQL for Beginners with Hands-on Project. (Learn Coding Fast with Hands-On Project).
- ✓ Jeffrey Shapiro (Author), Guy Yardeni (Author), Omar Droubi (Author), Michael Noel (Author), Andrew Abbate (Author), Chris Amaris (Author). (2013). Windows Server 2016 Unleashed (includes Content Update Program).
- ✓ Jon Loeliger and Matthew McCullough. (2012). Version Control with Git: Powerful tools and techniques for collaborative software development.
- ✓ Brent Laster (2016). Professional Git.
- ✓ Tanner Larsson. (2016). Ecommerce Evolved: The Essential Playbook To Build, Grow & Scale A Successful Ecommerce Business.
- ✓ Bonnie Eisenman. (2017). Learning React Native: Building Native Mobile Apps with JavaScript.
- ✓ Michele Bertoli . (2016). React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns.
- ✓ Alex Banks and Eve Porcello. (2017). Learning React: Functional Web Development with React and Redux.