Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

Προσωποποιημένη παροχή υπηρεσιών
Personalized web service provision

Διπλωματική Εργασία

Μικιρντιτσιάν Βικέν

Α’ Επιβλέπων
Δρ. Ασπασία Δασκαλοπούλου
Επίκουρη Καθηγήτρια

Β’ Επιβλέπων
Δρ. Χρήστος Σωτηρίου
Αναπληρωτής Καθηγητής

Βόλος, 2019

(κενή σελίδα)

Ευχαριστίες,

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω την βασική επιβλέποντα αυτής, Δόκτωρ Ασπασία Δασκαλοπούλου, για την υποστήριξη, την εμπιστοσύνη και τις πολύτιμες συμβουλές που μου παρείχε.

Επίσης, ευχαριστώ θερμά όλους τους καθηγητές με τους οποίους είχα την τιμή να συνεργαστώ όλα αυτά τα χρόνια και με βοήθησαν, ο καθένας με τον δικό τους μοναδικό τρόπο, να κατανοήσω την επιστήμη και τον τρόπο σκέψης ενός σύγχρονου μηχανικού υπολογιστών.

Τέλος, θα ήθελα να ευχαριστήσω θερμά την οικογένεια και τους φίλους που στάθηκαν δίπλα μου κατά την διάρκεια εκπόνησης της εργασίας και των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Καθώς το διαδίκτυο εισχωρεί ολοένα και περισσότερο στην κοινωνία μας, οι μηχανικοί προσπαθούν με κάθε μέσο και τεχνολογία που διαθέτουν, να δημιουργήσουν τις κατάλληλες υποδομές που θα διευκολύνουν τον τρόπο ζωής των ανθρώπων. Αριθμοί που τρομάζουν, εφαρμογές που ξεπερνούν τα τρία εκατομμύρια και ιστοσελίδες που έσπασαν προ πολλού το φράγμα του ενός δισεκατομμυρίου παγκοσμίως, κατακλύζουν το διαδίκτυο παρέχοντας απεριόριστες δυνατότητες και γνώσεις στα δισεκατομμύρια των χρηστών του.

Την τελευταία δεκαετία η κοινωνία στρέφεται και στις διαδικτυακές αγορές. Για κάθε νέα τεχνολογία στο διαδίκτυο έγκεινται επιμέρους εφαρμογές προς αξιοποίησή της. Για τον λόγο αυτό, αποφάσισα να ξεκινήσω τη δημιουργία μίας δυναμικής διαδικτυακής εφαρμογής η οποία θα έχει ως κύριο σκοπό την διευκόλυνση των χρηστών της στην αγορά ειδών ρουχισμού μέσω του διαδικτύου.

Σε αυτή την εργασία, θα αναδείξω τις τεχνικές, τα συστήματα και τον τρόπο με τον οποίο αναπτύχθηκε και σχεδιάστηκε η εν λόγω διαδικτυακή εφαρμογή. Θα ακολουθήσω την πορεία των δεδομένων από τις βάσεις δεδομένων έως την προβολή τους στον φυλλομετρητή του χρήστη και αντίστροφα, και θα παρουσιάσω τα βήματα και την λογική που χρησιμοποιήθηκε σε όλο το εύρος της εφαρμογής. Ταυτόχρονα, θα παρουσιάζω κομμάτια από τον κώδικα της εφαρμογής, τα οποία θα βασίζονται στις κυρίως τέσσερεις γλώσσες προγραμματισμού διαδικτυακών εφαρμογών (php, html, css και javascript).

Λέξεις κλειδιά:

Εφαρμογή, διαδίκτυο, σχεδιασμός και ανάπτυξη εφαρμογής, php, html, css, javascript

ABSTRACT

As the internet is increasingly penetrating our society, engineers try to create the appropriate infrastructure to make people's way of life easier by all means and technology. Scary numbers, over three million applications and web pages that have passed the one billion barrier globally, have flooded the web by providing unlimited possibilities and knowledge to billions of users.

Over the last decade, society is also turning to online shopping. For each new technology on the internet, there are individual applications to be exploited. For this reason, I decided to start creating a dynamic application that will primarily aim to facilitate its users in the purchase of clothing items over the Internet.

In this paper, I will highlight the techniques, the systems and the way in which this web application was developed and designed. I will follow the path of the data, from the databases to their display in the user's browser and vice versa and I will show off the steps and logic used throughout the application. At the same time, I will present snippets of code from the application, which will be based on the listed four main programming languages for web applications (php, html, css and javascript).

Keywords:

Application, internet, online, application design and development, php, html, css, javascript

Περιεχόμενα

Κατάλογος εικόνων

Κεφάλαιο 1ο
Εισαγωγή

1.1. Σκοπός Εργασίας

Η διαδικτυακή εφαρμογή που θα αναλύσω στα πλαίσια της εργασίας, θα αλληλοσυνδέει και θα καλύπτει τα κενά στο τρίγωνο εταιρία - προϊόν - πελάτη και αφορά την βελτιστοποιημένη και στοχευμένη διαδικτυακή αγορά ειδών ένδυσης. Έχει ως στόχο να προσωποποιήσει την αγορά των παραπάνω ειδών, και ως αυτού, να επιφέρει ραγδαία μείωση των επιστροφών των προϊόντων η οποία θα αποφέρει, όχι μόνο οικονομικά οφέλη, αλλά και καλύτερη επικοινωνία, εξυπηρέτηση και διαδικτυακή εμπειρία στον πελάτη και την εκάστοτε εταιρία λιανικής πώλησης που θα χρησιμοποιεί την εφαρμογή.

Ταυτόχρονα, η ανάλυση της εργασίας έχει ως βασικό σκοπό να αναδείξει όλα τα αναγκαία βήματα που απαιτούνται για τον πλήρη σχεδιασμό και την ανάπτυξη μία σύγχρονης διαδικτυακής εφαρμογής, όπως η παραπάνω, χρησιμοποιώντας ως βασικό περιβάλλον χρήστη τους φυλλομετρητές.

Τέλος, ως διαδικτυακή εφαρμογή θα πρέπει να επισημάνουμε πως στην συγκεκριμένη εργασία αναφερόμαστε σε μία πλατφόρμα χρήστη, μία πλατφόρμα εταιριών, ένα διαδικτυακό γραφικό στοιχείο (widget) και δύο Διασυνδέσεις προγραμματισμού εφαρμογών (API - Application Programming Interface) οι οποίες αλληλοσυνδέουν και βοηθούν στην μετάδοση πληροφορίας ανάμεσα στα τρία παραπάνω στοιχεία.

1.2. Κίνητρο

1.2.1. Θεωρητικό Κίνητρο

Το βασικό κίνητρο για την εκπόνηση της εργασίας αυτής, είναι οι δαιδαλώδες και συνεχείς εναλλαγές στα εργαλεία και στους δεκάδες τρόπους υλοποίησης διαδικτυακών εφαρμογών. Θα ήθελα να αναπτύξω και να παρουσιάσω τον τρόπο με τον οποίο ένα σύνολο εργαλείων και γλωσσών προγραμματισμού είναι ικανές να συμβάλλουν στον αρχικό και βασικό σχεδιασμό μίας πολυδιάστατης και δη πολύπλοκης διαδικτυακής εφαρμογής.

Χρησιμοποιώντας λοιπόν τα κατάλληλα εφόδια που μου παρείχε η σχολή, καθώς και τις γνώσεις με τον σχεδιασμό και την ανάπτυξη ιστοσελίδων όπου είχα την ευκαιρία να υλοποιήσω στο παρελθόν, η συγκεκριμένη ιδέα, μου δίνει την δυνατότητα να παρουσιάσω μία ολοκληρωμένη δομή και λειτουργία μίας διαδικτυακής εφαρμογής.

1.2.2. Η ιδέα σαν κίνητρο

Αρχικά θα ήθελα να αναφέρω πως η ιδέα της εφαρμογής πρωτοεμφανίστηκε το 2015 και όπως πολλές άλλες είχε μπει αρκετά χρόνια στο συρτάρι και ήταν έτοιμη να ξεχαστεί. Με τον καιρό όμως παρατήρησα πως ολοένα και αυξανόταν το πρόβλημα της επιστροφής των ρούχων τα οποία ένας πελάτης είχε αγοράσει διαδικτυακά. Οι περισσότερες εταιρίες λιανικού εμπορίου αδυνατούσαν και αδυνατούν ακόμη και σήμερα στο να παρέχουν τα κατάλληλα διαδικτυακά εργαλεία, ώστε να πετυχαίνουν χωρίς να δυσφορεί ο πελάτης τους το σωστό μέγεθος ενός ρούχου.

Κάθε εταιρία παρουσιάζει τους δικούς της κανόνες στον τρόπο μέτρησης των μεγεθών των προϊόντων της. Ορισμένες παραπέμπουν τον πελάτη τους στο να διαβάσουν τον τρόπο μέτρησης των ρούχων και των μεγεθών που διαθέτουν σε συνδέσμους ακόμη και εκτός της ιστοσελίδας του προϊόντος. Άλλες πιο ανταγωνιστικές, είτε κρατάνε τα δεδομένα τους από προηγούμενες αγορές, είτε δημιουργούν τα δικά τους widgets μέτρησης σωματικών διαστάσεων του πελάτη, ωθώντας τον τελευταίο να εισάγει τις σωματικές του διαστάσεις και να δίνει τα προσωπικά του δεδομένα σε κάθε εταιρία απ' όπου αγοράζει ένα είδος ένδυσης διαδικτυακά.

Παράλληλα, ο μεγάλος ανταγωνισμός σε μία αναπτυσσόμενη διαδικτυακή αγορά ωθεί τις εταιρίες στο να μην χρεώνουν τον πελάτη σε περίπτωση επιστροφής ή αλλαγής προϊόντος. Με αυτό τον τρόπο, επιβαρύνεται η εταιρία με μεταφορικά και όταν μιλάμε για ένα ποσοστό με μέσο όρο επιστροφών το 25% το οποίο φτάνει σε πολλές ιστοσελίδες υψηλής ραπτικής το 50% των ηλεκτρονικών αγορών [1], εύκολα καταλαβαίνουμε πως τα έξοδα στην εταιρία είναι κάθε άλλο παρά αμελητέα.

Βλέποντας τις ανεπιτυχείς προσπάθειες της αγοράς, τα νέα τεχνολογικά εργαλεία στον τομέα του σχεδιασμού ιστοσελίδων αλλά και θεωρώντας πως έχω την δυνατότητα να υλοποιήσω μία εφαρμογή που θα έχει ανταπόκριση στις ανάγκες των πελατών και των εταιριών, έχω ένα επιπλέον κίνητρο στην εκπόνηση της εφαρμογής.

1.3. Διάρθρωση Εργασίας

Στο σημείο αυτό, για την διευκόλυνση του αναγνώστη δίνεται μία αφηρημένη διάρθρωση της δομής της εργασίας.

Στο 2^{ο} κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο για την ανάπτυξη και τον σχεδιασμό της εφαρμογής αναφέροντας παράλληλα τα εργαλεία και τις τεχνικές που χρησιμοποιήθηκαν.

Στο 3^{ο} κεφάλαιο παρουσιάζεται η αρχιτεκτονική ανάπτυξη της εργασίας όπου αναλύονται διεξοδικά οι μέθοδοι που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

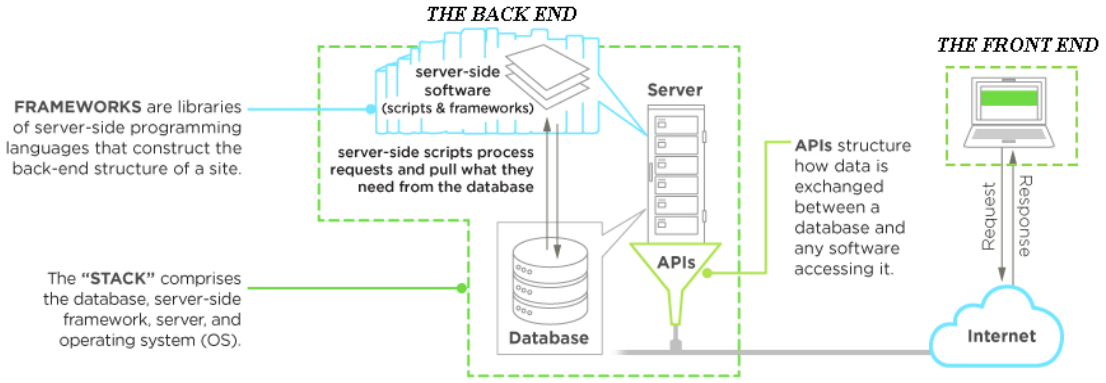
Το 4^{ο} κεφάλαιο αναφέρεται στον σχεδιασμό της εφαρμογής και στις τεχνικές που χρησιμοποιήθηκαν για την σχεδίαση και ανάπτυξη των επιμέρους σελίδων της.

Τέλος, στο 5^{ο} κεφάλαιο παρουσιάζονται συμπεράσματα και μελλοντικές ενέργειες ανάπτυξης και σχεδίασης της εφαρμογής.

Κεφάλαιο 2ο
Εργαλεία Ανάπτυξης και Σχεδιασμού

2.1. Εισαγωγή στα εργαλεία ανάπτυξης

Για την ομαλή ανάπτυξη και σχεδιασμό της εργασίας ώστε να μπορέσει να λειτουργήσει η εφαρμογή, έγκειται η χρήση διαφόρων εργαλείων, πλαισίων λογισμικού (**frameworks**) και γλωσσών προγραμματισμού. Πριν ξεκινήσω να τα αναφέρω ένα προς ένα, θα ήθελα να τα παρουσιάσω ως σύνολο για να αναδειχθεί ο τρόπος με τον οποίο συνδυάζεται το θεωρητικό υπόβαθρο ώστε να δημιουργηθούν οι κατάλληλες προδιαγραφές, όχι μόνο για την ομαλή λειτουργία της εφαρμογής, αλλά και για την μετέπειτα συντήρηση και αναβάθμισή της.



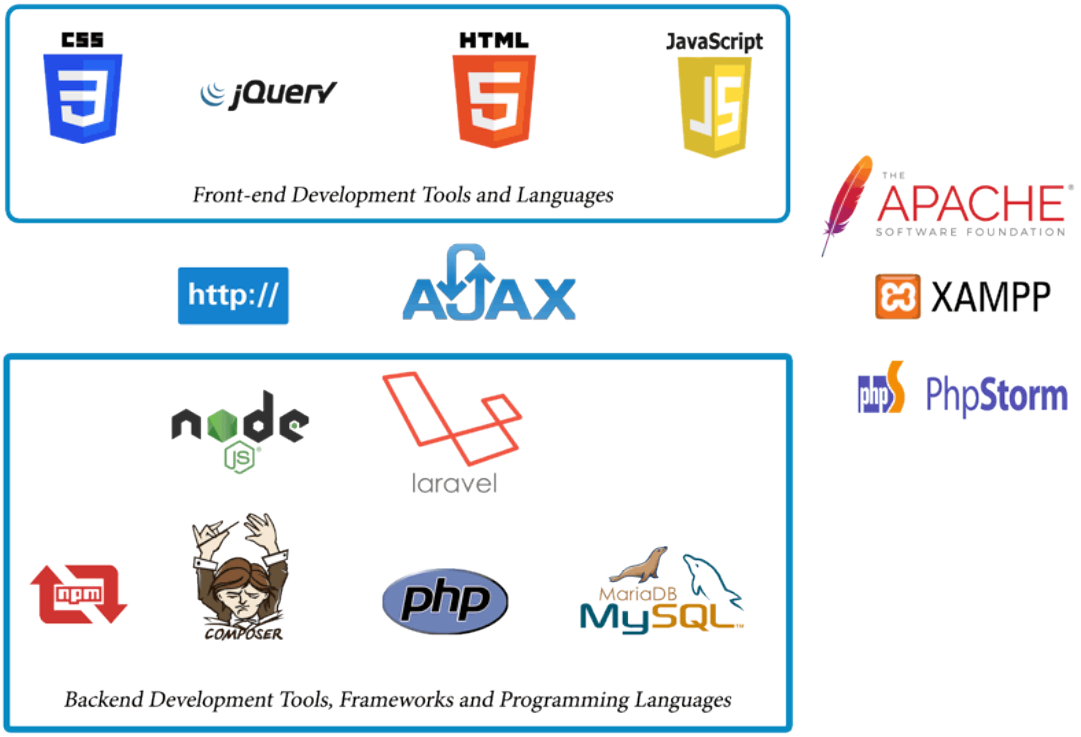
Εικόνα 2.1: Βασικές προδιαγραφές και συστήματα που απαιτούνται για μία διαδικτυακή εφαρμογή .

Στην εικόνα 2.1 γίνεται ένας βασικός διαχωρισμός ανάμεσα στις **backend** και τις **front-end** γλώσσες προγραμματισμού, εργαλεία, τεχνολογίες και πλαίσια λογισμικών (frameworks).

Ο όρος **backend** ή *server-side* στην κατασκευή διαδικτυακών εφαρμογών, περιλαμβάνει τρία βασικά κομμάτια. Το πρώτο είναι ο προγραμματισμός και η κατάλληλη διαμόρφωση του εξυπηρετητή (web-server), ο οποίος παρέχει δεδομένα και πακέτα κατόπιν αιτήματος, αναλόγως τα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται. Το δεύτερο αποτελεί τον πηγαίο κώδικα της ίδιας της εφαρμογής, όπου αναπτύσσονται και εκτελούνται οι βασικές λογικές λειτουργίες της. Τέλος, το τρίτο κομμάτι αποτελείται από τις βάσεις δεδομένων που οργανώνουν και αποθηκεύουν τα δεδομένα της εφαρμογής.

Ο όρος **front-end** ή εμπρόσθιο τμήμα προγράμματος, αφορά τον τελικό κώδικα που έχει αναπτυχθεί και είναι αυτός που δημιουργεί την αλληλεπίδραση του χρήστη με την ιστοσελίδα. Θεωρείται η «κορυφή του παγόβουνου» σε μία διαδικτυακή εφαρμογή και παρουσιάζει τα δεδομένα σε ανάλογη μορφή για την καλύτερη εμπειρία του χρήστη της (**UX** - *User Experience*).

Στην εικόνα 2.2 βλέπουμε τον παραπάνω διαχωρισμό ανάμεσα στα μέσα που χρησιμοποιήθηκαν για την περάτωση της εργασίας.



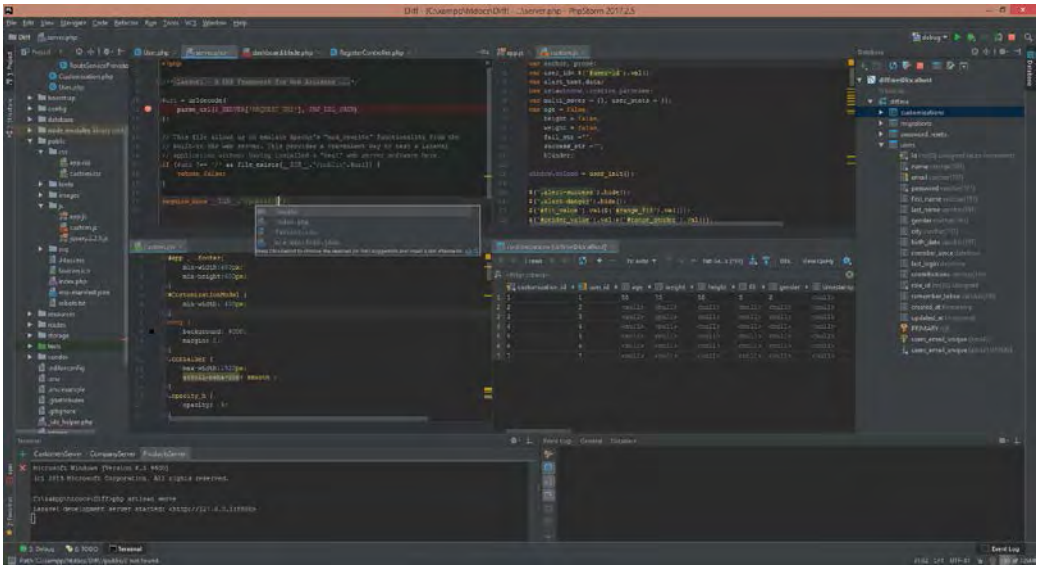
Εικόνα 2.2: το βασικό σύνολο βιβλιοθηκών, frameworks, λογισμικών και γλωσσών προγραμματισμού που χρησιμοποιήθηκαν.

2.2. Τεχνολογίες και εργαλεία

2.2.1. PhpStorm

Το **Phpstorm** [2] είναι το βασικό εργαλείο πάνω στο οποίο σχεδιάστηκαν οι πλατφόρμες για την λειτουργία της εφαρμογής. Είναι ένα διαπλατφορμικό (cross-platform) ολοκληρωμένο περιβάλλον ανάπτυξης (**IDE** – *integrated development environment*) και κατασκευάστηκε με βάση την πλατφόρμα intelliJ IDEA η οποία έχει γραφτεί σε JAVA από την εταιρία JetBrains. Παρέχει προγράμματα και συστήματα επεξεργασίας για τις γλώσσες PHP, HTML, Javascript, SQL και CSS.

Οι κύριοι και βασικοί λόγοι που οδήγησαν στην επιλογή του συγκεκριμένου εργαλείου είναι ότι παρουσιάζει σε πρώτο χρόνο ανάλυση κώδικα, πρόληψη και εντοπισμό συντακτικών σφαλμάτων. Ταυτόχρονα, έχει την δυνατότητα να επαναπροσδιορίζει και να εντοπίζει τυχόν εναλλαγές στο συντακτικό των γλωσσών PHP και Javascript (μετονομασίες, συγκρούσεις μεταβλητών κ.α.) σε όλο το φάσμα των αρχείων του προτζεκτ και παρέχει πλήρη υποστήριξη των ενεργειών της γλώσσας PHP από την έκδοση 5.3. Επίσης, έχει συστήματα υποστήριξης και επικοινωνίας με πολλές γλώσσες βάσεων δεδομένων μια εκ των οποίων και η MySQL και δυνατότητα για πολλαπλά παράθυρα γραμμών εντολών. Τέλος, έχει την δυνατότητα διαχείρισης πολλαπλών εξυπηρετητών, εύκολη λειτουργία αποσφαλμάτωσης των γλωσσών PHP (Zend Debugger, Xdebug) και Javascript, καθώς και εύκολη πρόσβαση για την εγκατάσταση πρόσθετων εργαλείων (plug-in). Πολλά απ' όσα περιγράφηκαν μπορούν να παρατηρηθούν στην εικόνα 2.3 όπου παρουσιάζονται διαφορετικές λειτουργίες και εργαλεία που μπορεί να συνδυάζει και να διευκολύνει την ανάπτυξη της εφαρμογής.

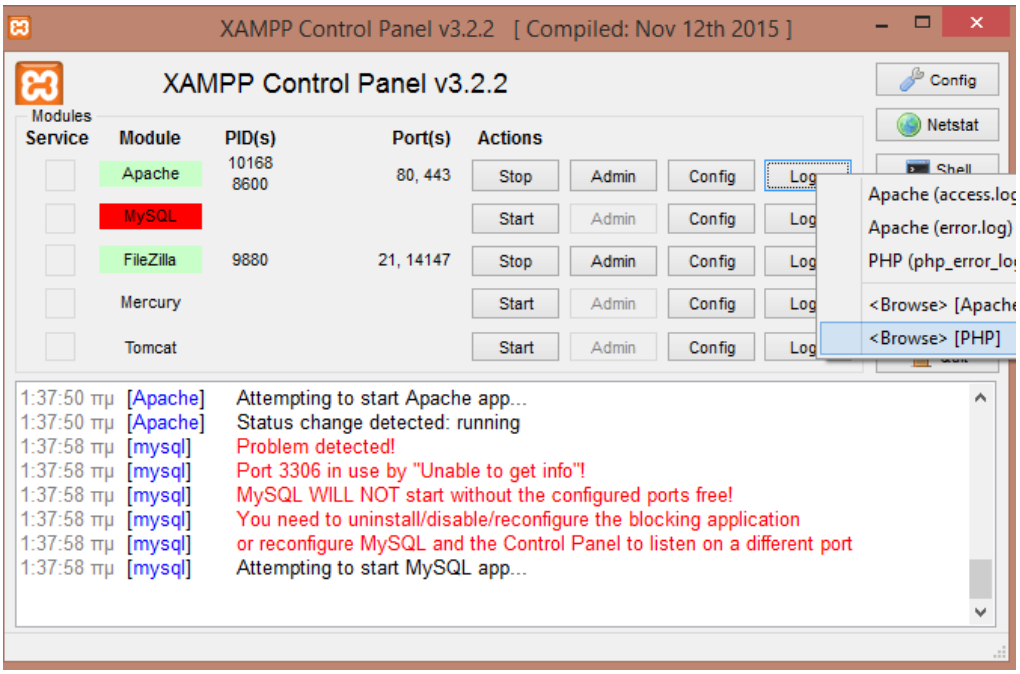


Εικόνα 2.3: Στιγμιότυπο του IDE Phpstorm

2.2.2. XAMPP

Το **XAMPP** [3] είναι ένα δωρεάν ανοιχτού κώδικα διαπλατφορμικό λογισμικό το οποίο προσφέρει όλα τα απαραίτητα εργαλεία για την ανάπτυξη και δημιουργία ενός διαδικτυακού εξυπηρετητή (web server - LAMP) και χρησιμεύει για την εύκολη εγκατάστασή και προσομοίωση της εφαρμογής τοπικά (local), σε πλήθος λειτουργικών συστημάτων. Η ονομασία του κατά ανεπίσημο τρόπο είναι ένα ακρωνύμιο των όρων 'X' που δηλώνει cross-platform, **A**pache HTTP server, την βάση δεδομένων **M**ariaDB, **P**HP και **P**ERL. Παρέχει ευκολία στην ενεργοποίηση των τεχνολογιών που χρειάζονται

για να τρέξει τοπικά ένας εξυπηρετητής και μας δίνει συνεχή εικόνα και έλεγχο για την κατάσταση των λειτουργιών που προαναφέρθηκαν. Είναι με λίγα λόγια ένα μέσο ενεργοποίησης και ελέγχου της βάσεως δεδομένων και του apache σε τοπικό επίπεδο. Στην εικόνα 2.4 δίνεται ένα στιγμιότυπο της λειτουργίας και του λόγου χρήσης του λογισμικού.



Εικόνα 2.4: Στιγμιότυπο λογισμικού XAMPP.

2.2.3. Apache HTTP server

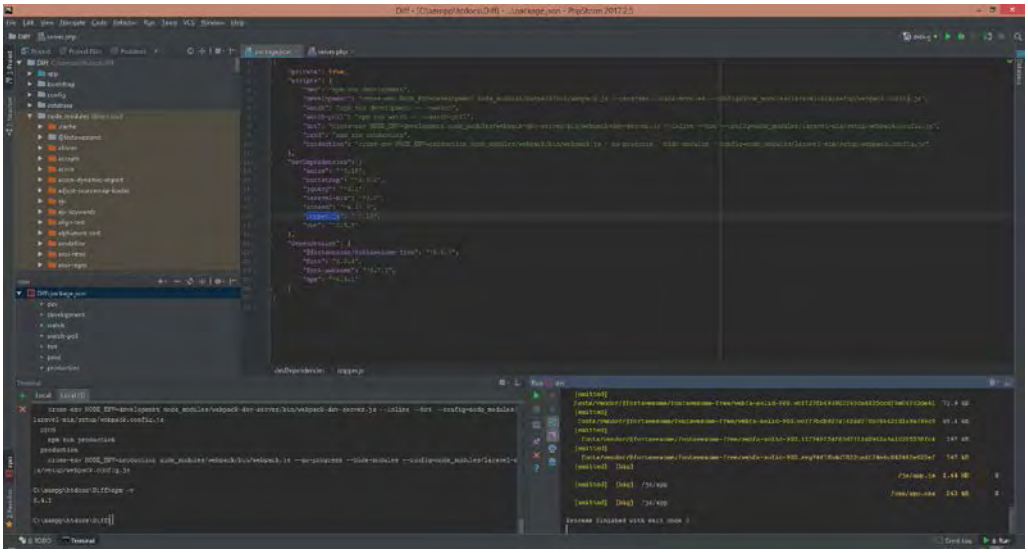
Ο **Apache HTTP server** [4] ή εν συντομία **Apache**, είναι ένα δωρεάν και ανοιχτού κώδικα διαπλατφορμικό λογισμικό διακομιστή ιστού (*web-server*). Έχει δημιουργηθεί και συντηρείται από μία ανοιχτή ομάδα προγραμματιστών η οποία λειτουργεί υπό την επίβλεψη της μη κερδοσκοπικής εταιρίας *Apache Software Foundation*. Στις τελευταίες μετρήσεις (Δεκέμβριος 2018) το 32.46% των παγκόσμιων εξυπηρετητών λειτουργούν υπό τη χρήση του Apache και αν και έχει σταθερή πτωτική πορεία χρήσης λόγω μεγάλου ανταγωνισμού, παραμένει σταθερά εδώ και δεκαετίες ως ο πρώτος και πιο έμπιστος διακομιστής παγκόσμιου ιστού [5].

Παρέχει δυνατότητες όπως εύκολη διαχείριση μεγάλου όγκου πληροφοριών και χιλιάδων ταυτόχρονων συνδέσεων, καταγραφή δεδομένων χρηστών και περιόδων λειτουργίας τους, λειτουργίες με βάση τα διαδικτυακά πρωτοκολλά (**IP** – *Internet Protocol*) όπως ο γεωγραφικός εντοπισμός και τέλος, διάφορες μορφές συνδέσεων, μεταφοράς δεδομένων και πρωτόκολλων επικοινωνίας, όπως *ftp, SSL, HTTP, IPv6 Proxy* και *Websocket*.

Στην εργασία, επιλέχθηκε η χρήση του Apache ως ο διακομιστής πάνω στον οποίο θα βασιστούν οι πλατφόρμες λόγω της ευκολίας στην εγκατάστασή του σε τοπικό επίπεδο αλλά και της συμβατότητάς του με τα υπόλοιπα εργαλεία, όπως το βασικό framework της PHP που χρησιμοποιείται στην εργασία , την Laravel.

2.2.4. Npm

Ο **npm** (*Node Package Manager*) [6] είναι ένας διαχειριστής πακέτων, εργαλείων και αρχείων για την γλώσσα προγραμματισμού javascript και ενός από τα περιβάλλοντα εκτέλεσής της, την **Node.js**. Απαρτίζεται από ένα πρόγραμμα γραμμής εντολών και παρέχει προς χρήση εκατοντάδες χιλιάδες καταγεγραμμένα πακέτα / δομοστοιχεία (**modules**) και βιβλιοθήκες τα οποία εκτελούν λειτουργίες γραμμένες σε **javascript**. Αποτελεί απαραίτητο στοιχείο για την περάτωση μεγάλων project, καθώς πέρα από την χρήση του από την Node.js ομαδοποιεί και διαχειρίζεται ως σύνολο όλα τα διαφορετικά αρχεία μορφής javascript που χρειάζονται για την περάτωση των project. Ταυτόχρονα, καθώς τα συγκεντρώνει σε αρχεία μορφής .json ενημερώνει και προσδιορίζει τις εκδόσεις που χρειάζονται ή όχι αναβάθμιση για την ομαλή λειτουργία των javascript πακέτων που χρησιμοποιούνται, καθιστώντας το απαραίτητο στοιχείο και κλειδί για την ομαλή λειτουργία και διαχείριση μεγάλων και πολυδαίδαλων εφαρμογών.



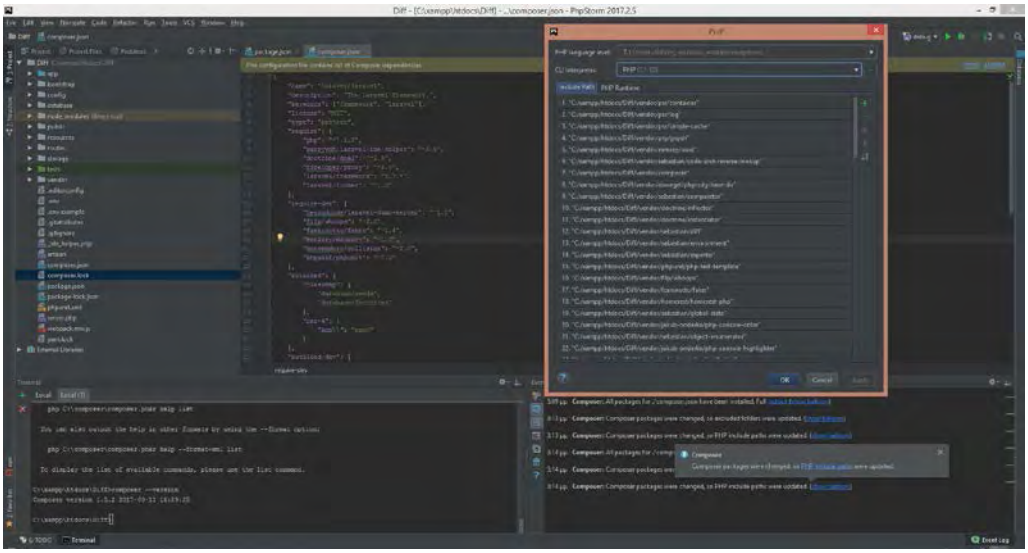
Εικόνα 2.5: Στιγμιότυπο διάρθρωσης και λειτουργιών του npm.

Στην εικόνα 2.5 βλέπουμε ένα στιγμιότυπο των λειτουργιών του npm το οποίο τρέχει στην πλατφόρμα των χρηστών της εφαρμογής, με εμφανή τα στοιχεία που προαναφέρθηκαν και τον τρόπο που τα ομαδοποιεί, αλλά και τις χιλιάδες εγκεκριμένες βιβλιοθήκες σε javascript που δίνονται για την λειτουργία του συστήματος σε τοπικό εξυπηρετητή. Μετά το πέρας και της υλοποίησης των βασικών λειτουργιών της εφαρμογής, έχουμε την δυνατότητα να αφαιρέσουμε

ό,τι πακέτο και αρχείο έχει μείνει αχρησιμοποίητο και βαραίνει το σύστημα αλλά και την εφαρμογή μας.

2.2.5. Composer

Αντίστοιχα με τον npm, ο **Composer** [7] είναι και αυτός ένα εργαλείο για την εύκολη διαχείριση και εγκατάσταση των αρχείων, βιβλιοθηκών και πλαισίων λογισμικών, με κύρια διαφορά ότι διαμορφώνει και διαχειρίζεται τα αρχεία της γλώσσας PHP. Είναι το βασικό λογισμικό μέσω του οποίου εγκαθίστανται frameworks όπως η **Laravel** και άλλα πρόσθετα υποστηρικτικά εργαλεία της, βοηθώντας τον προγραμματιστή να έχει καθολική γνώση των αρχείων και εκδόσεων που χρησιμοποιούνται. Αναγνωρίζει τις επιμέρους βιβλιοθήκες και εφαρμογές που είναι αναγκαίες από ένα framework ή μία ήδη υπάρχουσα βιβλιοθήκη και ενημερώνει αυτόματα τον χρήστη για πιθανά προβλήματα μη χρήσης τους. Τρέχει σε γλώσσα προγραμματισμού PHP, έκδοσης 5.3.2 ή μεταγενέστερης.



Εικόνα 2.6: Στιγμιότυπο λειτουργικότητας του Composer.

Στην εικόνα 2.6 φαίνονται ορισμένες από τις λειτουργίες του κατά την διάρκεια ανάπτυξης της εργασίας, καθώς μετά από μία αναβάθμιση ενός στοιχείου που χρησιμοποιείται μέσω του composer, μας ενημερώνει για τις αυτοματοποιημένες αλλαγές που έγιναν στα επιμέρους αρχεία της εφαρμογής ώστε να μπορούν να συμπεριλαμβάνονται στο προτζεκτ όλες οι επιμέρους απαραίτητες βιβλιοθήκες.

2.2.6. PHP

Η **PHP** [8] δημιουργήθηκε από τον *Rasmus Ledford το 1994* και θεωρείται μία υψηλού επιπέδου γλώσσα προγραμματισμού βασισμένη στην γλώσσα προγραμματισμού C και σχεδιασμένη κυρίως για την δημιουργία δυναμικών

ιστοσελίδων και εφαρμογών. Αρχικά ήταν ένα ακρωνύμιο της φράσης "*Personal Home Page*" αλλά ύστερα από αλλαγές και αναβαθμίσεις των εκδόσεών της, μετονομάστηκε σε "*PHP: Hypertext Preprocessor*". Μπορεί να ενσωματωθεί σε κώδικα γλώσσας σήμανσης **HTML**, αλλά ταυτόχρονα μπορεί να χρησιμοποιηθεί για την δημιουργία διαφόρων συστημάτων διαχείρισης περιεχομένου ιστού (**cms** – *content management system*) και πλαισίων διαδικτυακών λογισμικών (*web frameworks*).

Στις 1 εκατομμύριο πιο πολυσύχναστες ιστοσελίδες παγκοσμίως, η PHP χρησιμοποιείται στο 71% των διαδικτυακών εφαρμογών και ως αυτού καθίσταται η πιο διαδεδομένη και ευρέως γνωστή γλώσσα προγραμματισμού στο διαδίκτυο [9]. Πάνω σε αυτήν έχουν βασιστεί και αναπτυχθεί διαδικτυακά εργαλεία **CMS** και **CRM** (*Customer Relationship Management*) συστήματα, όπως τα *Wordpress, Joomla* και *Drupal,* καθώς και frameworks τα οποία βοηθούν στην ανάπτυξη εφαρμογών, όπως τα *Symfony, Laravel* και *Zend*.

Με την πάροδο του καιρού από την ημέρα κυκλοφορίας της, οι προγραμματιστές διαδικτυακών εφαρμογών άρχισαν να προβληματίζονται για τις επιδόσεις και την χρησιμότητα της γλώσσας. Νέες τεχνολογίες, αναβαθμίσεις και frameworks άλλων γλωσσών όπως η JAVA, η Python, η Ruby on Rails και η C++ αντιμετωπίζουν με μεγαλύτερη ευκολία προβλήματα που διέθετε εξ' ορισμού ο πυρήνας της γλώσσας της PHP. Η μετατροπή της από απλή διαχείριση και ενσωμάτωσή κώδικα HTML γραμμένη σε C, σε πολύπλευρη και πολύ-λειτουργική γλώσσα για τις σύγχρονες ανάγκες του διαδικτύου, άφησε πολλά αδόμητα κενά και ασάφειες από τον πυρήνα μέχρι και τις απλούστερες λειτουργίες της. Μερικές από αυτές είναι, ο χαώδης χώρος διευθύνσεων, αρχείων και μεταβλητών της και οι υπερβολικά χαμηλές επιδόσεις επεξεργασίας δεδομένων και λογικών πράξεων (βρόγχους, δυναμικούς πίνακες και βασικές αριθμητικές πράξεις) σε σχέση με άλλες γλώσσες υψηλού επιπέδου. Με την νεότερη έκδοση της γλώσσας PHP 7.0 έχει γίνει ελαφρώς πιο ανταγωνιστική ανάμεσα στις υπόλοιπες γλώσσες προγραμματισμού διαδικτυακών εφαρμογών και ιστοσελίδων. Στην εικόνα 2.7 παρουσιάζονται μετρήσεις σε υπολογιστικές επιδόσεις γλωσσών που χρησιμοποιούνται και για διαδικτυακές εφαρμογές [10].

Language	CPU time			Slower than		Language version	Source code
	User	System	Total	C++	previous		
C++ *(optimized with -O2)*	0.952	0.172	1.124	–	–	g++ 5.3.1	link
Java 8 *(non-std lib)*	1.332	0.096	1.428	27%	27%	1.8.0_72	link
Python 2.7 + PyPy	1.560	0.160	1.720	53%	20%	PyPy 4.0.1	link
Javascript (nodejs)	1.524	0.516	2.040	81%	19%	4.2.6	link
C++ *(not optimized)*	2.988	0.168	3.156	181%	55%	g++ 5.3.1	link
PHP 7.0	6.524	0.184	6.708	497%	113%	7.0.2	link
Java 8	14.616	0.908	15.524	1281%	131%	1.8.0_72	link
Python 3.5	18.656	0.348	19.004	1591%	22%	3.5.1	link
Python 2.7	20.776	0.336	21.112	1778%	11%	2.7.11	link
Perl	25.044	0.236	25.280	2149%	20%	5.22.1	link
PHP 5.6	66.444	2.340	68.784	6020%	172%	5.6.17	link

Εικόνα 2.7: Συγκριτικά αποτελέσματα επιδόσεων διαδικτυακών γλωσσών προγραμματισμού.

Βασική λειτουργία της γλώσσας όπως προαναφέρθηκε είναι η χρήση της για να δημιουργεί δυναμικές εφαρμογές και ιστοσελίδες. Αναπτύσσετε και τρέχει κυρίως στο περιβάλλον του διακομιστή (*server-side scripting*). Επομένως, για να εξαχθούν τα αποτελέσματα του κώδικα γραμμένα σε PHP χρειάζεται ένας διακομιστής, ένας αναλυτής PHP (*parser*) και ένας φυλλομετρητής. Ταυτόχρονα, βασικά χαρακτηριστικά που την κάνουν αρεστή για διαδικτυακές εφαρμογές και ιστοσελίδες είναι ότι μπορεί να αναζητήσει, να δημιουργήσει (**Create**), να διαβάσει (**Read**), να ενημερώσει (**Update**) και να διαγράψει (**Delete**) δεδομένα (**CRUD** *data*). Κατόπιν, αφού πραγματοποιεί τις απαραίτητες διεργασίες δεδομένων που έχουν ανατεθεί, μπορεί να χρησιμοποιηθεί για να εξάγει τις απαραίτητες πληροφορίες σε μορφή HTML, όπου η τελευταία μπορεί να διαβαστεί και να μεταφραστεί από τους φυλλομετρητές.

Στην εργασία που έχω υλοποιήσει, έχει χρησιμοποιηθεί η PHP ως η βασική γλώσσα προγραμματισμού για το back-end κομμάτι της ανάπτυξης της εφαρμογής. Εξάλλου, τα δύο κεντρικά Laravel frameworks που χρησιμοποιούνται το ένα για την δημιουργία της πλατφόρμας του χρήστη και το άλλο για την δημιουργία της πλατφόρμας των εταιριών, είναι frameworks της PHP. Με την χρήση της PHP, υλοποιούνται όλες οι συνδέσεις με τις βάσεις δεδομένων, όλες οι CRUD διεργασίες, καθώς και όλες οι λογικές λειτουργίες για την ομαλή και ασφαλή μετάδοση και ροή των δεδομένων από τις βάσεις δεδομένων στον χρήστη και αντίστροφα.

2.2.7. MariaDB

Η **MariaDB** [11] προέρχεται από μία διακλάδωση (*fork*) του συστήματος διαχείρισης σχεσιακών βάσεων δεδομένων **MySQL**. Είναι μία βάση δεδομένων που έχει αναπτυχθεί και συντηρείται από βασικούς ιδρυτές της γλώσσας MySQL, ύστερα από προβληματισμούς και ανησυχίες που υπήρξαν όταν η τελευταία αγοράστηκε από την *Oracle*. Εγγυάται να παραμείνει λογισμικό ανοικτού και ελεύθερου κώδικα και είναι μία τις πιο ευρέως διαδεδομένη βάση δεδομένων παγκοσμίως, με εφαρμογή και χρήση της από εταιρίες και πλατφόρμες όπως η *Google*, η *Wikipedia* και η *Wordpress*.

Επιλέχθηκε η χρήση αυτής της βάσης καθώς είναι συμβατή με την **Laravel**, μπορεί να διαχειριστεί μεγάλο όγκο δεδομένων και έχει νέο πεδίο μεταβλητών μορφής **JSON**.

2.2.8. Node.js

Η **Node.js** [12] είναι ένα ελεύθερου κώδικα διαπλατφορμικό περιβάλλον εκτέλεσης της γλώσσας javascript, όπου τρέχει και εκτελείται έξω από το περιβάλλον λειτουργίας ενός φυλλομετρητή. Δημιουργήθηκε το 2009 και έχει ως κύριο σκοπό να εκτελεί προγράμματα και λειτουργίες σε γλώσσα **javascript** σε έναν εξυπηρετητή (*server-side*) και όχι στο σύστημα του πελάτη (*client-side*).

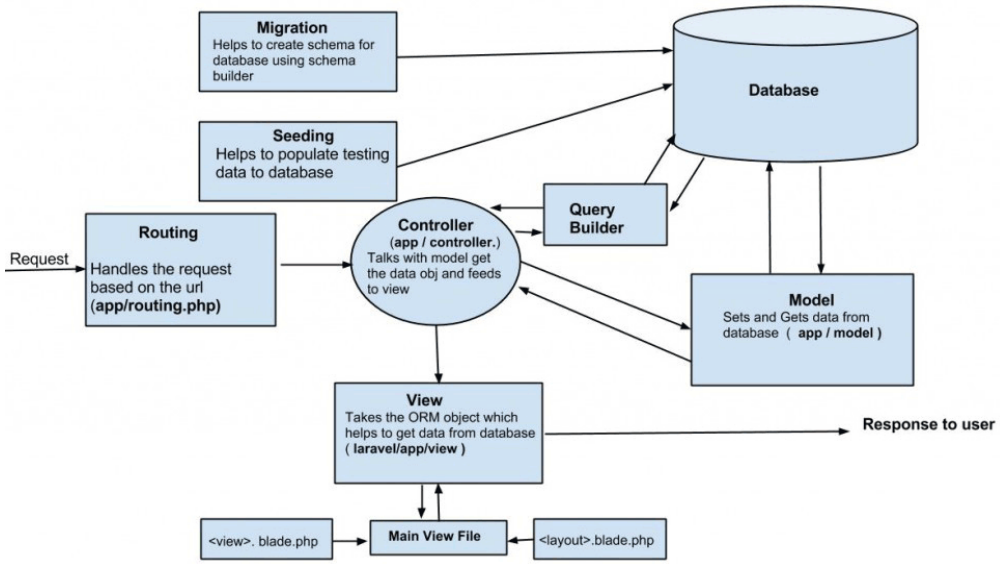
Αν και έχει εγκατασταθεί στο παρόν σύστημά των δύο πλατφόρμων της εργασίας μας λόγω του Npm, για την ώρα δεν εφαρμόζεται κάποια δυνατότητά της. Η χρήση της πρόκειται να πραγματοποιηθεί κατά την περαιτέρω ανάπτυξη της εργασίας σε μελλοντικό χρόνο, καθώς το θεωρώ κατάλληλο περιβάλλον ανάπτυξης ενός **API** επικοινωνίας και επεξεργασίας μεγάλου όγκου δεδομένων.

2.2.9. Laravel

Η **Laravel** [13] είναι ένα δωρεάν, ανοικτού κώδικα framework της PHP. Δημιουργήθηκε το 2011 από τον Taylor Otwell και έχει ως σκοπό την ανάπτυξη διαδικτυακών εφαρμογών, ακολουθώντας παρόμοια αρχιτεκτονική με την αρχιτεκτονική **Model-View-Controller** (MVC) και είναι βασισμένη στο εξίσου framework της PHP Symfony.

Μερικά από τα βασικά τεχνικά γνωρίσματα της Laravel που την έχουν κάνει ευρέως διαδεδομένο framework, είναι οι ποικίλοι τρόποι πρόσβασης στις σχεσιακές βάσεις δεδομένων και τα επιμέρους βοηθήματα και εργαλεία που βοηθούν στην ανάπτυξη και συντήρηση μίας διαδικτυακής εφαρμογής. Διαθέτει δικό της **CLI** (*Command Line / Language Interface*), τον *Artisan*, με το οποίο ο προγραμματιστής μπορεί να δώσει εντολές και να αναπτύξει πολλά στοιχεία και εφαρμογές του προγράμματός του. Με βάση το framework αυτό, έχουν αναπτυχθεί και βασιστεί οι τεχνικές για την υλοποίηση των δύο

πλατφορμών της εργασίας. Στην εικόνα 2.8 βλέπουμε την βασική αρχιτεκτονική δομή του Laravel framework, όπου περαιτέρω ανάλυση των δομικών στοιχείων πραγματοποιείται στα επόμενα κεφάλαια.



Εικόνα 2.8: Αρχιτεκτονική Laravel framework.

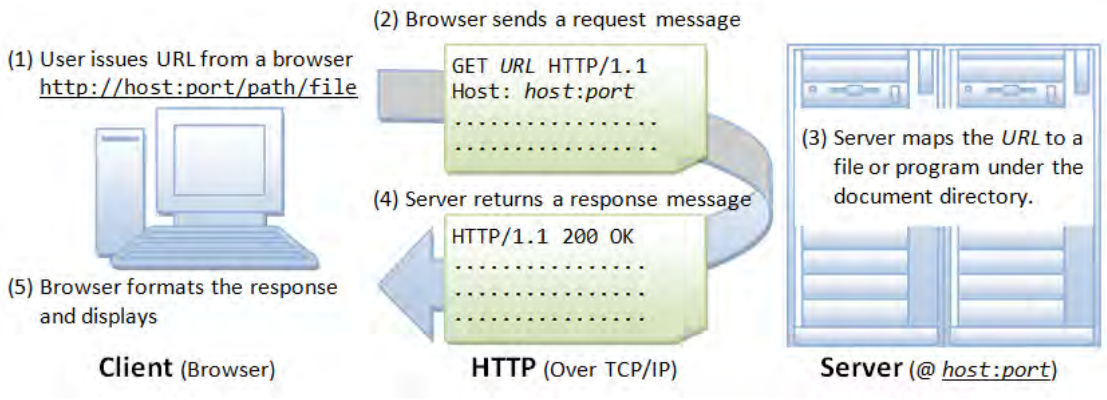
2.2.10. HTTP

Το **HTTP** (*HyperText Transfer Protocol*) ή Πρωτόκολλο Μεταφοράς Υπερκειμένου, είναι το βασικό και πιο διαδεδομένο πρωτόκολλο επικοινωνίας για την μεταφορά δεδομένων ανάμεσα σε έναν διακομιστή (*server*) και έναν πελάτη (*client*). Η ανάπτυξή του πραγματοποιήθηκε το 1989 στο πειραματικό κέντρο ερευνών CERN από την καθοδήγηση του *Tim Berners-Lee* και η νεότερή του έκδοση έχει την ονομασία HTTP/2.

Το πρωτόκολλο είναι κατασκευασμένο σύμφωνα με το μοντέλο επικοινωνίας Εξυπηρετητή-Πελάτη με την ονομασία **Request-response** ή **Request-reply protocol**. Παράλληλα, η διαδικτυακή εφαρμογή που έχει αναπτυχθεί, χρησιμοποιεί ως επί το πλείστον το συγκεκριμένο πρωτόκολλο για να επικοινωνούν και να μεταφέρονται τα δεδομένα ανάμεσα στις πλατφόρμες και τους χρήστες τους.

Βασικές επιμέρους χρήσεις και λειτουργίες του **HTTP** που χρησιμοποιούνται και στην εργασία είναι, τα πολλαπλά συστήματα επαλήθευσης ταυτότητας (*authentication schemes*) που παρέχει, οι πολλαπλοί μέθοδοι με τους οποίους ορίζεται κάποιο Αίτημα (Request) όπως οι **GET, HEAD, POST, PUT, DELETE** καθώς και η δυνατότητα δημιουργίας κρυπτογραφημένων

συνδέσεων (**HTTPS**) για μεγαλύτερη ασφάλεια μετάδοσης πληροφοριών. Στην εικόνα 2.9 βλέπουμε τα βήματα που πραγματοποιούνται για ένα Request μορφής GET με την λειτουργία του πρωτοκόλλου HTTP.

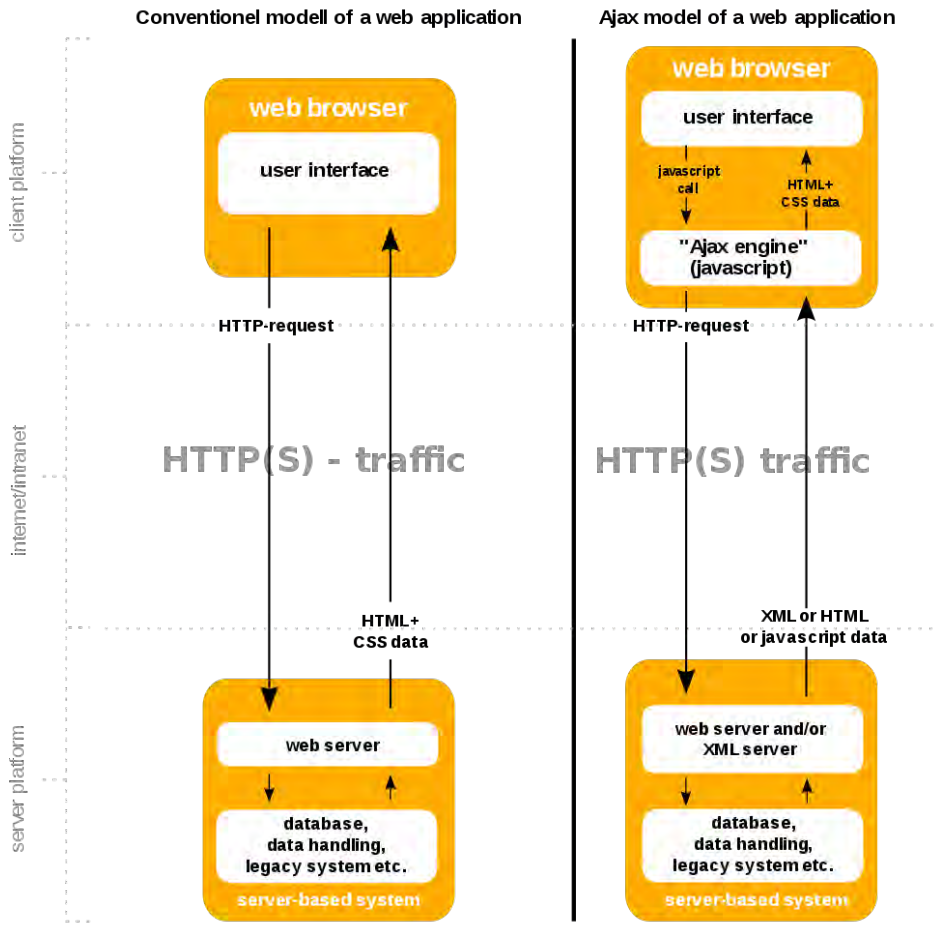


Εικόνα 2.9: Βήματα εκτέλεσης GET σε πρωτόκολλο HTTP.

2.2.11. Ajax

Το **AJAX** (***A****synchronous* ***Ja****vaScript and* ***X****ML*) συνδυάζει ένα σύνολο διαδικτυακών τεχνολογιών που εφαρμόζονται στο σύστημα του πελάτη (*client side*), με σκοπό να δημιουργήσει ασύγχρονες διαδικτυακές εφαρμογές (*Asynchronous Web Applications*). Με την χρήση του AJAX, οι διαδικτυακές εφαρμογές μπορούν να στέλνουν και να λαμβάνουν δεδομένα ασύγχρονα (στο παρασκήνιο) ανάμεσα στον χρήστη (*client*) και τον εξυπηρετητή (*server*), χωρίς να επεμβαίνουν στη γενική εικόνα που προβάλλεται στην ιστοσελίδα του χρήστη. Με την χρήση αυτής της τεχνολογίας, μπορούν να μειωθούν έως και να εκμηδενιστούν οι ανανεώσεις για την προβολή και αποστολή δεδομένων που θα χρειάζονταν στο μοντέλο Request-reply που χρησιμοποιείται στην εργασία.

Η λειτουργία του Ajax επιτυγχάνεται όταν συνδυάζονται οι γλώσσες προγραμματισμού **XML** (*Extensible Markup Language*) και **Javascript.** Μέσω της javascript συλλέγονται τα δεδομένα ενός αιτήματος (request) από την HTML και CSS στον φυλλομετρητή του χρήστη και στην συνέχεια μέσω του αντικειμένου XMLHttpRequest που δημιουργεί η javascript μεταφέρονται σε μορφή XML από τον χρήστη στον εξυπηρετητή. Στη συνέχεια τα δεδομένα της απάντησης (response) του εξυπηρετητή συλλέγονται με την χρήση της javascript η οποία τα ενσωματώνει στην τελική εικόνα του φυλλομετρητή μετατρέποντάς τα σε μορφή CSS και HTML. Στην εικόνα 2.10 βλέπουμε τα βήματα εκτέλεσης που πραγματοποιούνται κατά ένα request-reply με την χρήση Ajax και τις βασικές διαφορές με το κλασσικό μοντέλο επικοινωνίας.



Εικόνα 2.10: Βήματα εκτέλεσης και διαφορές με χρήση κλασσικού και Ajax μοντέλου σε διαδικτυακή εφαρμογή.

2.2.12. HTML

Η **HTML** (*HyperText Markup Language*) ή Γλώσσα Σήμανσης Υπερκειμένου είναι μία γλώσσα σήμανσης που χρησιμοποιείται κατά κόρον στην δημιουργία των βασικών δομικών στοιχείων των ιστοσελίδων. Δημιουργήθηκε το 1980 από τον φυσικό *Τιμ Μπέρνερς Λι* κατά την διάρκεια εργασίας του στο κέντρο πειραμάτων CERN, με σκοπό τον ταχύτερο διαμοιρασμό εγγράφων ανάμεσα στους ερευνητές. Οι βασικές λειτουργίες της ως γλώσσα σήμανσης υπερκειμένου αναπτύχθηκαν από τον ίδιο το 1989 όπου και ανέπτυξε της βασικές προδιαγραφές της, καθώς και τα πρώτα λογισμικά φυλλομετρητή και εξυπηρετητή για την μετάδοσή και απεικονιστική εμφάνιση των εγγράφων μορφής HTML.

Ύστερα από αρκετές ενημερώσεις και εκδόσεις *(HTML5 / XHTML5)*, αναπτύχθηκαν βασικά στοιχεία σήμανσης και ιδιότητες, ικανά να διαθέτουν μία πληθώρα λειτουργιών υπερκειμένου. Η δομή των αρχείων HTML

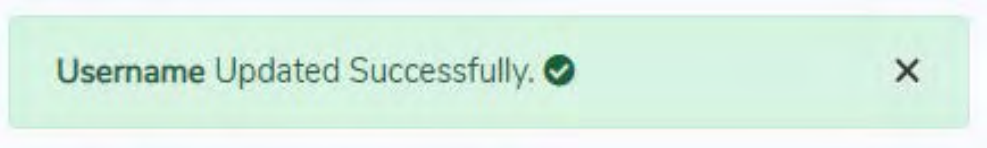
διατάσσεται με την χρήση στοιχείων (**elements**). Τα στοιχεία αυτά στην πιο γενική τους μορφή έχουν τρία γνωρίσματα. Αυτά τα τρία γνωρίσματα είναι, ένα ζεύγος από ετικέτες (**tag**) που καθορίζουν τα σημεία εκκίνησης και τερματισμού ενός στοιχείου, οι ανάλογες ιδιότητες που αφορούν το στοιχείο οι οποίες αναγράφονται μέσα στο σημείο εκκίνησης (start tag) και τέλος το περιεχόμενο μεταξύ των δύο ετικετών, το οποίο μπορεί να αποτελείται από ένα απλό κείμενο αλλά μπορεί να εμπεριέχει και άλλα στοιχεία εμφωλευμένα στο εσωτερικό του.

Στην εικόνα 2.11 βλέπουμε ένα κομμάτι κώδικα της εργασίας σε HTML, με εμφανή την δομή που παρουσιάσαμε προηγουμένως.

```
<div id="alert_success" class="alert alert-dismissable alert-success">
   <button type="button" class="close">
      <span aria-hidden="true">&times;</span>
   </button>
   <strong></strong>
   <span id="alert_msg"> Updated Successfully. </span><i class="fas fa-check-circle"></i>
</div>
```

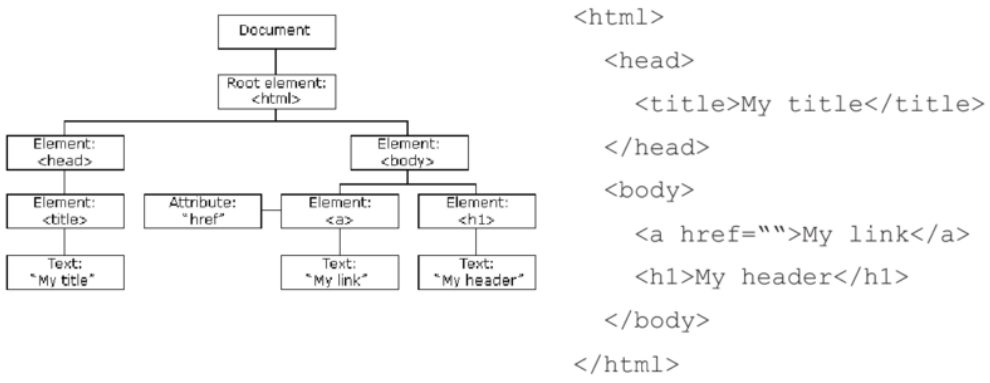
Εικόνα 2.11: Κομμάτι κώδικα html από το αρχείο index.blade.php που απαρτίζει το στοιχείο 'alert-success'.

Ο παραπάνω κώδικας μαζί με την χρήση των απαραίτητων ιδιοτήτων και λειτουργιών που έχουν γραφτεί σε **javascript** και την διαμόρφωση του ύφους και των γραφικών στοιχείων που συμπεριλαμβάνονται στα αρχεία της **CSS**, διαβάζεται και επεξεργάζεται από τον φυλλομετρητή και στην συνέχεια παρουσιάζεται στην οθόνη του πελάτη με την μορφή της εικόνας 2.12.



Εικόνα 2.12: Η προβολή του στοιχείου του κώδικα της εικόνας 2.11 στον φυλλομετρητή.

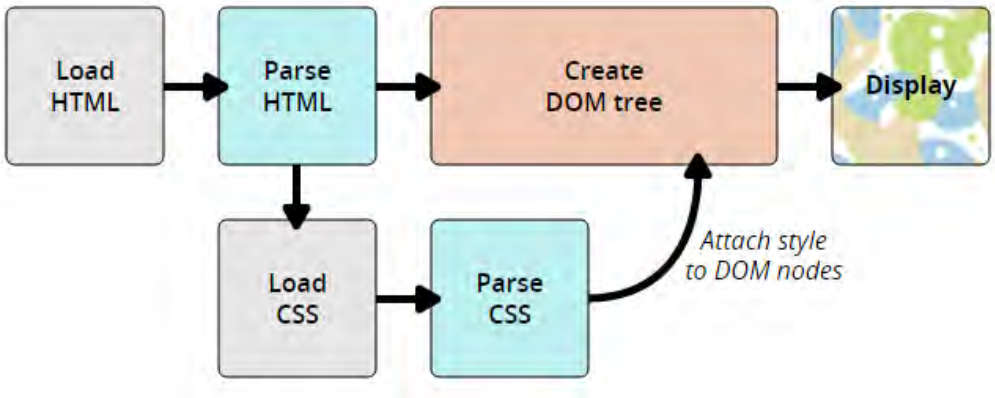
Ο τρόπος με τον οποίο έχουν δομηθεί τα αρχεία μορφής HTML και XML, δίνουν την δυνατότητα εύκολης δενδρικής αναπαράστασης των στοιχείων τους. Πάνω σε αυτό το δενδρικό μοντέλο, βασίστηκε η δημιουργία του διαπλατφορμικού λογισμικού δενδρικής δομής **HTML DOM** (*Document Object Model*), όπου κατασκευάζεται κυρίως στον φυλλομετρητή του πελάτη. Το μοντέλο αυτό αναπαριστά το αρχείο σε λογική δεντρική μορφή, όπου κάθε κόμβος του δέντρου αποτελείται από αντικείμενα. Με την βοήθεια αυτής της δομής, η γλώσσα προγραμματισμού javascript αποκτά πρόσβαση σε όλα τα στοιχεία του αρχείου HTML καθιστώντας την ικανή να πραγματοποιήσει κάθε είδους αλλαγή στο υπάρχον ή προσθήκη νέων κόμβων στον DOM και συνεπώς στοιχείων στην HTML. Στην εικόνα 2.13 παρουσιάζεται ένα παράδειγμα στην δημιουργία του DOM από κώδικα HTML.



Εικόνα 2.13: Δενδρική αναπαράσταση του κώδικα HTML με το μοντέλο DOM.

2.2.13. CSS

Η **CSS** (*Cascading Style Sheets*) ή Διαδοχικά Φύλλα Ύφους, είναι μία γλώσσα που ανήκει στην κατηγορία γλωσσών φύλλων ύφους και χρησιμοποιείται για την παρουσίαση και τον έλεγχο της εμφάνισης ενός αρχείου που έχει γραφτεί σε γλώσσα σήμανσης όπως η HTML. Διαθέτει πολλά χαρακτηριστικά και ιδιότητες για την διαμόρφωση και την γραφική απεικόνιση ενός αρχείου HTML και μαζί με την javascript οι τρείς αυτές γλώσσες θεωρούνται οι ακρογωνιαίοι λίθοι του παγκόσμιου ιστού. Στην εικόνα 2.14 βλέπουμε τον τρόπο με τον οποίο επεξεργάζεται ένας φυλλομετρητής τα αρχεία HTML και CSS.



Εικόνα 2.14: Στάδια επεξεργασίας αρχείων κώδικα HTML και CSS σε ένα φυλλομετρητή.

Στις πρώτες της εκδόσεις η γλώσσα συνέβαλλε σε βασικές γραφικές διαμορφώσεις των γλωσσών ύφους που ήταν συμβατές, όπως απλές αλλαγές στις ιδιότητες της γραμματοσειράς και στις μεταβολές χρωμάτων κειμένου και φόντου. Πλέον, διαθέτει μία μεγάλη γκάμα αλλαγών και επιλογών (*modules*) για την γραφική παρουσίαση μίας διαδικτυακής ιστοσελίδας όπως ή επιλογή προς διαμόρφωση συγκεκριμένων κόμβων του DOM (*CSS Selectors*) και η χρήση των ‘media queries’ που αποσκοπούν στην προσαρμογή του

περιεχομένου των κόμβων, αναλόγως τις διαστάσεις και τον προσανατολισμό της οθόνης (responsive web design).

Στην παρούσα εργασία, οι δύο διαδικτυακές πλατφόρμες χρησιμοποιούν μία ευρεία γκάμα των δυνατοτήτων της CSS καθώς είναι αναγκαίο να δημιουργηθεί μία άρρηκτη σχέση και διαδραστικότητα ανάμεσα στον πελάτη και την εφαρμογή. Για ευκολότερη και καλύτερη παρουσίαση των γραφικών στοιχείων των σελίδων χρησιμοποιείται ένα framework της CSS, το **Bootstrap**, καθώς διαθέτει ένα σύστημα πλέγματος (*grid system*) που μας βοηθάει στην σχεδίαση και χωροθέτηση των στοιχείων της ιστοσελίδας. Πάνω σε αυτό το πλέγμα έχει βασιστεί και η δομή των κεντρικών στοιχείων των δύο πλατφορμών.

2.2.14. JavaScript

Η **JavaScript** είναι μία διερμηνευμένη υψηλού επιπέδου γλώσσα προγραμματισμού και μία από τις βασικότερες γλώσσες για την υλοποίηση διαδικτυακών εφαρμογών. Οι βασικές ιδιότητες που την διέπουν, είναι η δυναμική της μορφή (*dynamic*), η κληρονομικότητα και επαναχρησιμοποίηση των αρχέτυπων αντικειμένων μέσω αναθέσεων (*prototype-based*, αντικειμενοστραφής), καθώς και η χρήση πολλαπλών προγραμματιστικών παραδειγμάτων (*multi-paradigm*). Έχει δυνατότητες συναρτησιακού (*functional*), χειρισμού γεγονότων (*event-driven*), αντικειμενοστρεφή (*object-oriented*) καθώς και προστακτικού (*imperative*) προγραμματισμού.

Αρχικά, ως μία γλώσσα που χρησιμοποιείτο κατά κόρον στις τεχνολογίες του διαδικτύου, εφαρμοζόταν μόνο στις τεχνολογίες και τα λογισμικά του πελάτη *(client-side*), αλλά πλέον έχουν αναπτυχθεί και περιβάλλοντα εκτέλεσης javascript και σε άλλους τύπους λογισμικών όπως στους εξυπηρετητές και στις βάσεις δεδομένων.

Στην παρούσα εργασία, η Javascript χρησιμοποιείται ως γλώσσα προγραμματισμού του front-side της διαδικτυακής εφαρμογής. Είναι η γλώσσα με την οποία επιτυγχάνεται μεγαλύτερη διαδραστικότητα ανάμεσα στον χρήστη και την εφαρμογή, καθώς και αυτή που διαμορφώνει τις επιμέρους ιδιότητες και λειτουργίες των στοιχείων της HTML, αλλά και τις μεθόδους επικοινωνίας ανάμεσα στον πελάτη και τον εξυπηρετητή. Ένα παράδειγμα των δυνατοτήτων της όταν καλείται από ένα στοιχείο του DOM της HTML παρουσιάζεται μέσω του στιγμιότυπου κώδικα στην εικόνα 2.15, όπου κύριος στόχος του παρακάτω κώδικα Javascript είναι να επιτρέπει την πληκτρολόγηση μόνο αριθμών σε ένα πλαίσιο κειμένου.

```
<input type="text" class="textfield" value="" id="extra7" name="extra7"
      onkeypress="return isNumber(event)" />

<script>
   function isNumber(evt) {
      evt = (evt) ? evt : window.event;
      var charCode = (evt.which) ? evt.which : evt.keyCode;
      if (charCode > 31 && (charCode < 48 || charCode > 57)) {
         return false;
      }
      return true;
   }
</script>
```

Εικόνα 2.15: Κώδικας σε γλώσσα HTML και Javascript.

Μία επιμέρους χρήση της Javascript πραγματοποιείται κατά την διάρκεια της ανάπτυξης της εφαρμογής καθώς μπορεί να ενημερώνει τον προγραμματιστή για της λειτουργίες των στοιχείων της ιστοσελίδας, μέσω προγράμματος γραμμής εντολών (**CLI** – *Command Line Interface*) που διαθέτουν οι φυλλομετρητές.

2.2.15. Jquery

Η **Jquery** [14] είναι μία βιβλιοθήκη της JavaScript σχεδιασμένη με σκοπό να απλοποιεί τον τρόπο με τον οποίο η JavaScript χειραγωγεί και διαβάζει την δεντρική μορφή του HTML DOM, τον χειρισμό των εκάστοτε συμβάντων *(event handling)*, τα εφέ κίνησης που χρησιμοποιούνται μέσω της CSS, καθώς και την σύνδεση και χρήση της τεχνολογίας Ajax. Η πρώτη έκδοση της βιβλιοθήκης εμφανίστηκε το 2006 και χρησιμοποιείται κατά κόρον σε όλο το φάσμα των διαδικτυακών εφαρμογών όντας με μεγάλη διαφορά η πιο διαδεδομένη βιβλιοθήκη της JavaScript στην πλευρά του πελάτη (client-side).

Τα τελευταία χρόνια οι φυλλομετρητές έχουν αναπτύξει τα εργαλεία και τα λογισμικά (DOM api) με τα οποία η JavaScript μπορεί να αλληλεπιδρά με τον HTML DOM και πάνω σε αυτά έχουν αναπτυχθεί frameworks της Javascript με καλύτερες επιδόσεις και δυνατότητες από την Jquery, όπως τα *AngularJS* και *React*. Αν και το πλέον φρόνιμο θα ήταν η εξαρχής υλοποίηση και χρήση ενός από τα δύο παραπάνω frameworks, στην εργασία χρησιμοποιείται Jquery και απλή JavaScript (*plain JavaScript*) για ταχύτερη ανάπτυξη του front-end. Παράλληλα, υπάρχουν μελλοντικές βλέψεις χρήσης ενός κεντρικού javascript framework (όπως τα *Angular* και *Vue*) και η εξάλειψη αρκετών λειτουργιών που γίνονται με την βοήθεια της βιβλιοθήκης Jquery, καθώς θα διευκολύνει σε μεγάλο βαθμό την πολυπλοκότητα και συντήρηση που θα αποκτήσει η εφαρμογή, όταν θα έχει πλήρως δομηθεί και αυτοματοποιηθεί.

2.3. Το Μοντέλο MVC (Model-View-Controller)

Το μοντέλο MVC (*Model View Controller*) είναι ένα γνωστό μοντέλο αρχιτεκτονικής λογισμικού, που χρησιμοποιείται κυρίως για την δημιουργία γραφικού περιβάλλοντος χρήστη (**GUI** – *Graphical User Interface*), αλλά και σε διαδικτυακά συστήματα και εφαρμογές. Το framework **Laravel** χρησιμοποιεί πολλά χαρακτηριστικά από το μοντέλο αυτό για την ανάπτυξη διαδικτυακών εφαρμογών και πάνω σε αυτό έχουν βασιστεί οι δύο πλατφόρμες (πελατών και εταιριών).

Με την χρήση του μοντέλου MVC η εφαρμογή χωρίζεται σε τρία βασικά δομικά στοιχεία, το **Model**, το **View**, και τον **Controller**, όπου οι λειτουργίες τους διαχωρίζονται στην Laravel συνοπτικά ως εξής:

- Το **Model** διαχειρίζεται θεμελιώδη συμπεριφορές και δεδομένα της εφαρμογής. Είναι η λειτουργική μονάδα που διατηρεί, ελέγχει και συσχετίζει τα δεδομένα της εφαρμογής με τις βάσεις δεδομένων, δημιουργώντας τα αντίστοιχα αντικείμενα, μεταβλητές και διεργασίες μέσω της laravel. Δεν εμπλέκονται οι λογικές λειτουργίες της με τα άλλα δύο στοιχεία, και αν αναπτυχθεί κατάλληλα, η εφαρμογή μπορεί να εκτελέσει τις λειτουργίες της (*business logic*), στον εξυπηρετητή και την βάση δεδομένων, χωρίς τον Controller και το View ή άλλες υπομονάδες. Στο Model περιλαμβάνεται ένα σύστημα αντικειμενοστραφούς ταξινόμησης (**ORM** - *object-oriented mapping*) το **Eloquent ORM** το οποίο διαθέτει ενεργό πρότυπο καταγραφής (*active record pattern*) αντικειμένων και διευκολύνει την επικοινωνία της εφαρμογής με την βάση δεδομένων. Τέλος, έχει την δυνατότητα να διαχειρίζεται ένα πλήθος λειτουργιών συμβάντων (*events*) καθώς και την ομαδοποίησή τους, δημιουργώντας παρατηρητές κίνησης δεδομένων (*Data Observers*).

- Το **View** είναι η μονάδα που διαμορφώνει το front-end περιβάλλον της εφαρμογής. Παρέχει λειτουργίες και τρόπους εξαγωγής των δεδομένων από το Model και καθορίζει τον τρόπο εμφάνισης των δεδομένων της εφαρμογής στον φυλλομετρητή, καθώς και τους τρόπους αλληλεπίδρασης του χρήστη με αυτήν. Έχει την δυνατότητα να ερμηνεύει και να χρησιμοποιεί τα αντικείμενα του ORM που έχουν δημιουργηθεί στο Model ώστε να μεταβούν τα δεδομένα στο front-end και να δημιουργηθούν οι ανάλογες διεπαφές του χρήστη με την εφαρμογή.

- Ο **Controller** ή **Resource Controller** (*Ελεγκτής Πόρων*) είναι η μονάδα στην οποία γίνεται ο έλεγχος των αιτημάτων (*Request*) του χρήστη της πλατφόρμας και αξιοποιεί όλες τις **CRUD** λειτουργίες της εφαρμογής. Δηλαδή, είναι το μέσο που δέχεται τα δεδομένα ενός αιτήματος του χρήστη, και τα ανακατευθύνει, σύμφωνα με την λογική και τους κανόνες που έχουν

αναπτυχθεί, στο Model ή απευθείας αν δεν χρειάζεται η χρήση βάσεων, στο View.

Παράλληλα με τις τρεις παραπάνω μονάδες, η Laravel όπως προαναφέρθηκε, έχει διαμορφώσει το υπάρχον μοντέλο MVC κατακερματίζοντας τις λειτουργίες των μονάδων με σκοπό να κατασκευάσει ένα ποιο προσιτό μοντέλο για τις ανάγκες των σύγχρονων διαδικτυακών εφαρμογών. Ορισμένες από αυτές τις επιμέρους λειτουργίες είναι η **Route**, όπου χειρίζεται το αίτημα του χρήστη με βάση το URL και κατευθύνει τα δεδομένα στον εκάστοτε Controller, αλλά και το **Middleware** όπου φιλτράρει τα HTTP αιτήματα και απαντήσεις πραγματοποιώντας ελέγχους προτού εισέλθουν στους Controllers ή στα Views.

Με αυτόν τον τρόπο επιτυγχάνονται αρκετά ζητήματα απαραίτητα για την βιωσιμότητα και ομαλή ανάπτυξη της εφαρμογής, όπου τρία από τα βασικότερα είναι:

- Η βελτίωση της δυνατότητας επεκτασιμότητας της εφαρμογής (*scalability*) καθώς τα δομικά της στοιχεία έχουν ελάχιστη αλληλοεξάρτηση μεταξύ τους. Για παράδειγμα, εάν η εφαρμογή αρχίζει και αντιμετωπίζει προβλήματα στην απόδοσή της λόγω προβλημάτων και καθυστερήσεων της βάσης δεδομένων, μπορεί να αναβαθμιστεί το υλικό της βάσης δεδομένων χωρίς να επηρεαστούν άλλα στοιχεία και τμήματα της εφαρμογής.
- Η εύκολη συντήρηση και ταχύτητα αλλαγών στις εσωτερικές λειτουργίες της εφαρμογής, καθώς από την στιγμή που τα δομικά της στοιχεία δεν εξαρτώνται μεταξύ τους, σε περίπτωση που χρειαστεί κάποιο τμήμα της εφαρμογής αλλαγές στην λειτουργικότητά του ή διορθώσεις λόγω κάποιου σφάλματος, τα υπόλοιπα τμήματα δεν επηρεάζονται και συνεχίζουν να λειτουργούν.
- Η επαναχρησιμοποίηση των δομικών στοιχείων. Για παράδειγμα, με το μοντέλο MVC μπορούν να υπάρχουν πολλαπλά Views για ένα συγκεκριμένο Model.

Κεφάλαιο 3ο
Αρχιτεκτονική Ανάπτυξη Εργασίας

3.1. Εισαγωγή στην Ανάπτυξη

Εφόσον έχουμε αναλύσει τις τεχνολογίες, τα εργαλεία και το βασικό μοντέλο σύμφωνα με το οποίο λειτουργεί η Laravel, στο παρόν κεφάλαιο θα αναδείξω τα βασικά μοντέλα περιπτώσεων χρήσης της εφαρμογής, καθώς και τα δομικά στοιχεία και τεχνικές που δημιουργήθηκαν για την ανάπτυξή τους.

Αρχική σκέψη για την υλοποίηση της εργασίας ήταν να υπάρξει μία ενιαία πλατφόρμα βασισμένη σε ένα framework Laravel όπου θα πραγματοποιούσε όλες τις λειτουργίες της εργασίας. Μία λογική που όμως στην πράξη θα επέφερε μεγάλη πολυπλοκότητα στα συστήματά της εφαρμογής, καθώς κάθε τρόπος συντήρησης και μελλοντικής ανάπτυξης και σχεδίασής της θα δημιουργούσε περισσότερα λογικά προβλήματα στα μέρη του κώδικα παρά λύσεις.

Το να υπάρχουν σε μία ενιαία πλατφόρμα διαφορετικοί χρήστες που δεν θα έχουν επικοινωνία μεταξύ τους όπως οι πελάτες και οι επιχειρήσεις, αλλά ταυτόχρονα και διαχειριστές πολλών επιπέδων (moderators, admins, super-admins) για τις λειτουργίες των πελατών, των επιχειρήσεών αλλά και για της πλατφόρμας ως σύνολο, θα επέφερε ένα δαιδαλώδη αρχικό λογικό σχεδιασμό για τις λειτουργίες του καθενός ξεχωριστά και δυσκολία σε επιμέρους λειτουργίες της εφαρμογής. Για να μην υπάρχουν αυτές οι συγκρούσεις και η μεγάλη πολυπλοκότητα στην ανάπτυξη της εφαρμογής που θα μπορούσε να επιφέρει σημαντικά προβλήματα χρήσης, οδηγήθηκα στην δημιουργία δύο πλατφόρμων, πελατών και εταιριών, οι οποίες εκτελούν ξεχωριστά τις ανάλογες λειτουργίες τους.

3.2. Σενάρια εργασίας

3.2.1. Σενάριο και λειτουργίες Πελάτη - Ηλεκτρονικό κατάστημα

Στην εικόνα 3.1 παρουσιάζεται ένα σενάριο περιπτώσεων χρήσης (*use case scenario*) του **widjet** της εφαρμογής με το ηλεκτρονικό κατάστημα που διαθέτει η εταιρία πώλησης ειδών ρουχισμού.

Εικόνα 3.1: Περιπτώσεις χρήσεις πελάτη – ηλεκτρονικού καταστήματος.

Αρχική μας θεώρηση είναι ότι ο πελάτης του ηλεκτρονικού καταστήματος μίας εταιρίας που χρησιμοποιεί το widjet είναι εν δυνάμει πελάτης της εφαρμογής. Σε κάθε σελίδα προϊόντος της εταιρίας ελέγχεται από το widjet εάν η εταιρία έχει δηλώσει το προϊόν στην εφαρμογή και εάν είναι δηλωμένο, τότε ενεργοποιείται η δυνατότητα στον πελάτη να χρησιμοποιήσει τις λειτουργίες του widjet και ανοίγει session ανάμεσα στο API της πλατφόρμας Companies και της σελίδας του προϊόντος της εταιρίας.

Σε περίπτωση που ο πελάτης έχει ξαναχρησιμοποιήσει ή πραγματοποιήσει είσοδο στο widjet του ηλεκτρονικού καταστήματος της εταιρίας στο παρελθόν (χωρίς να έχουν λήξει ή διαγραφεί τα cookies εισόδου στο widjet), γίνεται αυτόματη είσοδος στις λειτουργίες του widjet. Σε κάθε άλλη περίπτωση, ο πελάτης της εταιρίας έχει την δυνατότητα να επιλέξει ανάμεσα στην είσοδο ή την εγγραφή του στην εφαρμογή, αλλά και ως απλός επισκέπτης (*guest*) του widjet. Ταυτόχρονα, ανοίγει ένα session ανάμεσα στον πελάτη ή τον επισκέπτη με το API της πλατφόρμας Customers.

Εφόσον ο πελάτης της εταιρίας έχει χρησιμοποιήσει στο παρελθόν το widjet, αυτομάτως ενημερώνεται (ο πελάτης) για το καταλληλότερο μέγεθος ρούχου με βάση τα κριτήρια που έχει δηλώσει τις προηγούμενες φορές. Ταυτόχρονα, αν θέλει να αλλάξει τα προσωπικά του κριτήρια επιλογής ρούχου ή είναι ένας νέος χρήστη που μόλις εγγράφηκε στην εφαρμογή, έχει την δυνατότητα να διαμορφώσει τα βασικά χαρακτηριστικά του, όπως η ηλικία, το ύψος, τα κιλά, το πόσο εφαρμοστό θέλει να είναι το ρούχο του και άλλες παρεμφερείς ιδιότητες, ώστε να ενημερωθεί για το κατάλληλο μέγεθος του προϊόντος.

3.2.2. Σενάριο και λειτουργίες πλατφόρμας Companies

Στην εικόνα 3.2 παρουσιάζεται ένα σενάριο περιπτώσεων χρήσης της επιχείρησης ως πελάτης της εφαρμογής με την πλατφόρμα **Companies**.

Εικόνα 3.2: Περιπτώσεις χρήσης Πελάτης – Πλατφόρμα Companies

Μερικά από τα βασικά χαρακτηριστικά και λειτουργίες της πλατφόρμας Companies είναι τα παρακάτω:

- Οι πελάτες της είναι οι ίδιες οι εταιρίες που διαθέτουν τα προϊόντα τους σε ένα ηλεκτρονικό κατάστημα διαδικτυακά.
- Η είσοδος στις λειτουργίες της πλατφόρμας επιτρέπετε αποκλειστικά και μόνο στους χρήστες (*users*) της πλατφόρμας Companies.
- Εάν ένας χρήστης διατηρεί ηλεκτρονικό κατάστημα πώλησης ειδών ρουχισμού τότε μπορεί να εισάγει το widjet που διατίθεται από την πλατφόρμα στην διαδικτυακή σελίδα της εταιρίας του.
- Η ενεργοποίηση και διασύνδεση του widjet με την πλατφόρμα Companies επιτυγχάνεται με το ανάλογο API key που δίνεται από την πλατφόρμα.
- Ο χρήστης της πλατφόρμας έχει την δυνατότητα να εισάγει ένα νέο προϊόν συμπληρώνοντας κατάλληλα τα ανάλογα πεδία των χαρακτηριστικών που διέπουν το προϊόν προς πώληση.
- Ο χρήστης της πλατφόρμας έχει την δυνατότητα να δηλώσει πως διαθέτει στο ηλεκτρονικό του κατάστημα ένα ήδη υπάρχον προϊόν κάποιας άλλης εταιρίας κάνοντας είτε απλή δήλωση κατοχής του προϊόντος (pin product), είτε

δημιουργώντας μία μορφή διακλάδωσης (*fork*) σε περίπτωση που διαθέτει επιπλέον χαρακτηριστικά (χρώμα, μεγέθη κλπ.) ή θέλει να ενημερώσει επιπλέον το προϊόν.

- Εάν μία εταιρία θέλει να διασυνδέσει τα προϊόντα της εφαρμογής με το widjet της θα πρέπει να συμπληρώσει τα ανάλογα πεδία διασύνδεσης του προϊόντος, όπως την σελίδα του προϊόντος στο ηλεκτρονικό της κατάστημα και το δικό της αναγνωριστικό προϊόντος (*product_id*).
- Ο χρήστης έχει την δυνατότητα να αναζητήσει όλα τα προϊόντα που διαθέτει η εφαρμογή καθώς και να δει τις κριτικές των πελατών που έχουν αγοράσει τα προϊόντα ή ακόμα και των μοντέλων που τα έχουν δοκιμάσει. Με αυτό τον τρόπο δίνεται η δυνατότητα στην εταιρία να κάνει μία επιπλέον έρευνα αγοράς σε περίπτωση που θέλει να διαθέσει κάποιο νέο προϊόν.

3.2.3. Σενάριο και λειτουργίες πλατφόρμας Customers

Στην εικόνα 3.3 παρουσιάζεται ένα σενάριο περιπτώσεων χρήσης του πελάτη με την πλατφόρμα **Customers**.

Εικόνα 3.3: Περιπτώσεις χρήσης Πελάτης – Πλατφόρμα Customers.

Μερικά από τα βασικά χαρακτηριστικά και λειτουργίες της πλατφόρμας Customers είναι τα παρακάτω:

- Στην πλατφόρμα Customers, ο χρήστης είναι και πιθανός πελάτης της εταιρίας που έχει δηλώσει τα προϊόντα της στην πλατφόρμα Companies και χρησιμοποιεί το widjet στην διαδικτυακή ιστοσελίδα της.
- Η είσοδος στις λειτουργίες της πλατφόρμας επιτρέπετε αποκλειστικά και μόνο στους χρήστες (*users*) της πλατφόρμας Customers.
- Ο χρήστης μπορεί να διαμορφώσει το βασικό του προφίλ όπως την ηλικία, τον τόπο διαμονής, το email και τον κωδικό σύνδεσης.
- Μέσω του widjet και της διασύνδεσής του με το API Customers, σε περίπτωση που ο χρήστης πραγματοποιήσει μία αγορά σε ένα ηλεκτρονικό κατάστημα ή έχει ενδιαφερθεί για κάποιο προϊόν, του δίνεται η δυνατότητα να προσφέρει κριτική για το προϊόν με βάση τα χαρακτηριστικά του.
- Ο χρήστης, έχει την δυνατότητα να εισάγει πέρα από τα υποχρεωτικά πεδία για την έναρξη της λειτουργίας της εφαρμογής (ύψος, κιλά, τρόπος εφαρμογής κλπ.), πλήθος άλλων επιλογών και σωματικών μετρήσεων με στόχο να επιτύχει ακόμα καλύτερα αποτελέσματα στην επιλογή των μεγεθών, μέσω του ανάλογου αλγορίθμου εύρεσης καλύτερου δυνατού μεγέθους της εφαρμογής.
- Ο χρήστης της πλατφόρμας έχει την δυνατότητα να δημιουργήσει πολλαπλά μοντέλα σωματικών χαρακτηριστικών. Μπορεί για παράδειγμα, μέσω της πλατφόρμας να διατηρεί σε ένα λογαριασμό όλα τα σωματικά μοντέλα της οικογένειάς του και αναλόγως να επιλέγει το κατάλληλο μέγεθος και είδος.
- Ο χρήστης της πλατφόρμας μπορεί να ζητήσει (*pin user*) το χαρακτηριστικό (*user_id*) ενός άλλου χρήστη ή τα ήδη υπάρχοντα διαθέσιμα (μοντέλα) από τις εταιρίες και να το χρησιμοποιήσει στην εύρεση καλύτερου δυνατού μεγέθους προϊόντος. Με αυτό τον τρόπο διευκολύνει την δυνατότητα εύρεσης προϊόντος και αγοράς μεγέθους σε φίλους που δεν γνωρίζει με βεβαιότητα τα σωματικά χαρακτηριστικά τους.
- Με βάση τις αγορές και τις προτιμήσεις του χρήστη, η πλατφόρμα μπορεί να του προτείνει προϊόντα παρόμοια με αυτά που έχει ήδη επισκεφθεί στην ιστοσελίδα της εταιρίας ή να τον ενημερώσει για δημοφιλή προϊόντα άλλων χρηστών.

3.3. Δομικά στοιχεία εφαρμογής

Παρακάτω θα παρουσιάσω μερικά από τα κεντρικά δομικά στοιχεία που συνδυάστηκαν για την ομαλή λειτουργία της εφαρμογής σε μορφή σχημάτων, όπως την δομή και τους πίνακες των βάσεων δεδομένων που χρειάστηκαν να εφαρμοστούν, καθώς και κομμάτια από το εσωτερικό των **controllers**, **models**, **routes** και **views** που διαχειρίζονται τις λογικές λειτουργίες της εργασίας. Παράλληλα, με την ανάπτυξη των δομικών στοιχείων θα αναφέρονται και ανάλογα βήματα με στιγμιότυπα κώδικα για να γίνει εμφανής η ακολουθία που πραγματοποιείται στο

backend του συστήματος και η διαδρομή των δεδομένων ανάμεσα στα δομικά στοιχεία.

3.3.1. Βάσεις Δεδομένων

Για την ανάπτυξη των δύο πλατφόρμων χρησιμοποιήθηκαν δύο βάσεις δεδομένων. Η μία ονομάζεται **customers** και απαρτίζεται από πίνακες σχετικούς με την πλατφόρμα Customers και η δεύτερη ονομάζεται **companies** και απαρτίζεται από πίνακες σχετικούς με την πλατφόρμα Companies. Ο διαχωρισμός έχει γίνει με σκοπό να μην υπάρχουν δυσκολίες στις ομαδοποιήσεις και στις ιδιότητες των χρηστών και άλλων λειτουργιών όπως η ασφάλεια, η συντήρηση και η μελλοντική ανάπτυξη της εφαρμογής. Οι πίνακες που σχετίζονται με την αποθήκευση των δεδομένων και των λειτουργιών των προϊόντων καθώς και των reviews έχουν δημιουργηθεί στην βάση customers της πλατφόρμας Customers για ευκολότερη και γρηγορότερη ανάπτυξη της εφαρμογής. Σε απώτερο χρόνο θα πρέπει πιθανώς να διασπαστούν οι πίνακες αυτοί και να δημιουργηθούν επιπλέον αντίστοιχες βάσεις δεδομένων, αναλόγως τον τρόπο που θα αναπτυχθεί στο μέλλον η εφαρμογή.

Παράλληλα, λόγω της χρήσης της σχεσιακής βάσης δεδομένων MariaDB βασισμένη σε αρχιτεκτονική InnoDB είναι εφικτή η δυνατότητα να προσπελάσουμε και να διασυνδέσουμε δεδομένα και στοιχεία ανάμεσα σε διαφορετικές βάσεις του ίδιου συστήματος. Με αυτό τον τρόπο μπορούμε να δημιουργήσουμε συσχετισμούς και περιορισμούς ξένων κλειδιών (*foreign key constraints*) όπως φαίνεται στην εικόνα 3.4.

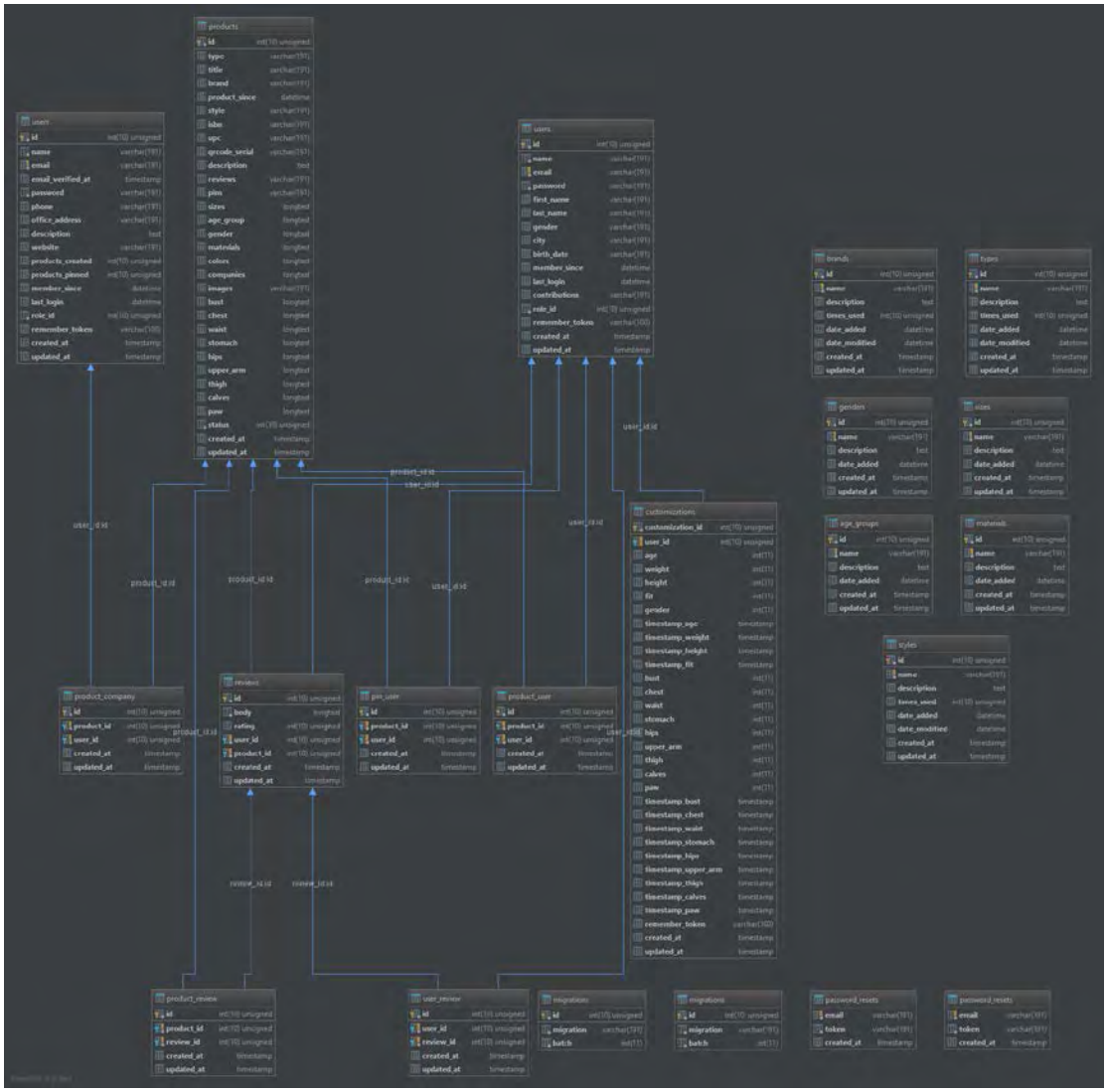
```
$table->integer('user_id')->unsigned();
$table->integer('product_id')->unsigned();

$table->foreign('product_id')->references('id')->on('products');
$table->foreign('user_id')->references('id')->on('diffme.users');
```

Εικόνα 3.4: Στιγμιότυπο κώδικα κλάσης CreateReviewTable πλατφόρμας Companies.

Στην εικόνα 3.5 παρουσιάζονται οι βασικοί πίνακες των βάσεων δεδομένων με τα στοιχεία τους, που δημιουργήθηκαν για την ανάπτυξη των λειτουργιών που αναφέραμε προηγουμένως ανάμεσα στις δύο πλατφόρμες της εφαρμογής.

Βλέπουμε πως έχουν χρησιμοποιηθεί πίνακες που διασυνδέουν τα αναγνωριστικά άλλων πινάκων (*pivot tables*), όπως ο companies.product_user που αποθηκεύει το αναγνωριστικό ενός προϊόντος που χρησιμοποιεί ένας χρήστης, καθώς και το αναγνωριστικό του ίδιου του χρήστη ώστε να γίνει η ανάλογη διασύνδεση. Ταυτόχρονα, υπάρχουν πίνακες που χρησιμοποιεί η laravel όπως οι migrations για την διατήρηση του ιστορικού των αλλαγών κατά την διάρκεια μορφοποίησης των πινάκων σε κάθε βάση δεδομένων. Τέλος, βλέπουμε και τους σχετικούς πίνακες όπου

αποθηκεύονται επιμέρους δεδομένα για την σχεδίαση και σωστή λειτουργία της εργασίας, όπως οι company.types και company.styles που περιέχουν το σύνολο των τύπων (t-shirt, coat κλπ.) και των τρόπων εφαρμογής (denim, slim κλπ.) ενός προϊόντος.



Εικόνα 3.5: Σχεδιάγραμμα / schema βάσεων δεδομένων Customers και diffme

Επίσης, θα πρέπει να επισημάνουμε πως με την τελευταία έκδοση της MariaDB πλέον υπάρχει η δυνατότητα χρήσης πεδίου μορφής json (longtext εικόνα 3.5). Ένα πεδίο πάνω στο οποίο βασίστηκε η χρήση λιγότερων στοιχείων και πινάκων διασύνδεσης που θα κατακερμάτιζαν σειριακά αρκετούς ήδη υπάρχοντες, αλλά και στην εύκολη χρήση και εξαγωγή των δεδομένων τους, επιταχύνοντας τις διεκπεραιώσεις διαφόρων λειτουργιών. Με το πεδίο αυτό, έχουμε την δυνατότητα να εμφωλεύσουμε σε μία σειρά πολλαπλά δεδομένα και στοιχεία άλλων πινάκων όπως βλέπουμε στην εικόνα 3.6.

Εικόνα 3.6: Δεδομένα πεδίου materials του customers.products σε μορφή json

Τα βήματα που ακολουθήθηκαν για τον σχηματισμό του πίνακα users της βάσεων δεδομένων customers είναι τα παρακάτω:

- Μέσω του CLI Artisan δίνεται η εντολή "*php artisan make:migration create_users_table*" όπου δημιουργείται ένα νέο αρχείο σε php στο εσωτερικό του φακέλου migrations της laravel και περιέχει την βασική κλάση *CreateUsersTable*.
- Στο εσωτερικό της κλάσης έχουμε την δυνατότητα να προσδιορίσουμε πλήρως τα απαραίτητα στοιχεία και πεδία του πίνακα users που πρόκειται να δημιουργήσουμε, καθώς και να εισάγουμε επιπλέον λειτουργίες όπως διαγραφής ή ανανέωσης του εν λόγω πίνακα.
- Αφού εισάγουμε τα απαραίτητα στοιχεία και πεδία του πίνακα, με την εντολή στο CLI, "*php artisan make:migration*", το framework μεταφράζει των κώδικα σε SQL κατασκευάζοντας το ανάλογο **query** και δημιουργεί τον πίνακα *users* με τα αντίστοιχα πεδία του.

Στην εικόνα 3.7 βλέπουμε την κλάση *CreateUsersTable* του αντίστοιχου αρχείου *create_users_table.php* της πλατφόρμας Customers που αναπτύχθηκε στον φάκελο αρχείων migrations της laravel για την δημιουργία του πίνακα users. Με παρόμοιο τρόπο έχουν αναπτυχθεί και οι περισσότεροι πίνακες των πλατφορμών customers και companies. Οι πίνακες αυτοί, με τις αντίστοιχες κλάσεις τους μας οδηγούν ιεραρχικά στο επόμενο δομικό στοιχείο, την δημιουργία των model.

```
class CreateUsersTable extends Migration
{
    public function up()
    {
        if(!Schema::hasTable('users')) {
            Schema::create('users', function (Blueprint $table) {

                /* Auth Data */

                $table->increments('id');
                $table->string('name');
                $table->string('email')->unique();
                $table->string('password');

                /* Personal Info */

                $table->string('first_name')->nullable();
                $table->string('last_name')->nullable();
                $table->string('gender')->nullable();
                $table->string('city')->nullable();
                $table->string('birth_date')->nullable();
                $table->dateTime('member_since')->nullable();
                $table->dateTime('last_login')->nullable();
                $table->string('contributions')->nullable();

                $table->integer('role_id')->unsigned()->default(3); //Simple user

                $table->rememberToken();
                $table->timestamps();
            });
        }
    }

    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

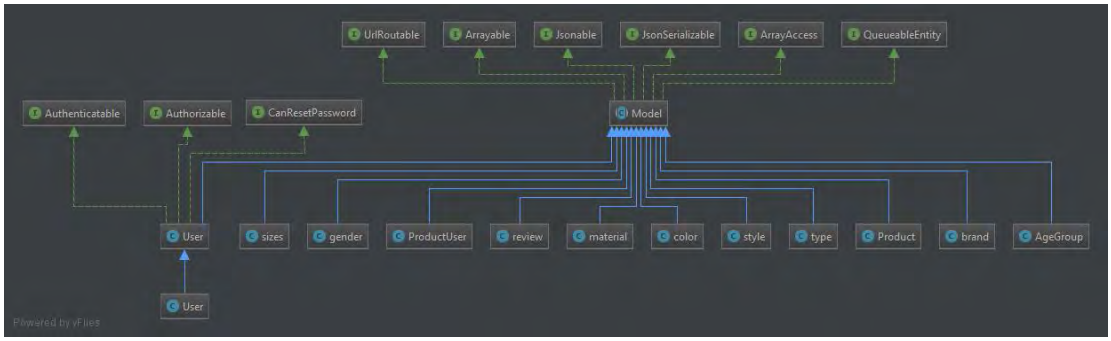
Εικόνα 3.7: Η κλάση CreateUsersTable του αρχείου create_users_table.php

3.3.2. Models

Στα **model** όπως αναφέρθηκε στην ενότητα 2.3 δηλώνονται και δημιουργούνται οι ανάλογοι λογικοί συσχετισμοί, λειτουργίες και μεταβλητές των πεδίων των πινάκων των βάσεων δεδομένων για την χρήση τους μέσω του framework laravel.

Με την δημιουργία ενός νέου αντικείμενου model επιτυγχάνουμε να δηλώσουμε τους κανόνες και τους τρόπους με τους οποίους το υπόλοιπο σύστημα θα χειρίζεται της μεταβλητές και τις διεργασίες του, καθώς και τις πιθανές αλληλεξαρτήσεις με μεταβλητές άλλων model. Χρησιμοποιώντας το *Eloquent ORM* βοηθάει τον προγραμματιστή να δημιουργεί με ευκολία τα απαραίτητα ερωτήματα για την επικοινωνία των model με τις βάσεις δεδομένων.

Στην εργασία μας σχεδόν για κάθε πίνακα είναι απαραίτητο να υπάρχει ένα model, ώστε να μπορούν να περνάνε τα δεδομένα των βάσεων από τα model μέσω των ανάλογων controller στα αντίστοιχα δομικά στοιχεία views με μεγαλύτερη ταχύτητα και ασφάλεια. Στην εικόνα 3.8 βλέπουμε ένα σχεδιάγραμμα των model που χρησιμοποιεί η πλατφόρμα companies.

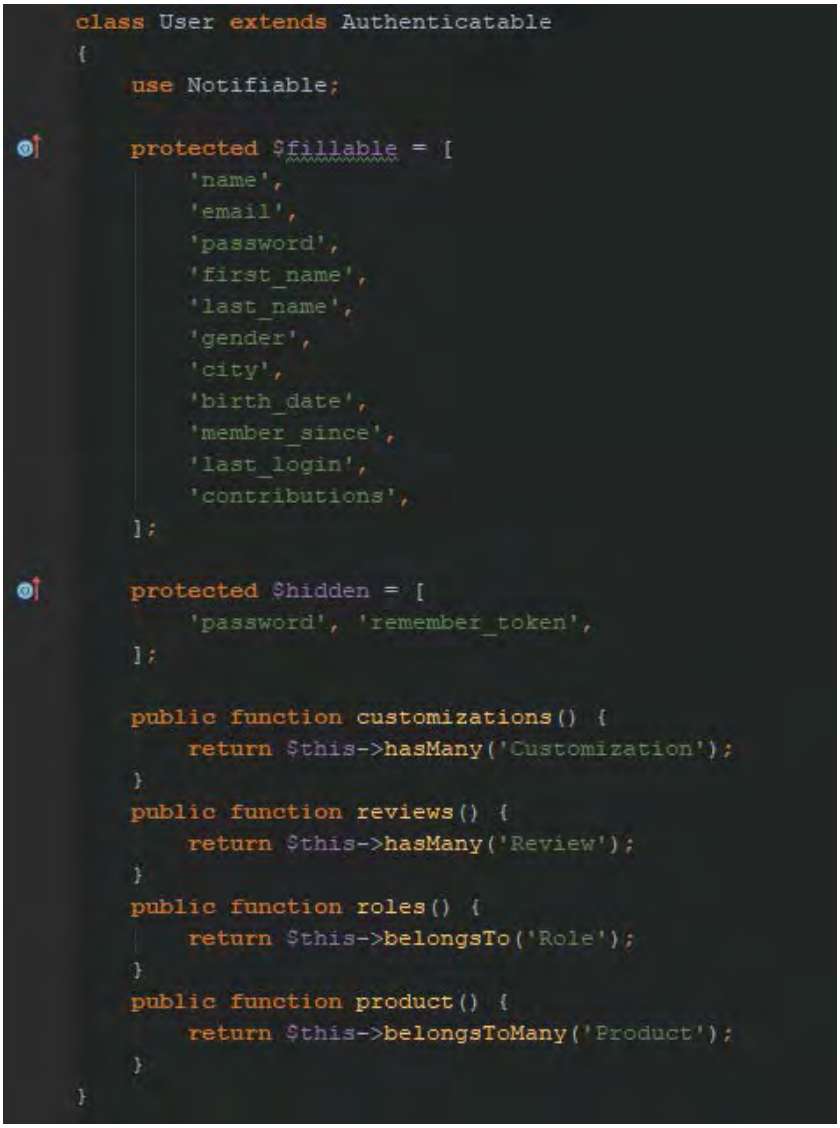


Εικόνα 3.8: Τα model της πλατφόρμας Companies

Στην εικόνα 3.8 παρατηρούμε πως το model User χρησιμοποιεί επιπλέον λειτουργίες για την ταυτοποίησή των χρηστών κατά την είσοδο αλλά και στο εσωτερικό της εφαρμογής του χρήστη. Υπάρχουν επίσης models που χρησιμοποιούνται μόνο σε περιπτώσεις αναζήτησης όταν ένας χρήστης χρειάζεται να μορφοποιήσει ή δημιουργήσει ένα προϊόν. Για παράδειγμα εάν θέλει ένας χρήστης να εισάγει ένα νέο μέγεθος (*size*) ο ανάλογος controller καλεί το model size να του επιστρέψει το σύνολο των μεγεθών που χαρακτηρίζουν τον τύπο (*type*) και είδος (*style*) του προϊόντος. Ταυτόχρονα, υπάρχουν και models που θα πρέπει να διεκπεραιώσουν όλες τις **CRUD** λειτουργίες της εφαρμογής, όπως το User και Product.

Ας επανέλθουμε στην ακολουθία των βημάτων της ενότητας 3.3.2 που σταματήσαμε στον σχηματισμό του πίνακα users της πλατφόρμας Customers. Τα ανάλογα βήματα για την δημιουργία του model User ώστε να δημιουργηθεί το αντικείμενο μέσω του οποίου θα επεξεργάζονται τα δεδομένα του πίνακα users είναι τα παρακάτω:

- Μέσω του CLI Artisan δίνεται η εντολή "*php artisan make:model User*" όπου δημιουργείται ένα νέο αρχείο User.php στον χώρο app του framework και περιέχει την βασική κλάση User.
- Στο εσωτερικό της κλάσης δηλώνουμε τα στοιχεία του πίνακα ως μεταβλητές που θα μπορεί να ενημερώσει ο χρήστης ή το framework και τους αντίστοιχους κανόνες που τις διέπουν, καθορίζοντας τον τρόπο με τον οποίο ο **Eloquent ORM** θα έχει πρόσβαση στους πίνακες ώστε να δημιουργήσει τα ανάλογα αντικείμενα στην php.
- Επίσης μέσα στην κλάση model δηλώνουμε τις ανάλογες συσχετίσεις ανάμεσα στα πεδία διαφορετικών πινάκων, ώστε να μπορούμε να τις συνδέσουμε πλήρως με τις λειτουργίες του Eloquent ORM.



```php
class User extends Authenticatable
{
    use Notifiable;

    protected $fillable = [
        'name',
        'email',
        'password',
        'first_name',
        'last_name',
        'gender',
        'city',
        'birth_date',
        'member_since',
        'last_login',
        'contributions',
    ];

    protected $hidden = [
        'password', 'remember_token',
    ];

    public function customizations() {
        return $this->hasMany('Customization');
    }
    public function reviews() {
        return $this->hasMany('Review');
    }
    public function roles() {
        return $this->belongsTo('Role');
    }
    public function product() {
        return $this->belongsToMany('Product');
    }
}
```

Εικόνα 3.9: Κλάση User αρχείου User.php

Στην εικόνα 3.9 βλέπουμε την κλάση User του model User. Γενικώς ο Eloquent ORM του model προστατεύει εξαρχής όλα τα στοιχεία του πίνακα από πιθανές μαζικές αναθέσεις (mass-assignment) που θα έκαναν ευάλωτο το σύστημα. Στην κλάση User η μεταβλητή $fillable ενημερώνει πως υπάρχει η δυνατότητα μαζικής ανάθεσης των συγκεκριμένων στοιχείων του πίνακα. Ταυτόχρονα με την δήλωση μίας μεταβλητής ως $hidden επιτυγχάνεται η δυνατότητα να μπορεί να επεξεργαστεί τα δεδομένα ο ORM αλλά να μην τα εισάγει σε πίνακές ή σε μορφοποιήσεις json για αποφυγή μετάδοσή τους από και προς το σύστημα.

Τέλος, στην κλάση παρατηρούμε πως αναφέρονται και τέσσερις διεργασίες που η χρήση τους είναι να δηλώσουν τους ανάλογους συσχετισμούς ανάμεσα σε διαφορετικά model. Ένας από αυτούς, είναι και ο συσχετισμός customizations, όπου δηλώνει πως σε κάθε χρήστη, User model, ανήκουν πολλά Customization model, καθώς όπως αναφέραμε στα σενάρια της ενότητας 3.2.3, ένα προφίλ μπορεί να διαθέτει πολλαπλά μοντέλα σωματικών χαρακτηριστικών. Αντίστοιχα γίνεται και η

ανάλογη δήλωση στο model Customization, καθώς κάθε σωματικό χαρακτηριστικό (*Customization model*) υπόκειται σε έναν μόνο χρήστη (model User).

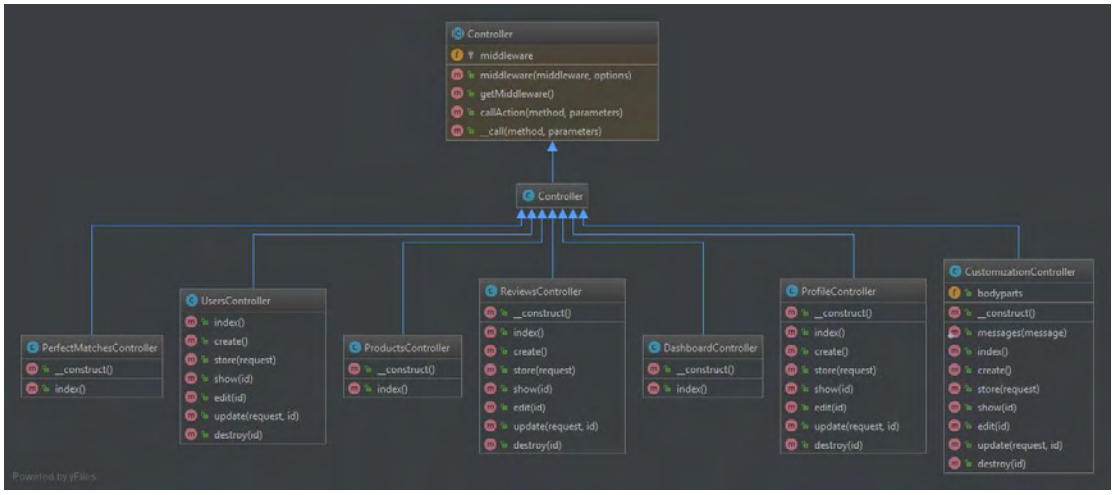
Με παρόμοιο τρόπο έχουν αναπτυχθεί και τα υπόλοιπα model των δύο πλατφόρμων της εφαρμογής. Επόμενο βήμα μετά την ανάπτυξη των model είναι η δημιουργία των απαραίτητων δομικών στοιχείων controller.

3.3.3. Controllers

Στην εργασία αυτή, μέσω των controller αναπτύσσεται και η λογική που μεταφέρει τα δεδομένα ανάμεσα στο model και το view. Αν και χρησιμοποιείται σε αρκετές περιπτώσεις η απευθείας μετάδοση στοιχείων από το model στο view, οι controllers χρησιμοποιούνται για να υλοποιηθούν τα περισσότερα αιτήματα και απαντήσεις (*requests-replies*) ανάμεσα στον χρήστη και την πλατφόρμα.

Μετά από ένα αίτημα του χρήστη προς την εφαρμογή, τα routes, συλλέγουν και ελέγχουν τις πληροφορίες του αιτήματος και ανακατευθύνουν το αίτημα στον αντίστοιχο controller. Αυτός, με την ανάλογη λογική που έχει αναπτυχθεί ελέγχει και έπειτα εξυπηρετεί το αίτημα του χρήστη επιστρέφοντας στο view την ανάλογη απάντηση.

Στην εικόνα 3.10 βλέπουμε τους controller με τις ανάλογες μεθόδους και διεργασίες τους, που έχουν αναπτυχθεί στην πλατφόρμα Customers. Παρατηρούμε πως ορισμένοι controller περιέχουν όλες τις CRUD διεργασίες όπως ο CustomizationController ο οποίος υποστηρίζει τα request του πελάτη σχετικά με τη διαμόρφωση των σωματικών χαρακτηριστικών του.



Εικόνα 3.10: Δομή των Controller της πλατφόρμας Customers.

Υπάρχουν επίσης και controllers που οι CRUD και άλλου είδους διεργασίες είτε είναι περιττές, είτε πραγματοποιούνται απευθείας μέσω των model όπως ο DashboardController. Ο controller αυτός υποστηρίζει την σελίδα Dashboard του

χρήστη, αλλά δεν υπάρχει κανένα αίτημα προς εκτέλεση στην σελίδα, παρά μόνο η ενημέρωση του χρήστη για συγκεκριμένα δεδομένα με απευθείας ανάθεσή τους από τα model ή άλλους controller. Έτσι η μοναδική του λειτουργία είναι να επαληθεύει εάν ο χρήστης είναι συνδεδεμένος ή όχι στην πλατφόρμα (μέσω του *guard Auth*) και να του επιτρέπει την πρόσβαση στην σελίδα dashboard ή να τον ανακατευθύνει στην σελίδα εισόδου χρήστη (*login page*). Στην εικόνα 3.11 βλέπουμε την αντίστοιχη κλάση του DashboardController.

```
class DashboardController extends Controller
{

    public function __construct()
    {
        $this->middleware('auth');
    }


    public function index()
    {
        if(Auth::check()){
            return view('dashboard');
        }
        return view('auth.login');

    }
}
```

Εικόνα 3.11: Η κλάση του controller DashboardController.

Από τα σενάρια έχουμε δηλώσει ότι ένας χρήστης της εφαρμογής Customers έχει την δυνατότητα να διαμορφώνει πολλαπλά σωματικά χαρακτηριστικά του προφίλ του. Αυτό επιτυγχάνεται μέσω της σελίδας customizations της πλατφόρμας. Έχοντας δημιουργήσει τα model User και Customization επόμενο βήμα είναι να αναπτυχθεί κατάλληλα ο CustomizationController για να υποστηρίξει τις λειτουργίες του χρήστη στην σελίδα customizations. Επομένως τα βήματα που εκτελέσθηκαν για την δημιουργία και λογική λειτουργία του, είναι τα παρακάτω:

- Δημιουργήθηκε το ανάλογο αρχείο μέσω του CLI Artisan δίνοντας την εντολή "*php artisan make:controller CustomizationController –resource*". Με αυτή την εντολή το framework δημιουργεί ένα αρχείο στον φάκελο Controllers το οποίο περιέχει την κλάση *CustomizationController* και όλες τις CRUD διεργασίες για να την υποστηρίξουν, όπως οι index, store, edit και update.
- Διαμορφώθηκε η λειτουργία **index** ώστε στο view της σελίδας customization να μπορούν να περάσουν τα ζητούμενα δεδομένα για την ομαλή λειτουργίας της. Αυτά μπορεί να είναι είτε στοιχεία του χρήστη που μας τα παρέχει το Customization model είτε στατικοί πίνακες και μεταβλητές για τις επιμέρους λειτουργίες της σελίδας. Στην εικόνα 3.12 βλέπουμε την αντίστοιχη διεργασία

index του CustomizationController όπου εκτελείται όποτε ο χρήστης μπαίνει στην σελίδα customization.

```php
//default customization is set to 0 asc sets the default on view

public function index()
{
    if(Auth::check()){
        try {
            $customization = Customization::where('user_id',Auth::user()->id)->
            orderBy('default','asc')->
            get()->
            keyBy('customization_id');
            $default_customization = -1;
            $message_handler = 1;
            foreach ( $customization as $custom) {
                if ($custom->default == 0) {
                    $default_customization = $custom->customization_id;
                    $message_handler =  2;
                }
            }
            return view(
                'customization.index',
                ['customization'=>$customization,
                'body_parts'=>$this->bodyparts,
                'default_customization'=>$default_customization,
                'set_errors'=>$this->messages('index',$message_handler)]
            );
        }
        catch (\Illuminate\Database\QueryException $ex) {
            error_log($ex->getMessage());
            view('customization.index',['set_errors'=>$this->messages('index',3)]);
        }
    }
return view('auth.login');
}
```

Εικόνα 3.12: Η διεργασία index του controller CustomizationController.

- Διαμορφώθηκε η διεργασία **update**, η οποία εκτελεί ενέργειες μορφής request-reply όταν ο χρήστης ενημερώνει την σελίδα customization για αλλαγές στα σωματικά χαρακτηριστικά του προφίλ του. Αυτές γίνονται με τη μέθοδο Ajax και δεν ανανεώνουν την σελίδα στις ανάλογες ενέργειες του χρήστη. Επίσης, τα requests αυτής της μορφής περιέχουν μεταβλητές μαζικής ανάθεσης οι οποίες έχουν δηλωθεί στο Customization model και, σε οποιαδήποτε περίπτωση ο φυλλομετρητής προσπαθήσει, είτε εν γνώσει του χρήστη, είτε κακόβουλα να μορφοποιήσει δεδομένα εκτός των δηλωθέντων μεταβλητών, ο Controller αστοχεί στην εκτέλεση της ανάλογης ενέργειας. Στην εικόνα 3.13 βλέπουμε την αντίστοιχη διεργασία update του CustomizationController.

```php
public function update(Request $request, $id)
{
    if(Auth::check()){
        $user_id =auth()->id();
        if(($user_id==$id)&&($user_id==$request->id)){
            try{
                Customization::
                where('customization_id', $request['customization_id'])->
                update($request->
                except('_token','id'));

                return response()->json(['success' => true,'set_errors'=>$this->messages('update',1)]);
            }
            catch (\Illuminate\Database\QueryException $ex){
                error_log($ex->getMessage());
                return response()->json(['success' => false,'set_errors'=>$this->messages('update',2)]);
            }
        };
        return response()->json(['auth' => false,'set_errors'=>$this->messages('auth',0)]);
    };
    return response()->json(['auth' => false,'set_errors'=>$this->messages('auth',1)]);
}
```

Εικόνα 3.13: Η διεργασία update του controller CustomizationController.

- Διαμορφώθηκε η διεργασία **store**, όπου εκτελεί τις ενέργειες request-reply όταν ο χρήστης ζητήσει τη δημιουργία ενός νέου customization στο προφίλ του. Και αυτή η διαδικασία εκτελείτε ασύγχρονα με την μέθοδο Ajax και περιέχει παρόμοια χαρακτηριστικά με την μέθοδο update. Στην εικόνα 3.14 βλέπουμε την αντίστοιχη διεργασία store, του CustomizationController.

```php
public function store(Request $request)
{
    if(Auth::check()) {
        if (Auth::id() == $request->user_id) {

            $check_duplication = Customization::where('name',$request->name)->first();
            if ($check_duplication){
                return response()->json(['success' => false, 'set_errors' => $this->messages('create', 3)]);
            }
            else {
                try {
                    Customization::create([
                        'name' => $request->name,
                        'user_id' => Auth::user()->id,
                        'default' => '1',
                        'gender' => '1'
                    ]);
                    $customization = Customization::
                    where('name',$request->name)->
                    orderBy('default', 'asc')->
                    get()->
                    keyBy('customization_id');
                    error_log($customization);
                    return response()->json(['success' => true,$customization, 'set_errors' => $this->messages('create', 1)]);
                } catch (\Illuminate\Database\QueryException $ex) {
                    error_log($ex->getMessage());
                    return response()->json(['success' => false, 'set_errors' => $this->messages('create', 2)]);
                }
            }
        }
        return response()->json(['success' => false, 'set_errors' => $this->messages('auth', 0)]);
    }
    return response()->json(['success' => false, 'set_errors' => $this->messages('auth', 1)]);
}
```

Εικόνα 3.14: Η διεργασία store του controller CustomizationController.

- Διαμορφώθηκε η διεργασία **destroy**, όπου εκτελεί τις ενέργειες request-reply όταν ο χρήστης ζητήσει τη διαγραφή ενός customization του προφίλ του. Στο view του χρήστη, η εκτέλεση του request επιτυγχάνεται με την μέθοδο Ajax όπως και στις προηγούμενες δύο. Στην εικόνα 3.15 βλέπουμε την αντίστοιχη διεργασία destroy, του CustomizationController.

```
public function destroy($id)
{
    if(Auth::check()) {
        $customization = Customization::find($id);
        error_log($customization);
        if(Auth::id()==$customization->user_id) {
            if ($customization->delete()) {
                return response()->json(['success' => true, 'set_errors' => $this->messages('delete', 1)]);
            }
            else {
                return response()->json(['success' => false,'set_errors'=>$this->messages('delete',2)]);
            }
        }
        return response()->json(['success' => false,'set_errors'=>$this->messages('auth',0)]);
    }
    return response()->json(['success' => false,'set_errors'=>$this->messages('auth',1)]);
}
```

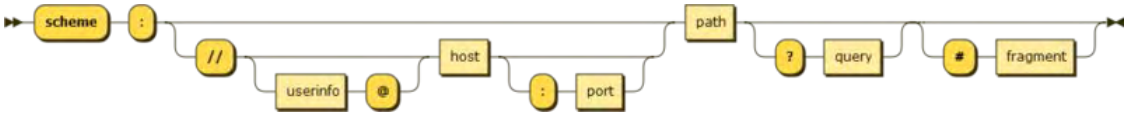
Εικόνα 3.15: Η διεργασία destroy του controller CustomizationController.

Από τις εικόνες 3.12~3.15 βλέπουμε πως δίνεται έμφαση στην συλλογή πληροφοριών για τον τρόπο που εκτελείται η λογική στις διεργασίες των controller. Με την βοήθεια της διεργασίας *messages($message,$num)*, επιστρέφονται οι ανάλογες μεταβλητές ώστε να μπορέσει το view στον φυλλομετρητή να εκτελεί μία διεργασία μορφής χειρισμού σφαλμάτων (*error handling*), ενημερώνοντας τον χρήστη αλλά και τον φυλλομετρητή για την τελική λογική του request-reply στον αντίστοιχο controller.

Με παρόμοιο τρόπο έχουν αναπτυχθεί και οι υπόλοιποι controller για την ομαλή λειτουργία των δύο πλατφορμών, Customers και Companies. Επόμενο βήμα για την λειτουργία του framework της εργασίας είναι η διαμόρφωση και δήλωση των ανάλογων Routes που ανακατευθύνουν τα requests στους αντίστοιχους Controllers.

3.3.4. Routes

Το **URI** (*Uniform Resource Identifier*) ενός HTTP αιτήματος του χρήστη μίας εκ των δύο πλατφορμών, ανακατευθύνεται από τον φυλλομετρητή στον εξυπηρετητή που βρίσκονται οι δύο πλατφόρμες και αφού ελέγχεται η εγκυρότητά του, έρχεται σε «επαφή» με το αντίστοιχο framework Laravel μέσω του **middleware**. Ο middleware περιέχει μηχανισμούς που φιλτράρουν τα αιτήματα μορφής HTTP που εισέρχονται στο framework όπως η ταυτοποίηση του χρήστη, καθώς και μηχανισμούς ώστε να κατευθύνει το αίτημα στον αντίστοιχο controller ή view μέσω των routes. Στην εικόνα 3.16 βλέπουμε την συντακτική δομή ενός URI.



Εικόνα 3.16: Διάγραμμα συντακτικής δομής ενός URI (URI Syntax diagram).

Στην εργασία, έχουν διαμορφωθεί οι **routes** των δύο πλατφορμών ώστε να γίνονται οι αντίστοιχες ανακατευθύνσεις των αιτημάτων του χρήστη. Στην εικόνα 3.17α παρουσιάζονται οι web routes της πλατφόρμας Customers όπου είναι εμφανής και η σύνδεσή τους με τους ανάλογους Controller. Βλέπουμε πως για τις σελίδες products, perfect-matches και dashboard καλούνται μόνο συγκεκριμένες διεργασίες από τους αντίστοιχους Controller, αλλά για τις υπόλοιπες, όπως στην profile και customization καλείται το σύνολο των διεργασιών των CRUD Controller (*Resource Controllers*) μέσω της συνάρτησης resource.

```
Route::get('/', function () {
    if(Auth::check()){
        return view('dashboard');
    }
    return view('welcome');
});

Auth::routes();
Route::middleware(['auth'])->group(function () {

    Route::get('/dashboard', 'DashboardController@index')->name('dashboard');
    Route::get('/products', 'ProductsController@index')->name('products');
    Route::get('/perfect-matches', 'PerfectMatchesController@index')->name('perfect-matches');

    Route::resource('/profile', 'ProfileController');
    Route::resource('/customization', 'CustomizationController');
    Route::resource('/reviews', 'ReviewsController');
    Route::resource('users', 'UsersController');

    // Route::put('/profile/{user_id}','ProfileController@update'); //test error handler

});
```

```
Route::get('/', function () {
    if(Auth::check()){
        return view('dashboard');
    }
    return view('welcome');
});
Auth::routes();

Route::middleware(['auth'])->group(function () {

    Route::get('/dashboard', 'DashboardController@index')->name('dashboard');
    Route::get('/database', 'DatabaseController@index')->name('database');
    Route::get('/reviews', 'ReviewsControlelr@index')->name('reviews');

    Route::resource('profile', 'ProfileController');
    Route::resource('products', 'ProductsController');
    Route::resource('users', 'UsersController');

});
```

Εικόνα 3.17: Πάνω: Στιγμιότυπο κώδικα των routes του αρχείου web.php πλατφόρμας Customers. Κάτω: Στιγμιότυπο κώδικα των routes του αρχείου web.php πλατφόρμας Companies.

Με την χρήση ενός Resource Controller, αναλόγως τον τύπο του αιτήματος του χρήστη (για παράδειγμα μορφής *GET*, *POST* ή *PUT*) ανακατευθύνεται το request στην αντίστοιχη διεργασία του controller. Στην εικόνα 3.18 βλέπουμε τον τρόπο με τον οποίο χειρίζονται τα request μίας σελίδας με κατάληξη «photos» από ένα Resource Controller.

Actions Handled By Resource Controller

Verb	URI	Action	Route Name
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

Εικόνα 3.18: Πίνακας χειρισμού ενεργειών από Controller μορφής Resource.

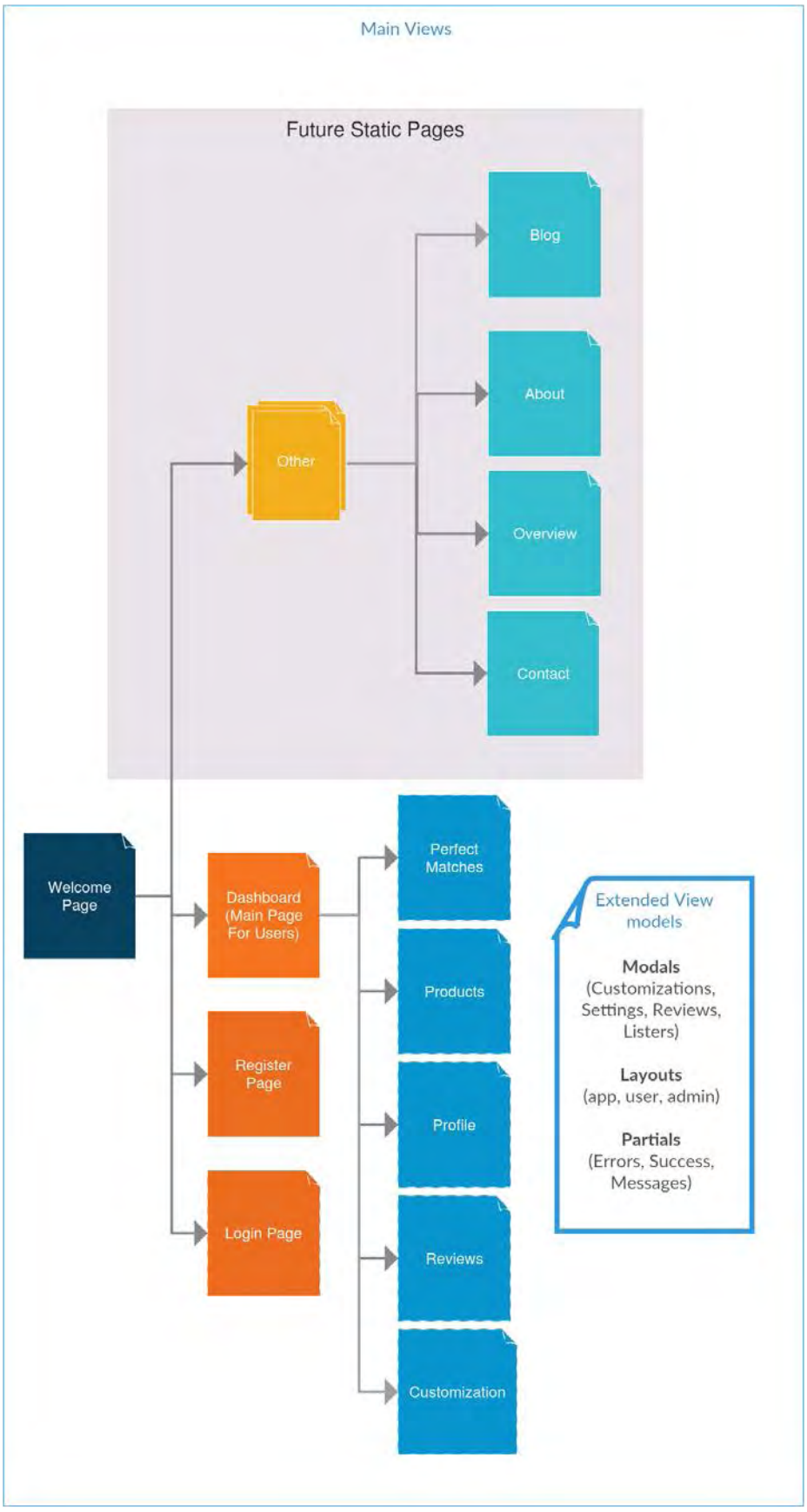
Εφόσον έχουν υλοποιηθεί οι routes των σελίδων που είναι αναγκαίες για την λειτουργία των δύο πλατφορμών, πλέον τα δυναμικά δεδομένα μπορούν να μεταφερθούν με ασφάλεια στα views όπου θα σχεδιαστούν με την χρήση των *HTML*, *CSS* και *Javascript*.

3.3.5. Views

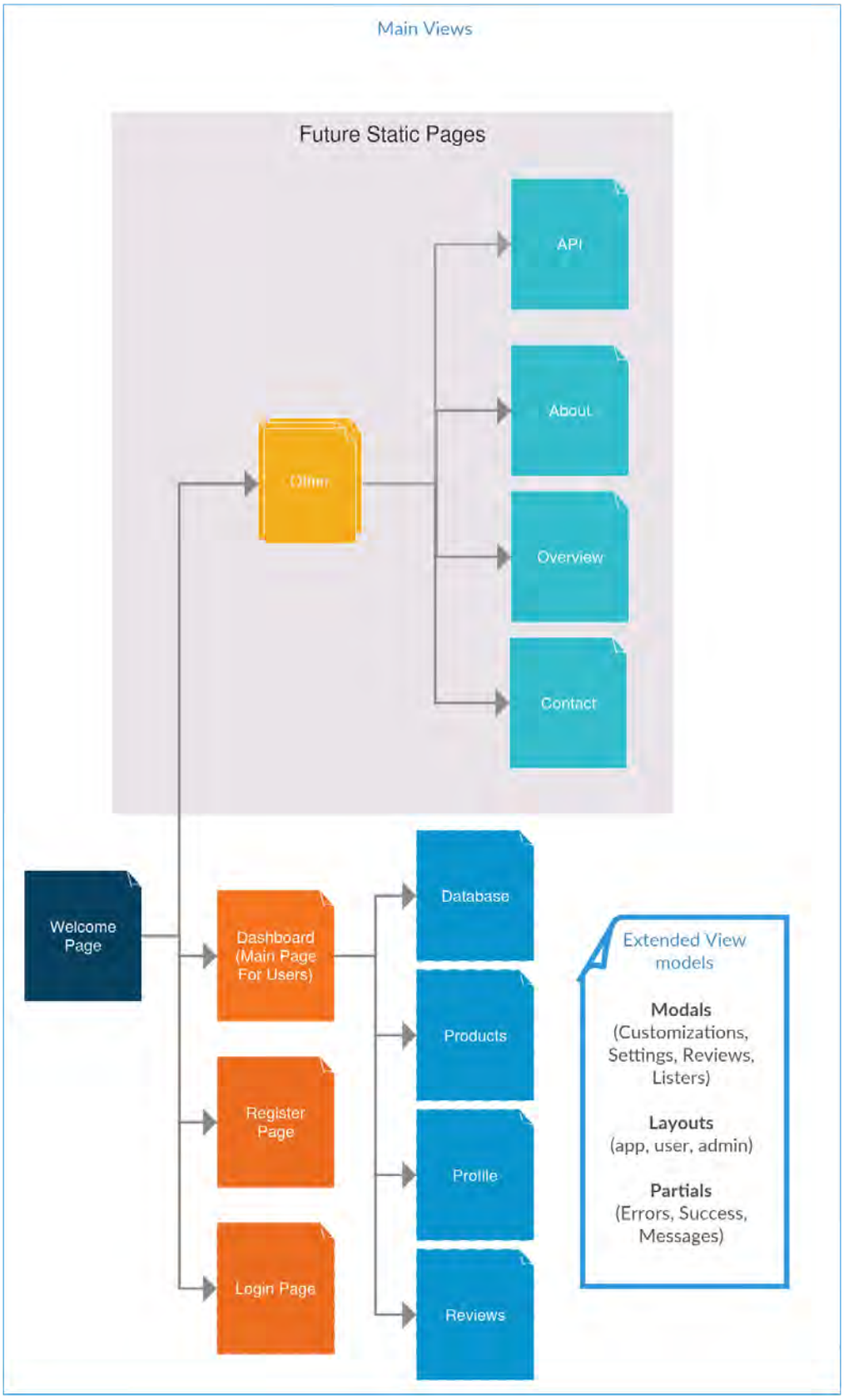
Στην εργασία εφόσον αναπτύχθηκαν και επεξεργάσθηκαν τα απαραίτητα δομικά στοιχεία του backend της εφαρμογής, δημιουργήθηκαν οι αντίστοιχες frontend λειτουργίες ώστε να παρουσιαστούν τα δεδομένα σε ανάλογη μορφή για την καλύτερη εμπειρία του χρήστη. Τα **views** είναι ένα κομμάτι τις εργασίας όπου γίνεται η μετάβαση από την ανάπτυξη στην σχεδίαση των εφαρμογών στη μεριά του χρήστη. Έχει δοθεί μεγάλη έμφαση στον τρόπο σχεδίασης και ανάπτυξης των Views ώστε να παρέχεται στον χρήστη η καλύτερη δυνατή εμπειρία κατά την περιήγησή του στις σελίδες των πλατφορμών.

Η laravel διαθέτει την δυνατότητα χρήσης ενός προτύπου (*template*) με την ονομασία **blade**. Με το πρότυπο αυτό δίνεται η δυνατότητα τα views να αποκτούν λειτουργίες κληρονομικότητας (inheritance), ενθέσεων (sections) και επεκτάσεων (extends) ώστε να μπορούν να συνδυάζονται με επιμέρους δεδομένα, διεργασίες και αρχεία της εφαρμογής. Για την σωστή ανάπτυξη και μετέπειτα συντήρηση των πλατφορμών έχουν χωριστεί οι βασικές τους σελίδες σε επιμέρους στοιχεία. Για παράδειγμα υπάρχει μία ενιαία δομή των σελίδων για τους χρήστες (app.blade.php) που περιέχει την επικεφαλίδα (header) καθώς και κομμάτια του body όπως το μενού και το υποσέλιδο. Παράλληλα, χρησιμοποιούνται και στοιχεία όπως τα success.blade.php messages.blade.php και διάφορα άλλα modals που υποστηρίζουν τις εκάστοτε λειτουργίες κάθε σελίδας.

Στις εικόνες 3.19 και 3.20 παρουσιάζονται μέσω των sitemap των δύο πλατφορμών ένα βασικό μέρος των views που έχουν αναπτυχθεί για την εργασία αυτή. Στα sitemap περιέχονται οι βασικές σελίδες (όπως customization, profile, products) καθώς και επιμέρους στοιχεία (Modals, Layouts, Partials), τα οποία χρησιμοποιούνται από τις βασικές σελίδες ως επεκτάσεις ή ενθέσεις.



Εικόνα 3.19: Sitemap πλατφόρμας Customers.

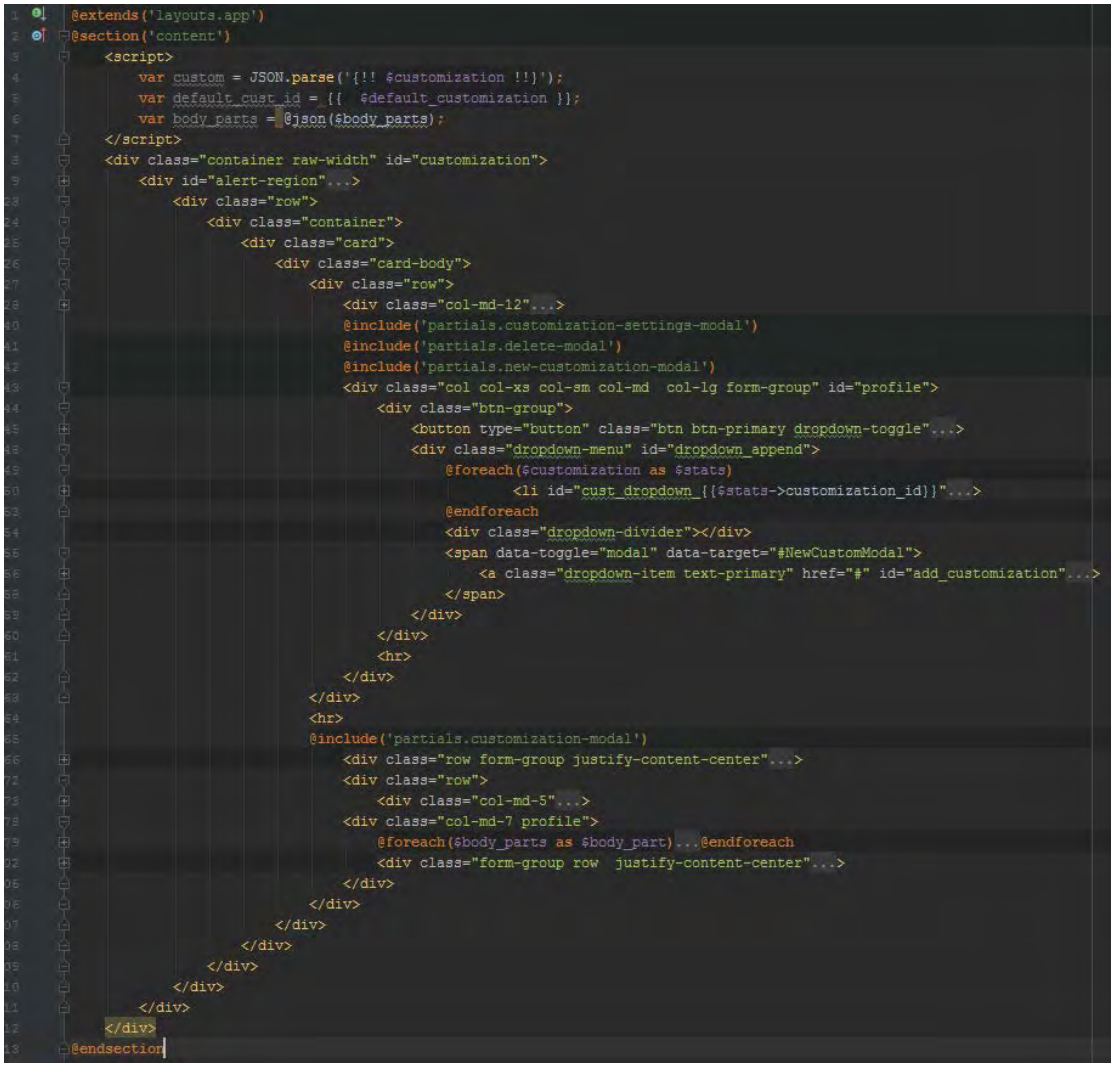


Εικόνα 3.20: Sitemap πλατφόρμας Companies.

Παρακάτω θα αναλύσουμε τα κυριότερα μέρη και την δομή του view customization της πλατφόρμας Customers, δείχνοντας παράλληλα την λογική καθώς και τον τρόπο με τον οποίο εισάγονται τα δεδομένα από τους controller και τις υπόλοιπες λειτουργίες του framework που έχουμε αναφέρει στις προηγούμενες ενότητες, ώστε να διαμορφωθεί ο τελικός κώδικας σε html, για να μπορέσει να διαβαστεί από τον φυλλομετρητή του χρήστη.

Στην εικόνα 3.21 βλέπουμε το αρχείο index.blade.php της σελίδας customization της πλατφόρμας customers, όπου έχουν αναδιπλωθεί αρκετά κομμάτια του κώδικα για να γίνουν εμφανείς ορισμένες από τις παρακάτω λειτουργίες.

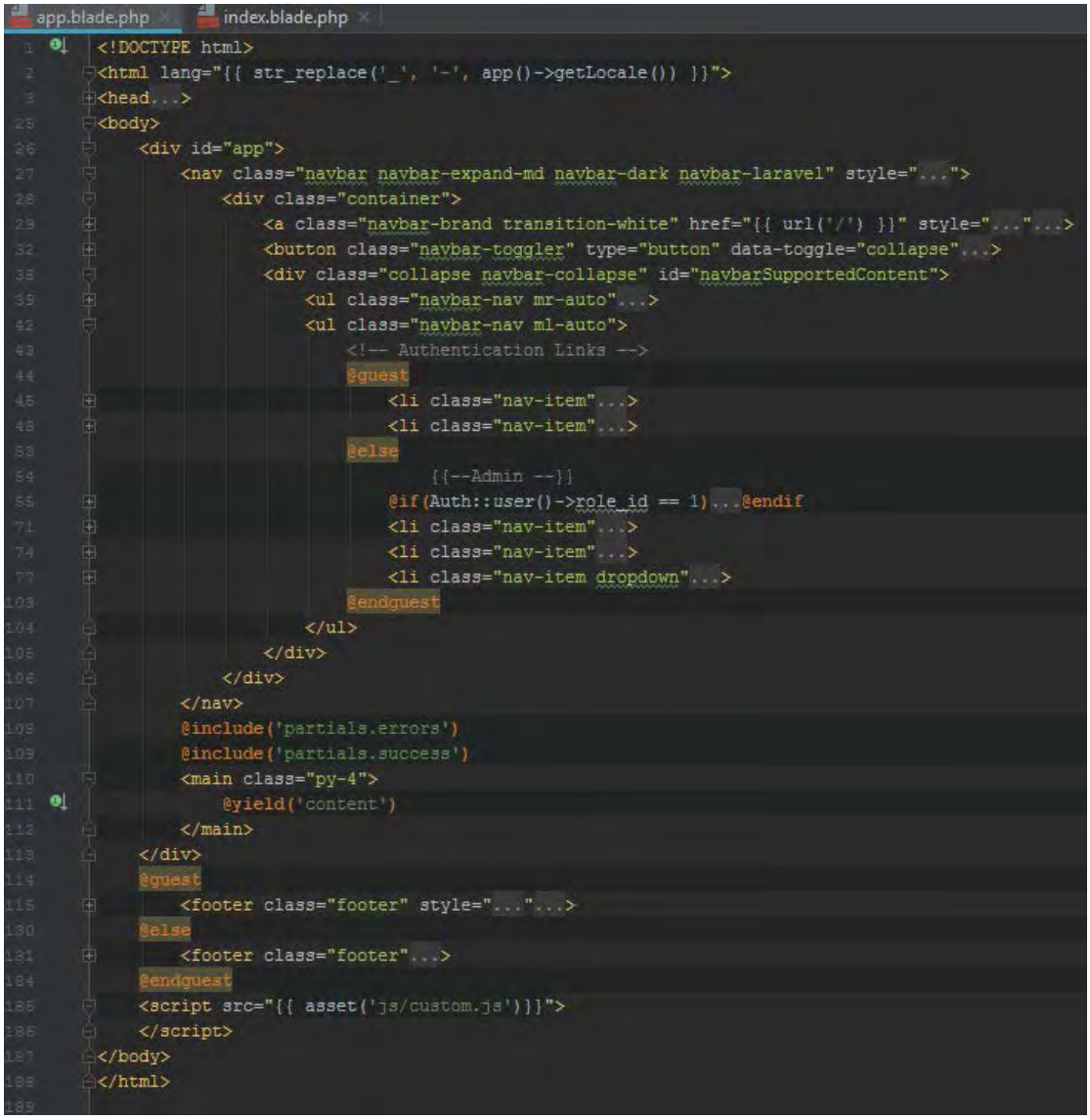
- Η λειτουργία *«@extends('layouts.app')*» δηλώνει πως το παρόν αρχείο αποτελεί επέκταση του αρχείου layouts.app.blade.php, δηλαδή της βασικής δομής των ιστοσελίδων της πλατφόρμας.
- Επίσης, η λειτουργία *«@section('content') … @endsection*» δηλώνει πως το παρόν τμήμα λειτουργεί ως ένθετο με την ονομασία content, καθώς και πως στο section με την λειτουργία *«@include(…)*» εισάγονται και επιμέρους αρχεία μορφής blade.
- Επίσης βλέπουμε πώς περνάνε, αλλά και πώς χρησιμοποιούνται οι ανάλογες μεταβλητές στο view customization από το index του CustomizationController (εικόνα 3.12), όπως στην «*@foreach($customization as $stats)…@endforeach*» όπου εμφανίζεται μία λίστα με την ονομασία του κάθε customization, καθώς και την δημιουργία των αντίστοιχων μεταβλητών και αντικειμένων σε javascript για να μπορούν να εκτελούνται επιμέρους ενέργειες στον φυλλομετρητή του χρήστη *(<script>… </script>).*



Εικόνα 3.21: Στιγμιότυπο κώδικα αρχείου index.blade.php της σελίδας customization πλατφόρμας Customers.

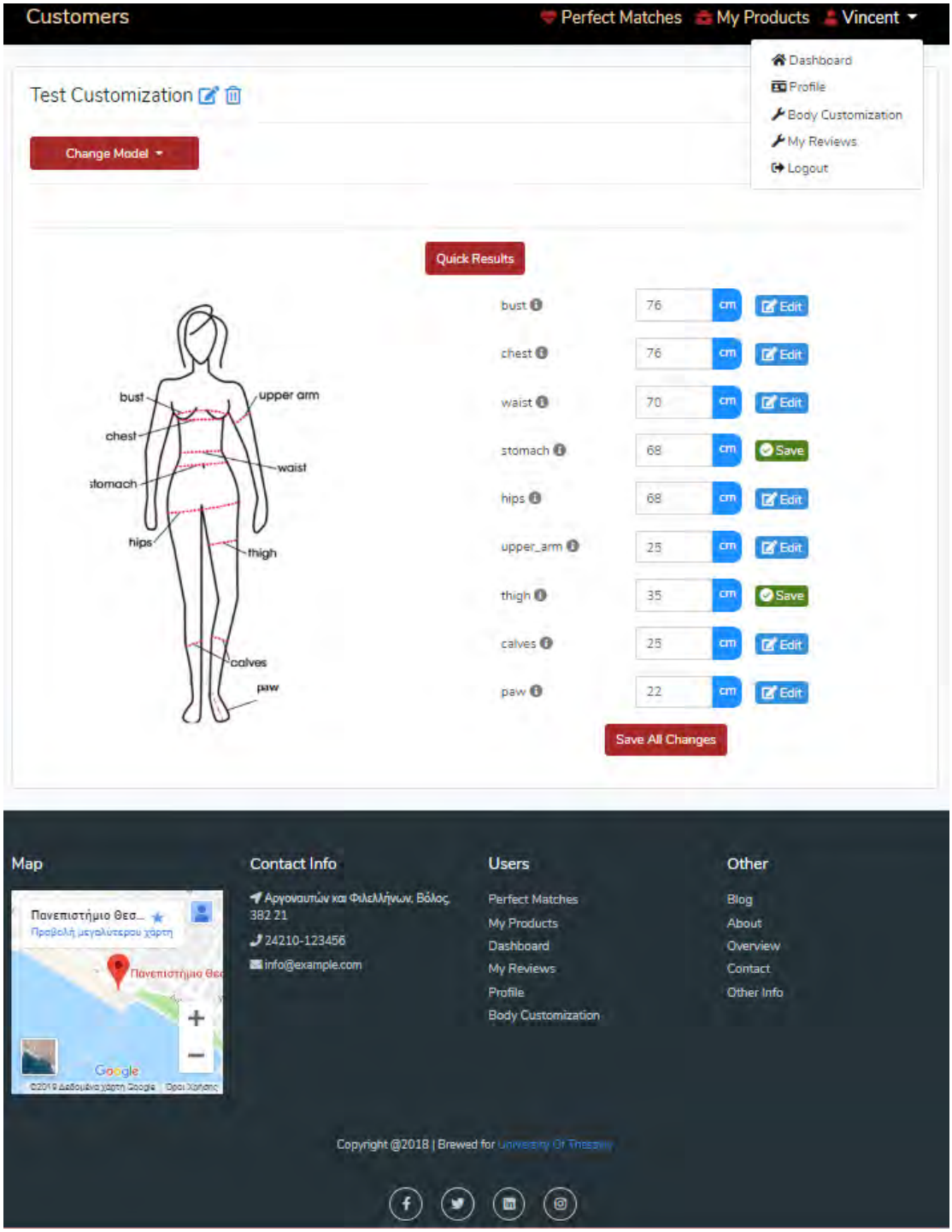
Στην εικόνα 3.22 βλέπουμε το αντίστοιχο τμήμα του *app.blade.php* της πλατφόρμας Customers και αυτό αναδιπλωμένο ώστε να γίνονται εμφανείς ορισμένες από τις βασικές λειτουργίες του όπως:

- Ο τρόπος ταυτοποίησης «*@guest…@else…@endguest*», όπου γίνεται η ανάλογη διαμόρφωση του κεντρικού menu σε περίπτωση που ο χρήστης δεν είναι εγγεγραμμένος ή έχει υπάρξει πρόβλημα κατά την ταυτοποίησή του στην εφαρμογή.
- Ο έλεγχος εάν ο χρήστης που έχει εισέλθει στην σελίδα είναι και διαχειριστής ώστε να διαμορφωθεί ανάλογα το μενού του διαχειριστή.
- Η εντολή «*@yield('content')*», όπου εισάγεται για την δημιουργία του τελικού αρχείου το section content του *index.blade.php* της σελίδας customization.
- Ο τρόπος με τον οποίο εισάγονται μεταβλητές και στοιχεία από την laravel για την σύνταξη του τελικού αρχείου «*{{ asset('js/custom.js')}}*».



Εικόνα 3.22: Στιγμιότυπο κώδικα αρχείου app.blade.php της πλατφόρμας Customers.

Στην εικόνα 3.23 βλέπουμε την προβολή της σελίδας customization από τον φυλλομετρητή έτσι όπως διαμορφώθηκε η σχεδίασή του στο view του framework, για έναν απλό χρήστη που διαμορφώνει ένα γυναικείο μοντέλο σώματος.

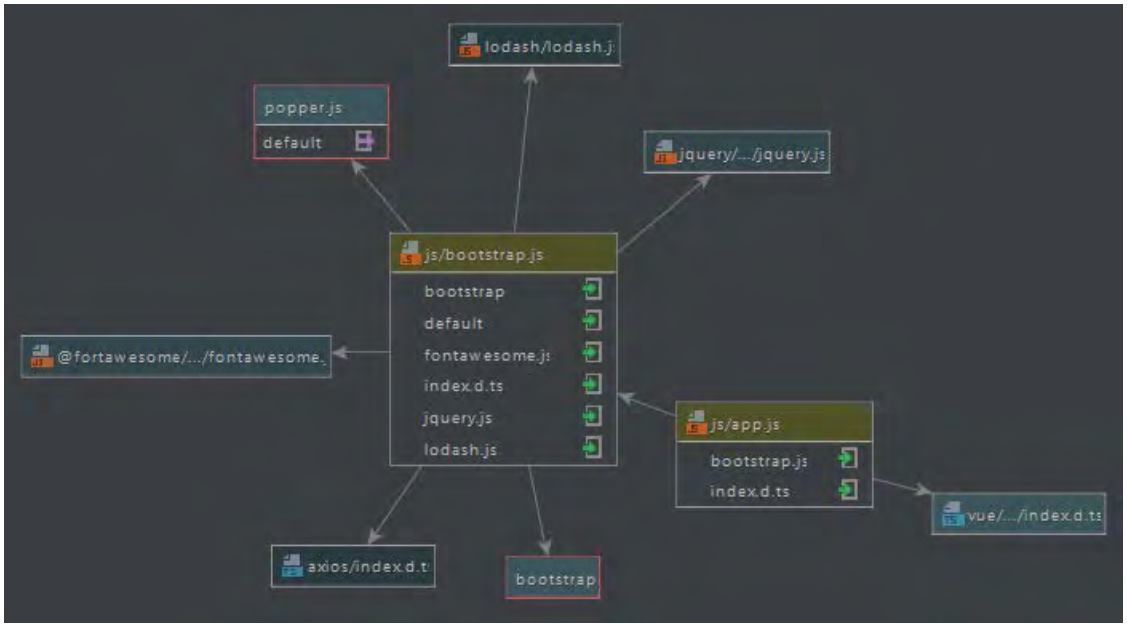


Εικόνα 3.23: Προβολή σελίδας customization της πλατφόρμας customers στον φυλλομετρητή.

Με παρόμοιο τρόπο έχουν διαμορφωθεί και τα υπόλοιπα views των σελίδων των δύο πλατφόρμων.

3.4. Επιπλέον χαρακτηριστικά

Όπως φαίνεται και στα στιγμιότυπα των εικόνων 3.21 και 3.22 το μεγαλύτερο κομμάτι του κώδικα (κυρίως αυτού που είναι συμπτυγμένος) απαρτίζεται από κώδικα HTML ο οποίος διαμορφώνεται κυρίως με βάση το framework της CSS, bootstrap 4, και την αντίστοιχη javascript για να το υποστηρίξει. Ο τελικός κώδικας του framework στο view καθώς και επιπλέων στοιχείων για την λειτουργία της σελίδας, παράγεται με την κλήση του npm από το αρχείο app.js. Στην εικόνα 3.24 βλέπουμε τις επιμέρους μονάδες που χρησιμοποιούνται για τη σωστή διαμόρφωση και προβολή των τελικών αρχείων (CSS και Javascript) της σελίδας στον φυλλομετρητή, αφού πρώτα έχουμε δημιουργήσει τις ανάλογες εξαρτήσεις (dependencies) όπως τα fontawesome και την jquery.



Εικόνα 3.24: Διάγραμμα εξαρτήσεων μονάδων bootstrap.js και app.js.

Κεφάλαιο 4ο
Σχεδίαση Εργασίας

4.1. Εισαγωγή στη Σχεδίαση

Αρχικά, ως Σχεδίαση μίας διαδικτυακής εφαρμογής, αναφερόμαστε σε όλες τις τεχνικές και γλώσσες που χρησιμοποιήθηκαν για το τελικό κομμάτι της ανάπτυξης του view της κάθε πλατφόρμας, δηλαδή του frontend της εφαρμογής όπου συνδυάζονται οι γλώσσες javascript, html και CSS.

Στο παρόν κεφάλαιο θα γίνει διεξοδική ανάλυση του κώδικα που υλοποιεί κομμάτια του σχεδιασμού της εφαρμογής στις τρεις βασικές γλώσσες προγραμματισμού του frontend, καθώς και στα αντίστοιχα framework που χρησιμοποιήθηκαν. Ως κεντρικό σημείο αναφοράς θα έχουμε την σελίδα customization της πλατφόρμας Customers (εικόνα 3.23), δείχνοντας τον τρόπο με τον οποίο σχεδιάστηκε για την αντίστοιχη προβολή της στον φυλλομετρητή. Αν και ο σχεδιασμός των πλατφόρμων έχει γίνει χωρίς designer έχει δοθεί αρκετή προσοχή στον τρόπο με τον έχουν σχεδιαστεί τα views, καθώς η εμπειρία του χρήστη (UX - User Experience) κατά την περιήγησή του σε αυτήν θεωρώ πως είναι κλειδί για την επιτυχία της.

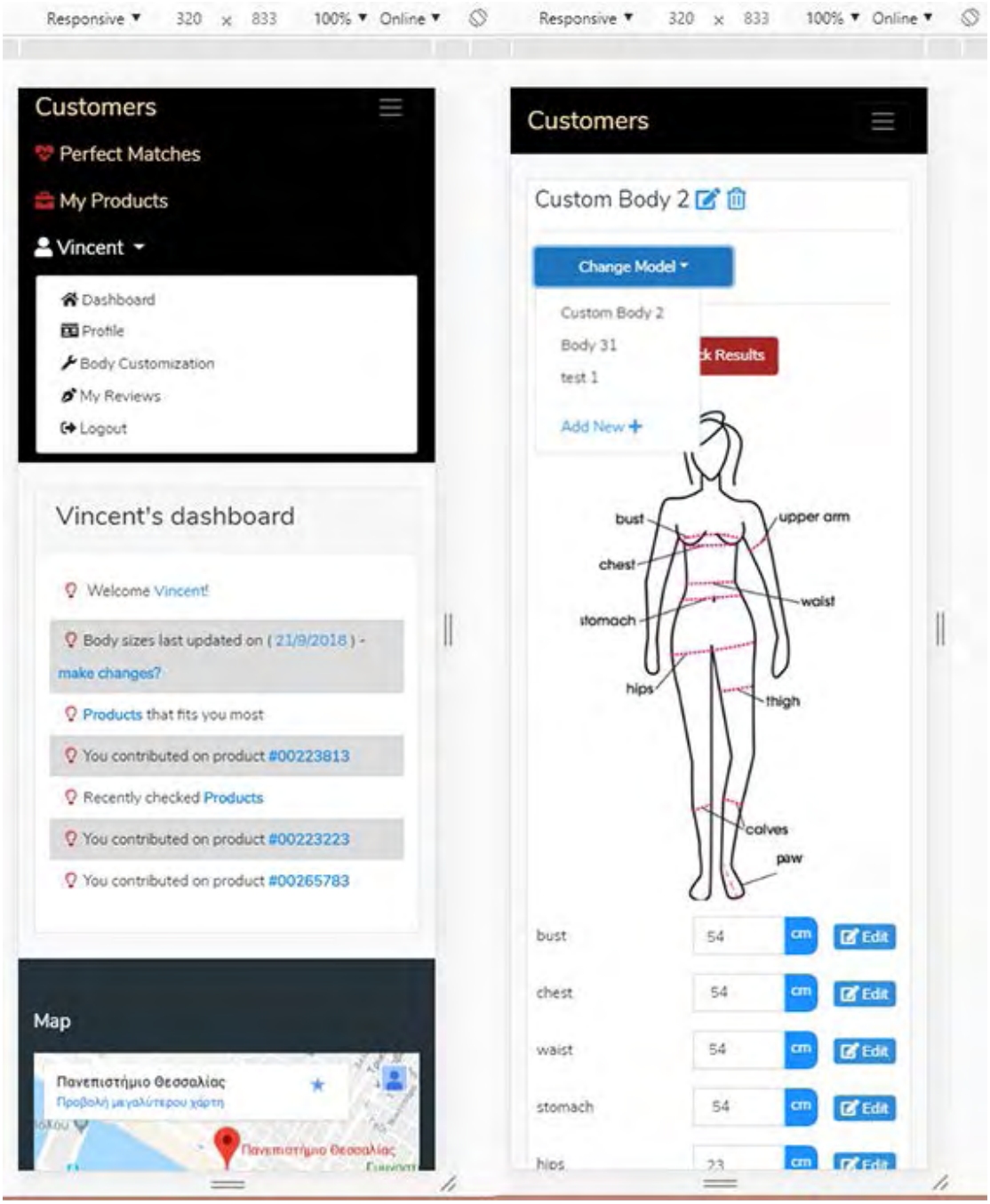
Αξίζει να αναφέρουμε επίσης πως οι σελίδες που έχουν σχεδιαστεί, λειτουργούν και προβάλλονται χωρίς προβλήματα σε όλες τις εκδόσεις των κυριότερων φυλλομετρητών (*chrome, firefox, opera, IE, safari*) από το 2017 μέχρι σήμερα.

4.2. Μέθοδοι και Τεχνικές

Οι δύο πλατφόρμες (*customers, companies*) χρησιμοποιούν το framework bootstrap v4 για την σχεδίαση της βασικής δομής τους. Παρουσιάζουν παρόμοια χαρακτηριστικά και τεχνικές σχεδίασης σε όλες τις σελίδες τους, με στόχο το UX να παραμένει ίδιο καθ' όλη την διάρκεια της περιήγησής των χρηστών στις πλατφόρμες, είτε αυτοί είναι χρήστες της πλατφόρμας Customers ή της Companies.

Ο χρήστης της πλατφόρμας customers είναι και ο ποιο απαιτητικός όσον αφορά την σχεδίαση και το UX κατά την χρήση της εφαρμογής, η οποία μπορεί να γίνεται με κάθε μέσο που έχει την δυνατότητα περιήγησης στο διαδίκτυο (H/Y, laptop, smartphone, tablets κ.α.). Αυτό έρχεται σε αντίθεση, με τον χρήστη της πλατφόρμας companies που τον ενδιαφέρει να μπορεί να επεξεργάζεται με ευκολία τις βασικές ιδιότητες των προϊόντων, να βλέπει ανά πάσα στιγμή όλες τις επιλογές και ανακοινώσεις της εφαρμογής και να αισθάνεται πως η πλατφόρμα είναι κομμάτι τις εργασίας του. Μία από τις βασικές λοιπόν διαφορές ανάμεσα στις δύο πλατφόρμες είναι το responsiveness που παρουσιάζει η πλατφόρμα customers σε όλες τις σελίδες της. Η δυνατότητα δηλαδή να προσαρμόζεται δυναμικά και άμεσα (*on the fly*) σε κάθε διάσταση και προσανατολισμό της οθόνης του χρήστη στην εικόνα . Αντίθετα, στις σελίδες επεξεργασίας, δημιουργίας ή αναζήτησης προϊόντων της πλατφόρμας

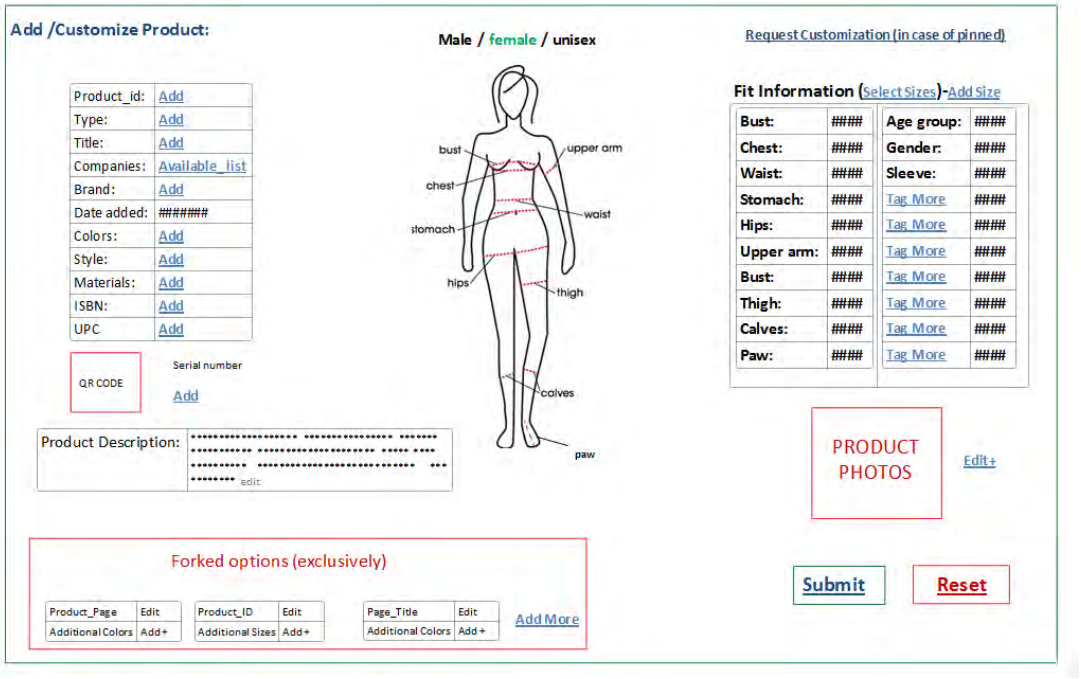
Companies, προέχει η παρουσίαση ενός γενικού και στατικού πλαισίου ικανό να παρέχει στον χρήστη όλη την πληροφορία που χρειάζεται. Στην εικόνα 4.1 φαίνεται το *responsiveness* των σελίδων dashboard και customization της πλατφόρμας customers.



Εικόνα 4.1: Η σελίδα dashboard (αριστερά) και customization (δεξιά) σε πλάτος οθόνης 320 pixels.

Σημαντικό βήμα για την σχεδίαση των πλατφόρμων είναι η δημιουργία των ανάλογων προσχεδίων για την σχεδίαση του περιεχομένου της κάθε σελίδας. Οι περισσότερες λειτουργίες εντός των πλατφόρμων είναι δυναμικές και το σχεδιαστικό

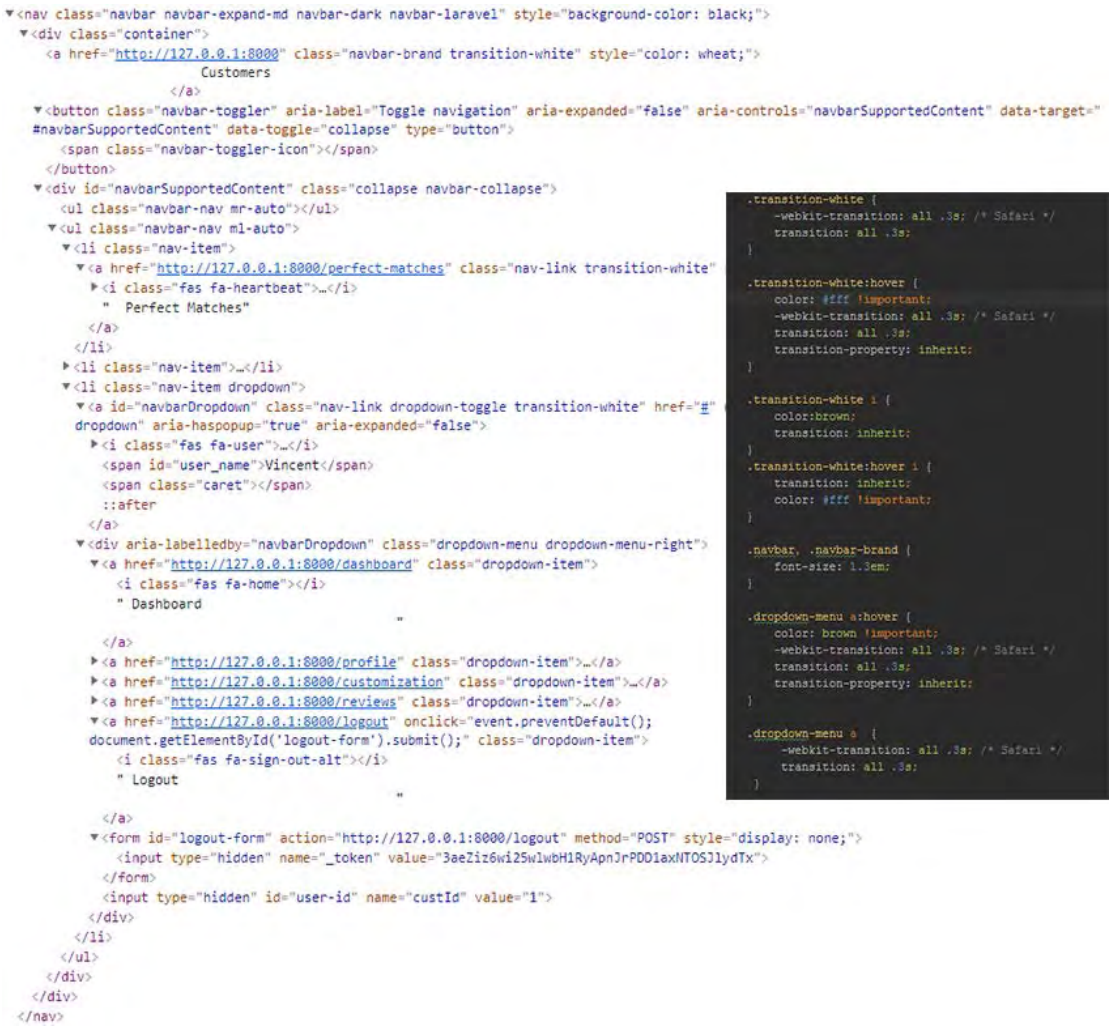
κομμάτι διαμορφώνεται και υλοποιείται με την βοήθεια της Javascript και της CSS. Για παράδειγμα, δηλώνοντας το είδος (*type*) του ρούχου, ανοίγουν αυτομάτως οι ανάλογες επιλογές μεγεθών και μετρήσεων στο πεδίο *Fit Information*. Στην εικόνα 4.2 βλέπουμε ένα προσχέδιο προσθήκης ή διαμόρφωσης ενός ήδη υπάρχοντος προϊόντος όπου διαφαίνονται πολλές από τις λειτουργίες που υποστηρίζει η πλατφόρμα με μικρότερο δυνατό πλάτος να έχει οριστεί στα 1240 pixels.



Εικόνα 4.2: Προσχέδιο διαμόρφωσης προϊόντος για την σελίδα products πλατφόρμας companies.

4.2.1. Σχεδίαση σελίδας customization

Γενικό χαρακτηριστικό της σελίδας αυτής αλλά και όλων των άλλων σελίδων είναι το γενικευμένο μενού πλοήγησης και υποσέλιδο που παρουσιάστηκε στην εικόνα 3.22. Για το μενού πλοήγησης χρησιμοποιούνται οι ιδιότητες του *navbar* του framework bootstrap [15] όπου έχει διαμορφωθεί κατάλληλα για να υποστηρίζει τις λειτουργίες των διαφόρων χρηστών της εφαρμογής. Περιέχει κλάσεις σε CSS για την δημιουργία τεχνικών όπως *transition*, *dropdown* και *hover*, καθώς και SVG αρχεία εικόνων της εταιρίας *fontawesome*. Στην εικόνα 4.3 παρουσιάζεται απόσπασμα της σχεδίασης του navbar που έχει δημιουργηθεί από τον φυλλομετρητή chrome, καθώς και μερικές από τις επιμέρους κλάσεις CSS (εκτός του bootstrap) που έχουν χρησιμοποιηθεί για την διαμόρφωσή του. Τέλος, η προβολή του μενού του κώδικα στον φυλλομετρητή παρουσιάζεται στις εικόνες 3.23 και 4.1.



Εικόνα 4.3: Απόσπασμα html του βασικού μενού της πλατφόρμας customers και κλάσεις μορφής CSS του αρχείου custom.css.

Αντίστοιχα, για την δημιουργία του υποσέλιδου (*footer*) έχει τροποποιηθεί ένα απόσπασμα (*snippet*) ελεύθερης χρήσης υποσέλιδου του χρήστη *webelance* της ιστοσελίδας "*https://bootsnipp.com*" [16]. Στην εικόνα 4.4 παρουσιάζεται απόσπασμα της σχεδίασης του footer που έχει δημιουργηθεί από τον φυλλομετρητή chrome, όπως και στην εικόνα 4.3. Η προβολή της σελίδας του κώδικα στον φυλλομετρητή παρουσιάζεται στην εικόνα 3.23.

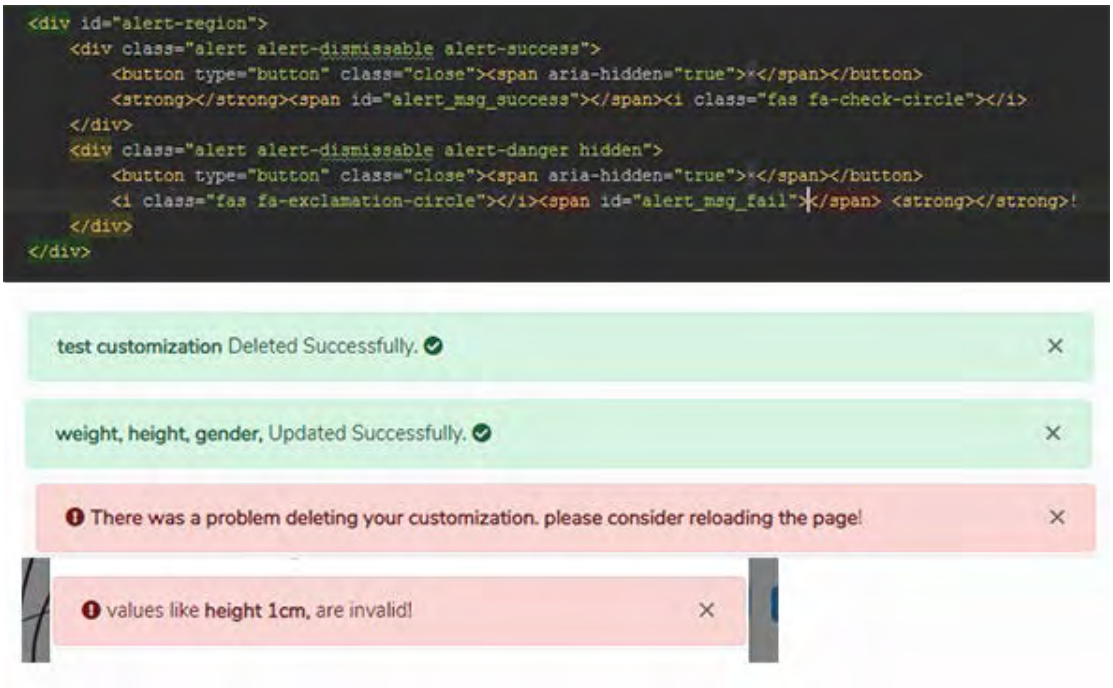
```
<footer class="footer">
    <div class="container bottom_border">
        <div class="row justify-content-center">
            <div class="col-md-3 col-sm-6">
                <h5 class="headin5_amrc col_white_amrc pt2">Map</h5>
                <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3084.936464790570 4!2d
            </div>
            <div class="col-md-3 col-sm-6">
                <h5 class="headin5_amrc col_white_amrc pt2">Contact Info</h5>
                <p><i class="fa fa-location-arrow"></i> Αργοναυτών και Φιλελλήνων, Βόλος, 382 21 </p>
                <p><i class="fa fa-phone"></i>  24210-123456  </p>
                <p><i class="fa fa fa-envelope"></i> info@example.com  </p>
            </div>
            <div class="col-md-3 col-sm-6">
                <h5 class="headin5_amrc col_white_amrc pt2">Users</h5>
                <ul class="footer_ul_amrc">
                    <li><a href="{{ route('perfect-matches') }}">Perfect Matches</a></li>
                    <li><a href={{ route('products') }}>My Products</a></li>
                    <li><a href="{{ route('dashboard') }}">Dashboard</a></li>
                    <li><a href="{{ route('reviews.index') }}">My Reviews</a></li>
                    <li><a href="{{ route('profile.index') }}">Profile</a></li>
                    <li><a href="{{ route('customization.index') }}">Body Customization</a></li>
                </ul>
            </div>
            <div class="col-md-3 col-sm-6">
                <h5 class="headin5_amrc col_white_amrc pt2">Other</h5>
                <ul class="footer_ul_amrc">...</ul>
            </div>
        </div>
    </div>
    <div class="container">
        <p class="text-center card-body">Copyright @2018 | Brewed for
            <a href="https://www.e-ce.uth.gr/" target="_blank">University Of Thessaly</a></p>
        <ul class="social_footer_ul" style="padding:0">...</ul>
    </div>
</footer>
```

```
/*footer*/
.col_white_amrc { color:#FFF;}
footer { width:100%; background-color:#263238; min-height:250px; padding:10px 0px 25px 0px ;}
.pt2 { padding-top:40px ; margin-bottom:20px ;}
footer p { font-size:13px; color:#CCC; padding-bottom:0px; margin-bottom:8px;}
.mb10 { padding-bottom:15px ;}
.footer_ul_amrc { margin:0px ; list-style-type:none ; font-size:14px; padding:0px 0px 10px 0px ; }
.footer_ul_amrc li {padding:0px 0px 5px 0px;}
.footer_ul_amrc li a{ color:#CCC;}
.footer_ul_amrc li a:hover{ color:#fff; text-decoration:none;}
.fleft { float:left;}
.padding-right { padding-right:10px; }

.footer_ul2_amrc {margin:0px; list-style-type:none; padding:0px;}
.footer_ul2_amrc li p { display:table; }
.footer_ul2_amrc li a:hover { text-decoration:none;}
.footer_ul2_amrc li i { margin-top:5px;}

.bottom_border { border-bottom:1px solid #323f45; padding-bottom:20px;}
.footer_bottom_ul_amrc {
    list-style-type:none;
    padding:0px;
    display:table;
    margin-top: 10px;
    margin-right: auto;
    margin-bottom: 10px;
    margin-left: auto;
}
.foote_bottom_ul_amrc li { display:inline;}
.foote_bottom_ul_amrc li a { color:#999; margin:0 12px;}

.social_footer_ul { display:table; margin:15px auto 0 auto; list-style-type:none;  }
.social_footer_ul li { padding-left:20px; padding-top:10px; float:left; }
.social_footer_ul li a { color:#CCC; border:1px solid #CCC; padding:8px;border-radius:50%;}
.social_footer_ul li i {  width:20px; height:20px; text-align:center;}
```

Εικόνα 4.4: Πάνω: Απόσπασμα html του βασικού υποσέλιδου της πλατφόρμας customers. Κάτω: κλάσεις υποστήριξης του υποσέλιδου της πλατφόρμας customers αρχείου custom.css.

Το κεντρικό περιεχόμενο (*content*) της σελίδας έχει διαμορφωθεί και αυτό με το bootstrap. Το εσωτερικό του έχει διαχωριστεί και κατασκευαστεί χρησιμοποιώντας το σύστημα πλέγματος δώδεκα στηλών που διαθέτει το framework (*12 col grid system*) και όλες οι αλλαγές πραγματοποιούνται σε ένα πλαίσιο (*container*) μέγιστου πλάτους 1280 pixel. Στην εικόνα 3.21 διακρίνεται ο τρόπος με τον οποίο έχει σχεδιαστεί το κεντρικό τμήμα της σελίδας.

Η σελίδα customization περιέχει χώρους ειδοποιήσεων (*alert regions*) οι οποίοι ενημερώνουν τον χρήστη για την κατάληξη της ενέργειας που έχει ή θέλει να πραγματοποιήσει. Αυτές ποικίλουν και μπορεί να είναι είτε μία απλή αλλαγή της ηλικίας του μοντέλου που παραμετροποιεί είτε μία καθολική διαγραφή αυτού, και σε συνεργασία με την *javascript* και την *Ajax* διαμορφώνεται η ανάλογη ειδοποίηση προς τον χρήστη. Στην εικόνα 4.5 παρουσιάζεται στιγμιότυπο κώδικα των ειδοποιήσεων και οι αντίστοιχες προβολές τους από τον φυλλομετρητή σε διάφορες περιοχές της σελίδας.



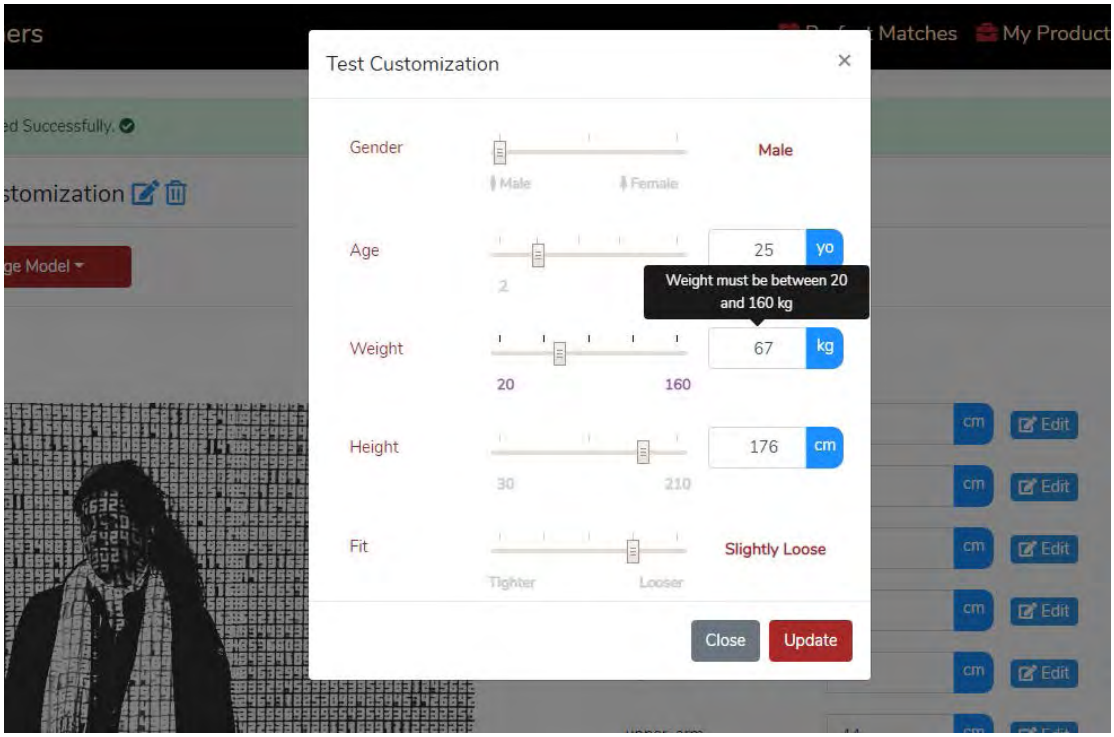
Εικόνα 4.5: Πάνω: Απόσπασμα html περιοχής ειδοποιήσεων (alert region) σελίδας customization. Κάτω: Προβολή διαμορφωμένων ειδοποιήσεων στον φυλλομετρητή.

Βασικό χαρακτηριστικό όχι μόνο της σελίδας customization αλλά και άλλων σελίδων των δύο πλατφορμών είναι η χρήση εσωτερικών βοηθητικών παραθύρων, που δημιουργούνται με τη χρήση του bootstrap και ονομάζονται modals (βοηθήματα). Τα modal φέρουν επιπλέον πληροφορίες και λειτουργίες για τον χρήστη, όπως η διαγραφή, η μετονομασία και η δημιουργία ενός νέου μοντέλου customization, η αναζήτηση χρηστών με βάση το αναγνωριστικό τους, καθώς και επιπλέον επιλογές για διαμόρφωση του ήδη υπάρχοντος μοντέλου που προβάλλεται στην σελίδα. Το βασικότερο από τα modal είναι αυτό με το αναγνωριστικό (id) *CustomizationModal*,

όπου ο χρήστης καλείται να εισάγει τα βασικά χαρακτηριστικά του μοντέλου σώματος που θέλει να διαμορφώσει ώστε να μπορέσει ο αλγόριθμος της εφαρμογής να κάνει μία πρώτη εκτίμηση των μεγεθών που αναλογούν στον χρήστη. Στην εικόνα 4.6 παρουσιάζεται ο συμπτυγμένος σε αρκετά σημεία κώδικας του αρχείου *customization-modal.blade.php* το οποίο περιέχει το εν λόγω modal και στην εικόνα 4.7 η προβολή του κώδικα από τον φυλλομετρητή στην σελίδα customization του χρήστη.

```
<!-- Modal -->
<div class="modal fade" id="CustomizationModal" tabindex="-1" role="dialog"
     aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel"></h5>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">×</span>
                </button>
            </div>
            <div class="modal-body"...>
            <div class="modal-body"...>
            <div class="modal-body">
                <div class="row justify-content-center">
                    <div class="col-3 lister">Weight</div>
                    <div class="col-5">
                        <div class="wrap">
                            <input id="range_weight" type="range" list="weight_range"  min="20"
                                   max="160"  step="1" value=""
                                   oninput="weight_value.value = range_weight.value;
                                   enable('modal_saves');"/>
                            <label for="range_weight"...>
                            <datalist id="weight_range"...>
                        </div>
                    </div>
                    <div class="wrap col-3">
                        <label for="weight_value"></label>
                        <div class="input-container">
                            <input id="weight_value" data-toggle="tooltip"
                                   data-placement="top" title="Weight must be between 20 and 160 kg"
                                   type="number" value="" min="20" max="160" style="..."
                                   class="form-control here" oninput="range_weight.value=weight_value.value;
                                   enable('modal_saves');" onkeypress="return isNumber(event)">
                            <span class="form-control icon">kg</span>
                        </div>
                    </div>
                </div>
            </div>
            <div class="modal-body"...>
            <div class="modal-body"...>
            <div class="alert alert-dismissable alert-danger"...>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                <button type="button" id="modal_saves" class="btn btn-primary"
                        disabled onclick="quick_results()">Update</button>
            </div>
        </div>
    </div>
</div>
```

Εικόνα 4.6: Στιγμιότυπο κώδικα αρχείου customization-modal.blade.php.



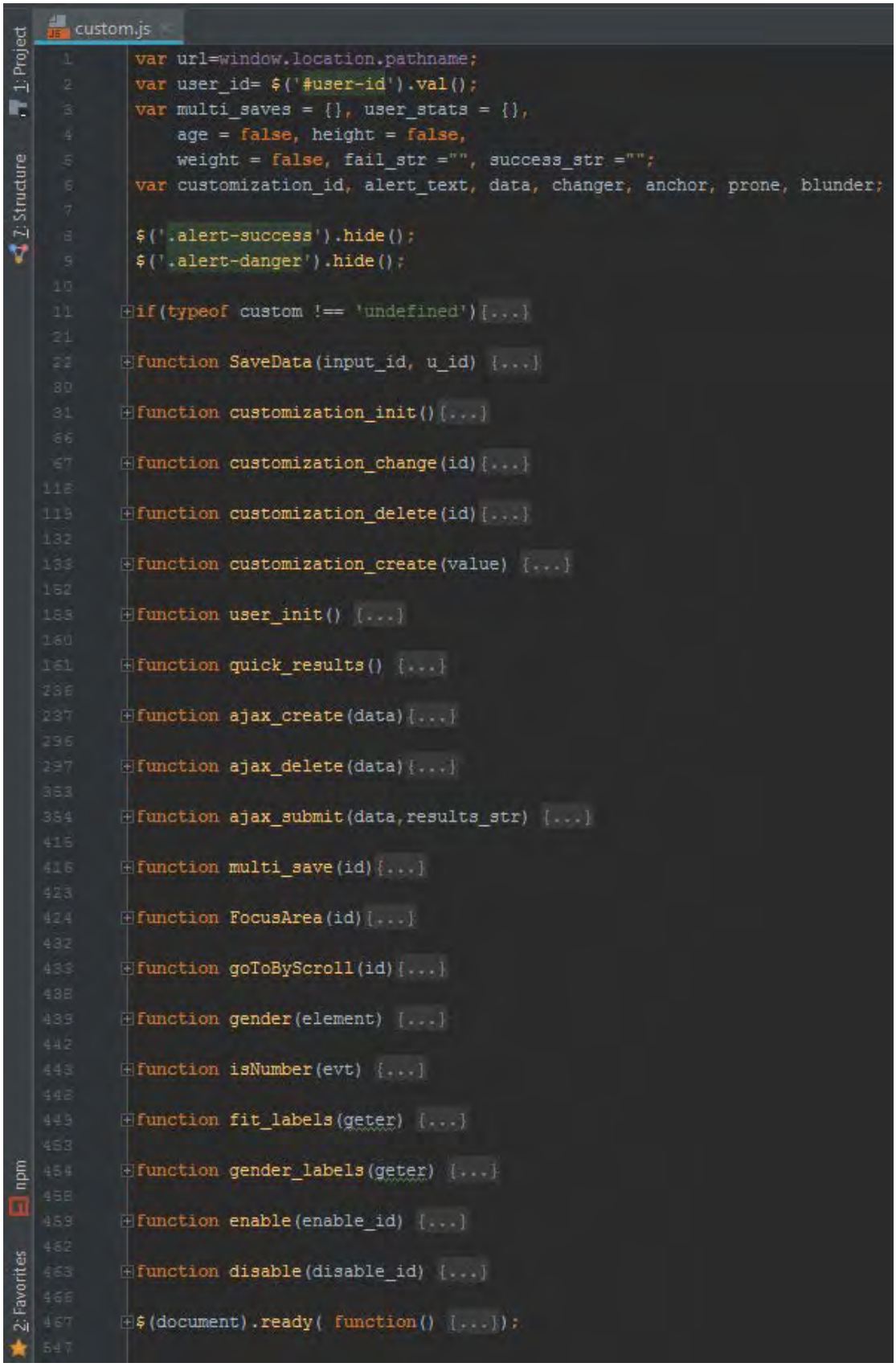
Εικόνα 4.7: Η προβολή του modal CustomizationModal στον φυλλομετρητή.

Ο σχεδιασμός και η σύνδεση των στοιχείων (*elements*) κάθε σελίδας και κυρίως αυτών που περιέχουν στο εσωτερικό τους λειτουργίες μορφής *request-reply*, πέρα από τη χρήση και υλοποίησή τους με τις γλώσσες html και css, επιτυγχάνεται την βοήθεια τεχνικών και λειτουργιών της javascript.

4.2.2. Ο ρόλος της Javascript

Σχεδόν όλα τα στοιχεία του content της σελίδας customization περιέχουν αναγνωριστικά, είναι δυναμικά και προβάλλονται στον φυλλομετρητή σύμφωνα με την λογική που έχει υλοποιηθεί στην javascript. Δεν υπάρχουν ανανεώσεις σελίδας στα *request-response* καθώς πραγματοποιούνται με τη χρήση του *Ajax*, και πολλά από τα χαρακτηριστικά των στοιχείων διαμορφώνονται μέσω της javascript καθ' όλη την διάρκεια περιήγησης του χρήστη στην σελίδα, με σκοπό να δημιουργηθεί το καλύτερο δυνατό UX ανάμεσα στον χρήστη και την πλατφόρμα.

Βασική βιβλιοθήκη η οποία απλοποιεί πολλές λειτουργίες της javascript και του DOM του φυλλομετρητή είναι η **jquery** και έχει χρησιμοποιηθεί σε όλες τις σελίδες των δύο πλατφορμών. Πέραν της jquery χρησιμοποιείται και απλή javascript για ταχύτερες ή πιο πολύπλοκες λειτουργίες ανάμεσα στα στοιχεία των σελίδων. Το βασικό αρχείο στο οποίο πραγματοποιούνται αρκετές από τις λειτουργίες της σελίδας customization είναι το *custom.js*, όπου σε συνάρτηση με αυτό της jquery και άλλων εμφωλευμένων κομματιών κώδικα javascript στα αρχεία html των σελίδων, παράγουν τις αντίστοιχες λειτουργίες σύμφωνα με τις οποίες έχουν σχεδιαστεί. Στην εικόνα 4.8 παρουσιάζονται οι συναρτήσεις και μεταβλητές του custom.js.



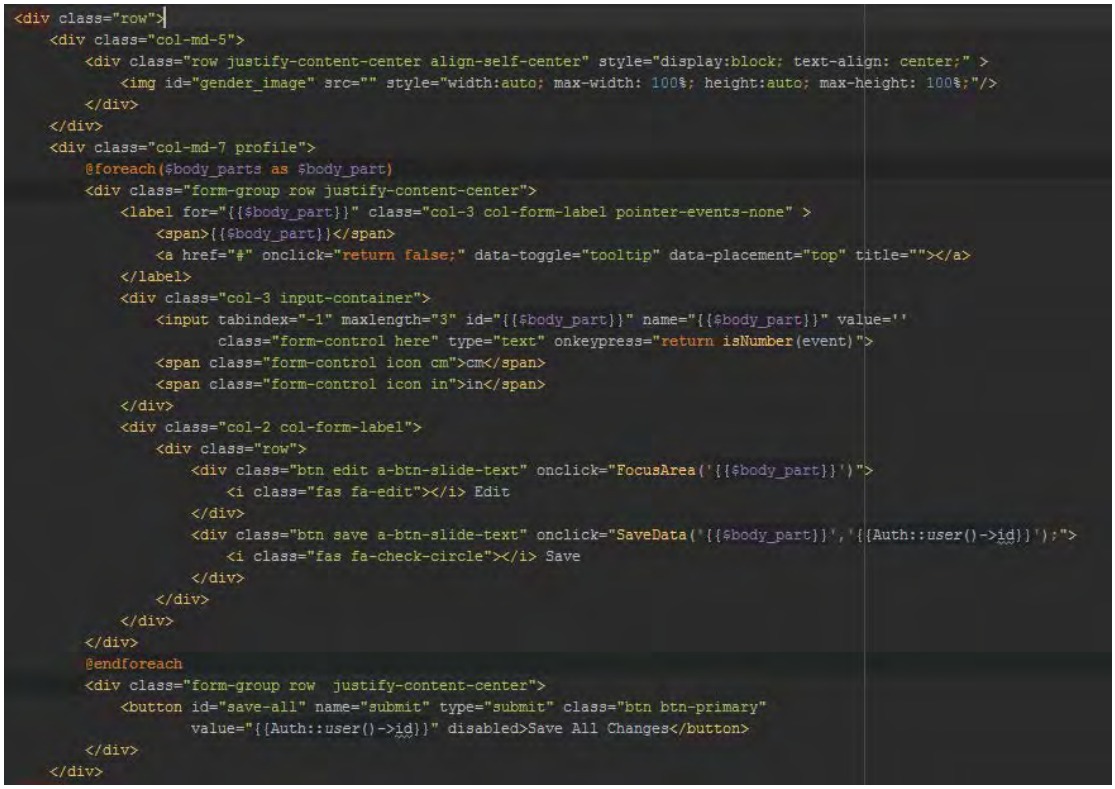
Εικόνα 4.8: Συναρτήσεις και μεταβλητές του αρχείου custom.js.

Κατά την είσοδο του χρήστη στην σελίδα customization, η javascript μέσω του αρχείου *custom.js*, εκτελεί ορισμένες βασικές λειτουργίες αρχικοποίησης των στοιχείων της σελίδας. Στην εικόνα 4.9 βλέπουμε μερικές από αυτές, όπως ο έλεγχος και η αρχικοποίηση του αρχικού μοντέλου σώματος (*default customization*) και άλλων αναγκαίων μεταβλητών.

```
if(typeof custom !== 'undefined'){
    if (custom[default_cust_id]['default'] !== 0) {
        default_cust_id = Object.keys(custom)[0]; // this may not bring the first ^-^
        // console.log('error loading default customization, default is set to first customization');
    }
    $(document).ready(customization_init,user_init);
    $('.cm').show();
    $('.in').hide();
}

function customization_init(){

    customization_id = default_cust_id;

    $('#exampleModalLabel').html(custom[customization_id]['name']+' (Default)');
    $('#customization_name').html(custom[customization_id]['name']);
    $('#gender_image').attr('src',window.location.origin+'/images/'+custom[customization_id]['gender']+'-body-parts.PNG');
    $('#default_customization_button').html(custom[customization_id]['name']);
    $('#delete_cust_request').html(custom[customization_id]['name']);
    $('#range_gender').val(custom[customization_id]['gender']);
    $('#gender_value').val(custom[customization_id]['gender']);
    $('#range_age').val(custom[customization_id]['age']);
    $('#age_value').val(custom[customization_id]['age']);
    $('#range_weight').val(custom[customization_id]['weight']);
    $('#weight_value').val(custom[customization_id]['weight']);
    $('#range_height').val(custom[customization_id]['height']);
    $('#height_value').val(custom[customization_id]['height']);
    $('#range_fit').val(custom[customization_id]['fit']);
    $('#fit_value').val(custom[customization_id]['fit']);

    body_parts.forEach(function(body_part) {
        document.getElementById(body_part).value = custom[customization_id][body_part];
    });

    fit_value.innerHTML=fit_labels(custom[customization_id]['fit']);
    gender_value.innerHTML=gender_labels(custom[customization_id]['gender']);
}

function user_init() {
    user_stats['age']= $('#range_age').val();
    user_stats['weight']= $('#range_weight').val();
    user_stats['height']= $('#range_height').val();
    user_stats['gender']= $('#range_gender').val();
    user_stats['fit']= $('#range_fit').val();
}
```

Εικόνα 4.9: Συναρτήσεις και έλεγχοι αρχικοποίησης μεταβλητών, αρχείο custom.js.

Βασικές λειτουργίες και συναρτήσεις του αρχείου *custom.js* εκτελούνται και όταν ο χρήστης ζητήσει να διαμορφώσει κάποια στοιχεία από το παρόν μοντέλο σώματός του. Για παράδειγμα, στην εικόνα 4.10 βλέπουμε τον κώδικα που διαμορφώνει τα πεδία σωματικών διαστάσεων του μοντέλου του χρήστη όπου η προβολή του στον φυλλομετρητή φαίνεται στην εικόνα 3.23.



Εικόνα 4.10: Στιγμιότυπο κώδικα αρχείου index.view.php σελίδας customization.

Ο κώδικας της εικόνα 4.10 καλεί συναρτήσεις της javascript που έχουν διαμορφωθεί στο αρχείο *custom.js*, όπως ο έλεγχος της μεταβλητής των input μέσω της *"isNumber(var)"*, η οποία επιτρέπει να εισαχθούν μόνο αριθμητικοί χαρακτήρες, καθώς και οι *"FocusArea(var)"* και *"SaveData(var,var)"* που ενεργοποιούνται όταν ο χρήστης ζητήσει να διαμορφώσει κάποιο πεδίο ή αντίστοιχα, να το αποθηκεύσει.

Να επισημάνουμε επίσης πως ο χρήστης σε πολλά σημεία της εφαρμογής μπορεί να αποθηκεύσει μεταβλητές (και άρα να ενεργοποιήσει κάποιο *request*), είτε μεμονωμένα (στοιχείο *div*, κλάση *.save*) είτε μαζικά (στοιχείο *button*, χαρακτηριστικό *save-all*). Στην εικόνα 4.11 βλέπουμε τις βασικές συναρτήσεις που χρησιμοποιούνται για να εντοπίζονται τα στοιχεία που θέλει να αποθηκεύσει ο χρήστης ώστε να διαμορφωθούν κατάλληλα τα δεδομένα που χρειάζονται για το επικείμενο request.

```
function SaveData(input_id, u_id) {
    anchor = input_id;
    user_id = u_id;
    delete multi_saves[anchor];
    if (Object.keys(multi_saves).length<=1) {
        $('#save-all').prop('disabled',true);
    }
}

function multi_save(id){
    if (Object.keys(multi_saves).length>0) {
        $('#save-all').prop('disabled',false);
        multi_saves[id]='';
    }
    multi_saves[id]='';
}

function FocusArea(id){
    var changer =  document.getElementById(id);
    var getvalue = changer.value;
    changer.focus();
    $('#'+id).val('').val(getvalue);
    $('#'+id).css('pointer-events','auto');
    multi_save(id);
}

function isNumber(evt) {
    evt = (evt) ? evt : window.event;
    var charCode = (evt.which) ? evt.which : evt.keyCode;
    return !(charCode > 31 && (charCode < 48 || charCode > 57));
}
```

```
$('.save, #save-all').click(function (e) {
    alert_text ="";
    prone=document.getElementById(anchor);
    $("#"+anchor).css('pointer-events','none');

    e.preventDefault();
    var single_key=$(prone).attr('id');

    data = {};

    if ($(this).is('#save-all')) {
        Object.keys(multi_saves).forEach(function (key) {
            multi_saves[key]=$(document.getElementById(key)).val();
            alert_text +=key+', '
        });
        data = multi_saves;

        $('.save').fadeOut(10, function() {
            window.setTimeout(function (){$('.edit').fadeIn(500); }, 10);
        });
        $('#save-all').prop('disabled',true);
        multi_saves ={};
    }
    else {
        data[single_key]= $(prone).val();
        alert_text =$('label[for='+ single_key  +']').text();
    }
    ajax_submit(data,alert_text);
});
```

Εικόνα 4.11: Πάνω: Συναρτήσεις SaveData, multi_save, FocusArea, isNumber, αρχείου custom.js. Κάτω: Χειριστής γεγονότων (event handler), κλάσης .save και χαρακτηριστικού #save-all, αρχείου custom.js.

Επίσης, πέρα από τις κλάσεις και τα αναγνωριστικά του κώδικα της εικόνας 4.9 που χρησιμοποιούνται για να ελέγξουν και να διαμορφώσουν τα δεδομένα προς αποθήκευση, η συνάρτηση *"quick_results"* καλείται όταν ο χρήστης ζητήσει την αποθήκευση των αλλαγών που έχει πραγματοποιήσει στο *CustomizationModal* της εικόνας 4.6. Η συνάρτηση αυτή, όπως και οι προηγούμενες της εικόνας 4.11, εκτελεί πολλαπλούς ελέγχους στα στοιχεία που έχει εισάγει ο χρήστης και ετοιμάζει τα δεδομένα καθώς και την ανάλογη ειδοποίηση για να πραγματοποιηθεί το request. Στην εικόνα 4.12 παρουσιάζεται η συνάρτηση «*quick_results*».

```
function quick_results() {
    blunder = false;
    data ={};
    fail_str ="";
    success_str ="";

    if ($('#age_value').val()>110 || $('#age_value').val()<2) {
        blunder = true;
        fail_str +='age '+$('#age_value').val()+'yo, ';
    }
    else {
        if ($('#age_value').val() !== user_stats['age']) {
            data['age']= parseInt($('#age_value').val());
            success_str +='age, ';
        }
    }
    if ($('#weight_value').val()>160 || $('#weight_value').val()<20) {...}
    else {...}
    if ($('#height_value').val()>210 || $('#height_value').val()<30) {...}
    else {...}
    if (blunder){
        $("#customization " + " .modal "+".alert-danger").show(500);
        $("#customization " + " .modal " + ".alert-danger " +'strong').fadeOut(100, function() {
            $("#customization " + " .modal " + ".alert-danger " +'strong').text(fail_str).fadeIn(500);
        });
    }
    else {
        if ($('#range_fit').val() !== user_stats['fit']) {
            data['fit']= parseInt($('#range_fit').val());
            success_str +='fit, ';
        }
        if ($('#range_gender').val() !== user_stats['gender']) {
            data['gender']= parseInt($('#range_gender').val());
            success_str +='gender, ';
        }
        $("#customization " + " .modal "+".alert-danger").hide(500);
        $("#customization " + " .modal " + ".alert-danger " +'strong').hide();
        $('#CustomizationModal').modal('toggle');
        disable('modal_saves');
        if(success_str===''){
            $("#alert-region "+".alert-danger").hide(500);
            $("#alert-region "+".alert-success").show(500);
            $("#alert-region "+'.alert-success '+'strong').fadeOut(500, function() {
                $(this).text('').fadeIn(500);
            });
        }
        else {
            ajax_submit(data,success_str);
        }
    }
}
```

Εικόνα 4.12: Συνάρτηση quick_results αρχείου custom.js.

Στην εικόνα 4.13 παρουσιάζεται η συνάρτηση «*ajax_submit*» που καλείται μέσω της «*quick_results*» (εικόνα 4.12) και του χειριστή γεγονότων για την κλάση *.save* και το

χαρακτηριστικό *#save-all* (εικόνα 4.11). Στην συνάρτηση αυτή, διαμορφώνονται τα δεδομένα για το request στο αντικείμενο data και ύστερα, με την χρήση του Ajax, δημιουργείται το αντίστοιχο request τύπου PUT και url κατάληξης *"/customization/user_id"*. Επομένως, όταν αυτό φτάσει στο *CustomizationController*, σύμφωνα με την εικόνα 3.18 θα περάσει στην διεργασία *update* (εικόνα 3.13) όπου και θα πραγματοποιηθούν οι ανάλογες ενέργειες από το framework της laravel.

```
function ajax_submit(data,results_str) {
    changer= data; //changer = JSON.parse(JSON.stringify(data));
    if(typeof custom !== 'undefined') {
        data['id'] = parseInt(user_id);
        data['customization_id'] = parseInt(customization_id);
    }
    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        }
    });
    $.ajax({
        type: "PUT",
        url: url+"/"+user_id,
        data:  data,
        dataType: 'json',
        success: function (data) {
            if(data.success){
                $("#alert_msg_success").html(" Updated Successfully. ");
                $("#alert-region "+".alert-danger").hide(500);
                $("#alert-region "+".alert-success").show(500);
                $("#alert-region "+'.alert-success '+'strong').fadeOut(100, function() {
                    $(this).text(results_str).fadeIn(500);
                });
                if(anchor==='name'){
                    $("#user_name ").fadeOut(100, function() {
                        $(this).text($(prone).val()).fadeIn(500);
                    });
                }
                Object.assign(custom[customization_id],changer);
            }
            else {
                $("#alert_msg_fail").html(" Problem Updating ");

                $("#alert-region "+".alert-success").hide(500, function() {
                    $("#alert-region "+".alert-danger").show(500);
                });
                $("#alert-region "+'.alert-danger '+'strong').fadeOut(100, function() {
                    $(this).text(results_str).fadeIn(500);
                });
            }
            goToByScroll('app');
            user_init();
        },
        error: function () {
            $("#alert-region "+".alert-success").hide(500, function() {
                $("#alert-region "+".alert-danger").show(500);
            });
            $("#alert-region "+'.alert-danger '+'strong').fadeOut(100, function() {
                $(this).text(results_str+' please consider reloading the page').fadeIn(500);
                results_str='';
            });
        }
    });
}
```

Εικόνα 4.13: Συνάρτηση ajax_submit αρχείου custom.js.

Τέλος, στην εικόνα 4.14 παρουσιάζονται οι συναρτήσεις *"customization_delete"* και *"customization_create"* όπου πραγματοποιούνται έλεγχοι, διαμορφώνονται τα αντικείμενα *data* και εκτελούνται οι αντίστοιχες διεργασίες *"ajax_delete"* και *"ajax_create"*.

```
function customization_delete(id){
    data ={};
    if(custom[id]['default']===0) {
        alert("Cannot delete your default body customization");
    }
    else {
        $('#DeleteModal').modal('toggle');
        data['id']=parseInt(user_id);
        data['customization_id']=parseInt(customization_id);
        ajax_delete(data);

    }
}

function customization_create(value) {
    if (Object.keys(custom).length >5) {
        alert("you have reached maximum customizations per user");
        return false;
    }
    var bat=true;
    $.each(custom, function (key) {
        if(custom[key]['name']===value) {
            alert("Customization "+value+" Already Exists");
            bat=false;
        }
    });// recursive/no returns
    if (!bat) return false;

    data ={};
    data['name']=value;
    data['user_id']=user_id;
    ajax_create(data);
}
```

Εικόνα 4.14: Συναρτήσεις ajax_delete και ajax_create αρχείου custom.js.

Κεφάλαιο 5ο
Μελλοντική Ανάπτυξη και Σχεδίαση

5.1. Μελλοντικές ενέργειες και σκέψεις

Στην εργασία αυτή, προσπάθησα να αναδείξω, παρουσιάζοντας κυρίως αποσπάσματα από το backend και frontend της σελίδας customization, τον τρόπο με τον οποίο έχουν αναπτυχθεί και σχεδιασθεί οι δύο πλατφόρμες. Η jquery έχει εφαρμοστεί στο frontend των σελίδων με σκοπό την γρήγορη υλοποίηση των βασικών λειτουργιών της εφαρμογής, καθώς κάθε άλλο frontend javascript framework θα ήταν χρονοβόρο για την εκπόνηση της σχεδίασης της εργασίας. Για να μπορέσει η εφαρμογή στο σύνολό της να είναι βιώσιμη και να αναπτυχθεί περεταίρω, θεωρώ πως κύριο και βασικό στοιχείο, είναι η μετάβαση του frontend σε ένα ολοκληρωμένο framework της javascript, καθώς οι επιλογές των χρηστών και κυρίως της πλατφόρμας companies θα μπορούν να συντηρούνται και να πραγματοποιούνται μαζικότερα και με μεγαλύτερη ευκολία.

Το κομμάτι στο οποίο έχει δοθεί η μεγαλύτερη έμφαση αλλά δεν έχει μέχρι στιγμής ολοκληρωθεί στο επιθυμητό αποτέλεσμα σχεδιαστικά, είναι οι λειτουργίες και οι διασυνδέσεις των προϊόντων με τα υπόλοιπα στοιχεία της εφαρμογής. Όπως για παράδειγμα η δημιουργία αλγορίθμου εύρεσης κατάλληλου μεγέθους με βάση τα στοιχεία που έχει ορίσει ο χρήστης. Οι διασυνδέσεις αυτές πρόκειται στο άμεσο μέλλον να υλοποιηθούν.

Μετά την ολοκλήρωσή των βασικών στοιχείων της εφαρμογής και την διασύνδεση των προϊόντων με τα στοιχεία των χρηστών, η πλατφόρμα Companies θα έχει φτάσει σε ένα επιθυμητό επίπεδο ικανό να παρουσιαστεί μία πρώτη έκδοση της, ως εφαρμογή αποθήκευσης, εύρεσης και μεταποίησης προϊόντων είδους ένδυσης.

Θεωρώ πως η εφαρμογή έχει προοπτικές καθώς πέρα από την λειτουργία της πλατφόρμας customers και του widjet, που χρειάζονται ένα μεγάλο κόστος για την σωστή κατασκευή και λειτουργία τους ώστε να μπουν δυναμικά στην αγορά, θα μπορούσε να λειτουργήσει σε πρώτη φάση η πλατφόρμα companies ως ένα μέσο παρουσίασης ειδών ένδυσης ανάμεσα στις εταιρίες.

Όπως και οι περισσότερες σύγχρονες εφαρμογές, έτσι και αυτή είναι αδύνατο να υλοποιηθεί μονομερώς από ένα και μόνο πρόσωπο καθώς όχι μόνο απαιτεί γνώσεις από μία ευρεία γκάμα τεχνικών, γλωσσών και εργαλείων προγραμματισμού αλλά παράλληλα χρειάζονται συμβουλές και εργασίες από επιμέρους επαγγέλματα όπως αυτά της βιομηχανίας ένδυσης.

6. Βιβλιογραφία

[1] S. Cullinane, B. Michael, K. Elisabeth και W. Yingli , Retail clothing returns: A review of key issues., University of Gothenburg: 10.13140/RG.2.2.35163.46882., 2017.

[2] JetBrains, «https://www.jetbrains.com,» 2009. [Ηλεκτρονικό]. Available: https://www.jetbrains.com/phpstorm/.

[3] «https://www.apachefriends.org,» Apache Friends, 22 5 2002. [Ηλεκτρονικό]. Available: https://www.apachefriends.org/index.html.

[4] A. S. Foundation, «https://httpd.apache.org/,» 1995. [Ηλεκτρονικό]. Available: https://httpd.apache.org/.

[5] Netcraft, «https://www.netcraft.com,» 17 12 2018. [Ηλεκτρονικό]. Available: https://news.netcraft.com/archives/2018/12/17/december-2018-web-server-survey.html. [Πρόσβαση 27 12 2018].

[6] npm, «https://www.npmjs.com/,» 12 1 2010. [Ηλεκτρονικό]. Available: https://www.npmjs.com/.

[7] composer, «https://getcomposer.org/,» Composer, 2012. [Ηλεκτρονικό]. Available: https://getcomposer.org/.

[8] PHP, «http://php.net/,» 1995. [Ηλεκτρονικό]. Available: http://php.net/.

[9] BuiltWith Company, «https://trends.builtwith.com,» [Ηλεκτρονικό]. Available: https://trends.builtwith.com/framework/programming-language. [Πρόσβαση 1 2019].

[10] I. Zahariev, «https://blog.famzah.net,» 9 2 2016. [Ηλεκτρονικό]. Available: https://blog.famzah.net/2016/02/09/cpp-vs-python-vs-perl-vs-php-performance-benchmark-2016/. [Πρόσβαση 28 12 2018].

[11] MariaDB, «https://mariadb.org/,» MariaDB Corporation AB, 2009. [Ηλεκτρονικό]. Available: https://mariadb.org/.

[12] NodeJs, «https://www.nodejs.org,» Node.js, 27 5 2009. [Ηλεκτρονικό]. Available: https://www.nodejs.org.

[13] «https://www.laravel.com/,» Taylor Otwell, 6 2011. [Ηλεκτρονικό]. Available: https://www.laravel.com/.

[14] jQuery, «https://jquery.com/,» 26 8 2006. [Ηλεκτρονικό]. Available: https://jquery.com/.

[15] bootstrap, «bootstrap navbar,» [Ηλεκτρονικό]. Available: https://getbootstrap.com/docs/4.0/components/navbar/.

[16] webenlance, «bootsnipp,» [Ηλεκτρονικό]. Available: https://bootsnipp.com/snippets/bxDBA.