ΤΕΙ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

Ανάπτυξη διαδικτυακού περιβάλλοντος διδασκαλίας του PHP Framework Laravel.

Πτυχιακή Εργασία

του φοιτητή Ηλία Κατσούκα

Α.Μ.: 2011123

Επιβλέπων Καθηγητής

Μπάρδης Γεώργιος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την .....

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
.....	.....	.....

Σπάρτη

Μάιος 2016

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):

……

Υπογραφή (Ολογράφως, χωρίς μονογραφή):

……

Ημερομηνία (Ημέρα – Μήνας – Έτος):

……

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Μπάρδη Γεώργιο, ο οποίος με στήριξε αλλά και με εμπιστεύθηκε δίνοντάς μου την πολύτιμη καθοδήγησή του. Η συνεργασία μας ήταν εξαιρετική καθώς μου παρείχε απρόσκοπτα τη βοήθειά του και με συμβούλευε σε κάθε βήμα μου.

Δεν θα μπορούσα βέβαια να παραλείψω τους γονείς μου που με πίστεψαν, μου παρείχαν αμέριστη στήριξη και χάρη σε αυτούς ξεκίνησα και ολοκλήρωσα τις σπουδές μου στην επιστήμη της Πληροφορικής πραγματοποιώντας έτσι τους στόχους μου.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στη φίλη μου Αγγελική, η οποία αποτέλεσε καθοριστικό παράγοντα για την περάτωση των σπουδών μου. Σημαντική ήταν επίσης και η ενθάρρυνση συγγενών και φίλων.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία σχετίζεται με τις νέες τεχνολογίες του διαδικτύου. Σκοπός της πτυχιακής εργασίας είναι η δημιουργία μιας e-learning πλατφόρμας μέσω της οποίας θα παρουσιάζονται έτοιμα software labs με τη μορφή video διαλέξεων τα οποία θα καλύπτουν τις ακόλουθες ενότητες: Routing, Views, Forms, Database, Fundraising Website, MVC & REST Theory, Controllers, Advanced Database, Miscellaneous, Registration and Login System.

Απώτερος στόχος είναι το συγκεκριμένο PHP Framework που θα διδαχθεί να είναι διαθέσιμο σε όλους τους φοιτητές του ΤΕΙ Πελοποννήσου του τμήματος Μηχανικών Πληροφορικής Τ.Ε. μέσω ενός μαθήματος που θα δημιουργηθεί στο e-class και θα έχει σκοπό την προετοιμασία των αποφοίτων της σχολής μας για την αγορά εργασίας.

Η πτυχιακή εργασία χωρίζεται σε δύο μέρη:

1. Το πρώτο μέρος περιέχει την ανάλυση του Learning Management System που χρησιμοποιήθηκε καθώς και την παρουσίαση και ανάλυση του PHP framework Laravel.
2. Το δεύτερο μέρος της είναι η παρουσίαση του υφιστάμενου μαθήματος «Εισαγωγή στο Laravel».

Το πρώτο μέρος αποτελείται από το πρώτο κεφάλαιο που αποτελεί μια γενική εισαγωγή, το δεύτερο κεφάλαιο στο οποίο γίνεται μια αναφορά στα LMS συστήματα και αναλύεται το Moodle. Το τρίτο κεφάλαιο περιγράφει το PHP Framework Laravel, τα χαρακτηριστικά του, σε τι διαφέρει από άλλα PHP Frameworks.

Το δεύτερο μέρος της πτυχιακής περιέχει το τέταρτο κεφάλαιο στο οποίο αναλύεται το μάθημα «Εισαγωγή στο Laravel». Το κεφάλαιο αυτό περιέχει τις προσωπικές μου σημειώσεις που δημιούργησα για το σκοπό υλοποίησης του μαθήματος, αλλά και ένα παράρτημα με τον κώδικα υλοποίησης παραδειγμάτων και δύο projects.

Λέξεις – κλειδιά: PHP Framework Laravel, LMS Moodle, MVC, Νέες τεχνολογίες διαδικτύου, Εισαγωγή στο Laravel.

ABSTRACT

The particular thesis is relevant to the contemporary web technologies. The aim of the thesis is the creation of an e-learning platform through which software labs will be presented in the form of video lectures and the purpose of them will be to cover the following units: Routing, Views, Forms, Database, Fundraising website, MVC & REST Theory, Controllers, Advanced Database, Miscellaneous, Registration and Login System.

The overall objective is the accessibility of the particular PHP Framework that will be taught to all undergraduate students of Technological Educational Institute of Peloponnese (Department of Computer Engineering). This will be achieved through a specifically designed lesson in e-class, the aim of which will be to prepare graduate students for the market.

Thesis is divided into the following two parts:

1. The first part is composed of the LMS system analysis that was used, the presentation and the analysis of the PHP Framework Laravel.
2. The second part is the actual presentation of the seminar called “A Very Complete Introduction to Laravel”.

The first part includes the first unit which is in fact a general introduction, the second unit in which there is a reference to LMS and an analysis of Moodle. The third unit contains the description of the PHP Framework Laravel, its features and its differences from other PHP Frameworks.

The second part of thesis includes the fourth unit in which there is the analysis of the whole seminar “A Very Complete Introduction to Laravel”. This unit consists of my personal notes that I created with the aim of implementation of this seminar and an appendix with the PHP code and two projects.

Key – words: PHP Framework Laravel, LMS Moodle, MVC, Contemporary web technologies, A Very Complete Introduction to Laravel.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1

ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο της πτυχιακής

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η δημιουργία μιας e-learning πλατφόρμας μέσω της οποίας θα παρουσιάζονται σε μορφή video διαλέξεων έτοιμα software labs τα οποία θα καλύπτουν ένα ευρύ φάσμα του PHP Framework Laravel. Οι ενότητες που θα υλοποιηθούν και θα αναλυθούν είναι οι ακόλουθες: Routing, Views, Forms, Database, Controllers, MVC and REST, Advanced Database, Miscellaneous, Project 1: Website με δωρεές, Project 2: Πλατφόρμα με Εγγραφή χρήστη και Είσοδο χρήστη.

Η πτυχιακή εργασία έχει ως απώτερο σκοπό το συγκεκριμένο PHP Framework να είναι διαθέσιμο σε όλους τους φοιτητές του ΤΕΙ Πελοποννήσου του τμήματος Μηχανικών Πληροφορικής Τ.Ε. μέσω ενός μαθήματος που θα δημιουργηθεί στο e-class για την προετοιμασία των αποφοίτων της σχολής μας για την αγορά εργασίας.

1.2 Ανάλυση κεφαλαίων

Η παρούσα πτυχιακή εργασία αποτελείται από τέσσερα κεφάλαια και ένα παράρτημα. Στο πρώτο κεφάλαιο παρουσιάζεται το αντικείμενο της πτυχιακής.

Στη συνέχεια στο δεύτερο κεφάλαιο παρουσιάζεται ο όρος της τηλεκπαίδευσης, οι μορφές της (σύγχρονη και ασύγχρονη τηλεκπαίδευση), γίνεται μια παρουσία των δωρεάν και εμπορικών Συστημάτων Διαχείρισης Μάθησης και καταλήγουμε σε μια λεπτομερείς ανάλυση του Συστήματος Moodle.

Το επόμενο κεφάλαιο είναι το τρίτο στο οποίο παρουσιάζονται η γλώσσα προγραμματισμού PHP και το PHP Framework Laravel με το οποίο θα ασχοληθούμε.

Το τέταρτο και τελευταίο κεφάλαιο της πτυχιακής εργασίας περιλαμβάνει σε θεωρητικό επίπεδο σημειώσεις του μαθήματος «Εισαγωγή στο Laravel». Οι σημειώσεις καλύπτουν σε ένα αξιόλογο επίπεδο την εκμάθηση του Framework.

Εκτός από τα τέσσερα κεφάλαια υπάρχει και ένα παράρτημα που περιέχει τον κώδικα των παραδειγμάτων. Τα παραδείγματα δημιουργήθηκαν με σκοπό την αρτιότερη κατανόηση των δυνατοτήτων του Laravel Framework.

2

ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΜΑΘΗΣΗΣ

2.1 Τηλεκπαίδευση

Η τηλεκπαίδευση είναι μια μορφή εκπαίδευσης από απόσταση στην οποία ο εκπαιδευτής και ο εκπαιδευόμενος δεν απαιτείται να βρίσκονται στον ίδιο χώρο, τον ίδιο χρόνο. Η επικοινωνία μεταξύ εκπαιδευτή και εκπαιδευόμενου γίνεται μέσω μιας αμφίδρομης σύγχρονης ή ασύγχρονης επικοινωνίας. Το περιεχόμενο της διδασκαλίας είναι προσαρμοσμένο στην εξέλιξη της τεχνολογίας και περιλαμβάνει αρχεία video, ήχου ή/και εικόνας, έντυπα σε ηλεκτρονική μορφή.

2.2 Μορφές Τηλεκπαίδευσης

Οι μορφές της τηλεκπαίδευσης διακρίνονται ανάλογα με τον τρόπο της επικοινωνίας μεταξύ εκπαιδευτή και εκπαιδευόμενου. Έτσι έχουμε τη σύγχρονη και την ασύγχρονη εκπαίδευση.

2.2.1 Σύγχρονη Τηλεκπαίδευση

Η σύγχρονη τηλεκπαίδευση έχει πολλές ομοιότητες με την παραδοσιακή εκπαίδευση, διότι η διδασκαλία πραγματοποιείται σε πραγματικό χρόνο με τη χρήση της τηλεδιάσκεψης και έχουμε την ταυτόχρονη συμμετοχή εκπαιδευτή και εκπαιδευομένων. Ωστόσο υπάρχει η δυνατότητα ο καθηγητής και ο μαθητής να βρίσκονται σε διαφορετικούς χώρους εξαλείφοντας έτσι τους γεωγραφικούς περιορισμούς. Το μάθημα πραγματοποιείται σε μια εικονική αίθουσα διδασκαλίας με δυνατότητες ανταλλαγής αρχείων, παράλληλης παρουσίασης εκπαιδευτικού υλικού,

χρήσης ηλεκτρονικού πίνακα, διαμοιρασμού της επιφάνειας εργασίας του εκπαιδευτή (desktop share) αλλά και χρήσης κάμερας και μικροφώνου.

Παρακάτω παρουσιάζονται κάποια από τα πλεονεκτήματα της σύγχρονης τηλεκπαίδευσης:

- Εκπαιδευτής και εκπαιδευόμενος μπορούν να έχουν συνομιλούν σε πραγματικό χρόνο.
- Υπάρχει δυνατότητα καταγραφής και επανάληψης του μαθήματος.
- Μπορούν να χρησιμοποιηθούν διάφορα προγράμματα προσομοίωσης με σκοπό την πραγματοποίηση εικονικών εργαστηρίων.
- Μέσω συνομιλίας φωνής (voice chat) δύο ή περισσότεροι μπορούν να επικοινωνήσουν μέσω συνδιάσκεψης φωνής (audioconference) σε πραγματικό χρόνο.

Εκτός από πλεονεκτήματα η σύγχρονη τηλεκπαίδευση έχει και μειονεκτήματα όπως:

- Οι εκπαιδευόμενοι είναι αποξενωμένοι.
- Ο εκπαιδευτής απαιτείται να αφιερώσει πολύ χρόνο στην προετοιμασία του μαθήματος και τη δημιουργία του ψηφιακού υλικού.
- Είναι απαραίτητο ο εκπαιδευτής και οι εκπαιδευόμενοι να είναι εξοικειωμένοι με την τεχνολογία.
- Για την καλύτερη διεξαγωγή του μαθήματος απαιτείται δίκτυο υψηλού εύρους ζώνης.

2.2.2 Ασύγχρονη Τηλεκπαίδευση

Στην ασύγχρονη τηλεκπαίδευση ο εκπαιδευόμενος παρακολουθεί τις διαλέξεις του εκπαιδευτή σε διαφορετικό χώρο αλλά και σε διαφορετικό χρόνο από αυτόν. Στη μορφή της ασύγχρονης εκπαίδευσης ο μαθητής έχει τη δυνατότητα να επικοινωνήσει ασύγχρονα με τον εκπαιδευτή και τους υπόλοιπους εκπαιδευόμενους, ακόμη μπορεί να χρησιμοποιήσει το υλικό, που έχει από την κάθε διάλεξη, οποιαδήποτε στιγμή και σε οποιοδήποτε μέρος επιθυμεί. Η ασύγχρονη τηλεκπαίδευση λειτουργεί χρησιμοποιώντας κατάλληλα εργαλεία με τα οποία αναπτύσσονται ηλεκτρονικά μαθήματα. Αυτά τα εργαλεία ονομάζονται «Συστήματα Διαχείρισης Μάθησης» (Learning Management Systems – LMS).

Τα Συστήματα Διαχείρισης Μάθησης παρέχουν διάφορες λειτουργικότητες όπως είναι η δημιουργία ρόλων μαθητής – καθηγητής, δημιουργία μαθημάτων, η ανάθεση εργασιών από την πλευρά του εκπαιδευτή, η προσθήκη εκπαιδευτικού υλικού, κλπ. Ο τρόπος λειτουργίας των Συστημάτων Διαχείρισης Μάθησης είναι απλός. Αρχικά ο καθηγητής δημιουργεί το μάθημα, στη συνέχεια προσθέτει το κατάλληλο εκπαιδευτικό υλικό, οι εκπαιδευόμενοι επισκέπτονται το μάθημα και το παρακολουθούν.

Ορισμένα από τα πλεονεκτήματα της ασύγχρονης τηλεκπαίδευσης αποτελούν τα παρακάτω:

- Υπάρχει ευελιξία στο χρόνο και το χώρο μάθησης.
- Ο ρυθμός μεταφοράς της γνώσης από τον εκπαιδευτικό στον εκπαιδευόμενο αυξάνεται.
- Καταργούνται τα γεωγραφικά σύνορα.
- Ο εκπαιδευόμενος έχει πρόσβαση στο εκπαιδευτικό υλικό οποιαδήποτε στιγμή θέλει.
- Ο εκπαιδευτής μπορεί να εμπλουτίσει το εκπαιδευτικό υλικό χρησιμοποιώντας τις νέες τεχνολογίες.

Ωστόσο εκτός από πλεονεκτήματα η ασύγχρονη τηλεκπαίδευση έχει και μειονεκτήματα, κάποια από τα οποία είναι τα παρακάτω:

- Απουσιάζει η ζωντανή επικοινωνία και η άμεση υποβολή ερωτήσεων προς τον καθηγητή με αποτέλεσμα οι εκπαιδευόμενοι νιώθουν απομονωμένοι.
- Ο καθηγητής πρέπει να αφιερώσει επιπλέον χρόνο στην προετοιμασία και συντήρηση του εκπαιδευτικού υλικού.
- Η επικοινωνία του εκπαιδευτή με τον εκπαιδευόμενο είναι απρόσωπη.
- Το κόστος για την αγορά και συντήρηση του απαραίτητου εξοπλισμού είναι μεγάλο.

2.3 Εισαγωγή στα Συστήματα Διαχείρισης Μάθησης

Στις μέρες μας λόγω της αυξημένης χρήσης του διαδικτύου παρατηρείται ότι υπάρχουν όλο και περισσότερα Συστήματα Διαχείρισης Μάθησης (Learning Management Systems) τα οποία προσφέρουν μια σειρά από εργαλεία και υπηρεσίες για την υλοποίηση της σύγχρονης αλλά και ασύγχρονης εκπαίδευσης.

Με τον όρο Συστήματα Διαχείρισης Μάθησης (Learning Management Systems - LMS), εννοούμε τα συστήματα (πλατφόρμες λογισμικού) που έχουν ως κύριο σκοπό τη διανομή και τη διαχείριση όλων των μαθησιακών αναγκών. Σε αυτό το σημείο θα ήθελα να επισημάνω, ότι με τον όρο «διαχείριση» αναφερόμαστε στην πληροφορία που συμβάλλει στη μάθηση και όχι μεμονωμένα στη μάθηση.

Πιο αναλυτικά, μέσω των Συστημάτων Διαχείρισης Μάθησης έχουμε τις εξής δυνατότητες:

- Η εγγραφή των νέων χρηστών – μαθητών είναι αυτοματοποιημένη και ελέγχεται η είσοδός τους στα μαθήματα.

- Ο εκπαιδευτικός έχει τη δυνατότητα δημιουργίας test, ερωτηματολογίων αλλά και ολοκληρωμένης εξέτασης.
- Ο εκπαιδευτικός μπορεί να διαχειριστεί εύκολα το εκπαιδευτικό υλικό. Για παράδειγμα μπορεί να δημιουργήσει μαθήματα χρησιμοποιώντας έτοιμα εργαλεία που περιλαμβάνονται στην πλατφόρμα, να εισάγει έτοιμα μαθήματα, να εμπλουτίσει τα ήδη υπάρχοντα ακόμη και να τα διαγράψει.
- Ενημέρωση του εκπαιδευτικού και του εκπαιδευόμενου με την ολοκλήρωση του μαθήματος.
- Ο εκπαιδευτικός έχει τη δυνατότητα να διαχειριστεί την τάξη χρησιμοποιώντας εργαλεία για επικοινωνία και παρακολούθηση.
- Για κάθε εκπαιδευόμενο ξεχωριστά ορίζεται μια πορεία εκμάθησης με σκοπό να καλυφθούν τυχόν υπάρχοντα κενά.
- Υπάρχει η δυνατότητα παρακολούθησης των ενεργειών των χρηστών από την είσοδό τους μέχρι την έξοδο τους από την πλατφόρμα. Τα δεδομένα που έχουν καταγραφεί είναι διαθέσιμα στον διαχειριστή της πλατφόρμας και στον εκπαιδευτικό. Κάποια από τα δεδομένα αυτά είναι: η εγγραφή και η συμμετοχή στα μαθήματα, οι βαθμοί των εργασιών, τα αποτελέσματα test, quiz, ή διαγωνισμάτων, η συμμετοχή σε forum.

2.4 Πλατφόρμες Συστημάτων Διαχείρισης Μάθησης

Τα Συστήματα Διαχείρισης Μάθησης (LMS) διακρίνονται σε δυο κατηγορίες. Μπορεί να είναι είτε εμπορικά, είτε ανοικτού κώδικα (open source). Υπάρχουν πολλές εμπορικές και ανοικτού κώδικα πλατφόρμες, όμως λαμβάνοντας υπόψιν το κόστος λειτουργίας και συντήρησης που έχουν οι εμπορικές πλατφόρμες αποφασίσαμε να χρησιμοποιήσουμε μια πλατφόρμα ανοικτού κώδικα.

Ακόμη, οι πλατφόρμες ανοικτού κώδικα είναι ταχύτερα εξελισσόμενες διότι δουλεύουν πολλά άτομα στην ανάπτυξή και τη

συντήρησή τους. Δεν υπάρχουν σοβαρά θέματα ασφαλείας, διότι ο κώδικας είναι διαθέσιμος ελεύθερα και οι νέες εκδόσεις του λογισμικού είναι διαθέσιμες στους χρήστες ανά τακτά χρονικά διαστήματα.

2.4.1 Δωρεάν ή Ανοικτού κώδικα πλατφόρμες

Παρακάτω θα δούμε κάποιες από τις δωρεάν ή ανοικτού κώδικα πλατφόρμες LMS.

- Open eClass: Η πλατφόρμα Open eClass (http://www.openeclass.org/) είναι ένα ολοκληρωμένο Σύστημα Διαχείρισης Μαθημάτων. Σχεδιάζεται, αναπτύσσεται και υποστηρίζεται από το Ελληνικό Ακαδημαϊκό Διαδίκτυο (GUnet) και αποτελεί ελεύθερο λογισμικό ανοιχτού κώδικα. Ξεκίνησε να χρησιμοποιείται το 2003 στα Ακαδημαϊκά Ιδρύματα της Ελλάδας, ενώ παράλληλα σήμερα χρησιμοποιείται από την πρωτοβάθμια και τη δευτεροβάθμια εκπαίδευση.
- Moodle: Το Moodle (https://moodle.org/) είναι ένα Σύστημα Διαχείρισης Μάθησης που δημιουργήθηκε το 1999 από τον Martin Dougiamas και παρέχεται δωρεάν ως λογισμικό ανοικτού κώδικα. Όσον αφορά τα τεχνικά του χαρακτηριστικά έχει αναπτυχθεί με την γλώσσα προγραμματισμού PHP και για Βάση Δεδομένων χρησιμοποιεί τη MySQL.

2.4.2 Εμπορικές πλατφόρμες

Παρακάτω θα δούμε την εμπορική πλατφόρμα Blackboard LMS. Το Blackboard (http://uki.blackboard.com/) είναι ένα ολοκληρωμένο Σύστημα Διαχείρισης Μάθησης το οποίο έχει αναπτυχθεί για εκπαιδευτικά ιδρύματα και προσφέρει τριπλή λειτουργία μέσω του διαδικτύου, όπως είναι: η διδασκαλία του εκπαιδευτικού υλικού, η

επικοινωνία μεταξύ εκπαιδευτή και εκπαιδευόμενου και τέλος η αξιολόγηση των εκπαιδευομένων. Είναι μια ευέλικτη πλατφόρμα την οποία μπορούν εύκολα οι εκπαιδευτές να προσαρμόσουν ανάλογα με την θεωρία μάθησης ή το μοντέλο διδασκαλίας που θέλουν να χρησιμοποιήσουν. Η αρχιτεκτονική του Blackboard επιτρέπει να ενσωματωθούν κι άλλες εφαρμογές.

2.5 Τι είναι το Moodle

Το όνομα του Moodle είναι το ακρωνύμιο του Modular Object Oriented Developmental Learning Environment. Είναι ένα λογισμικό ανοικτού κώδικα (κάτω από την GNU Public Licence) μέσω του οποίου δημιουργούμε δυναμικά και ευέλικτα online μαθήματα. Εκτός όμως από LMS, συχνά αναφέρεται και είτε ως CMS (Course Management System), είτε ως VLE (Virtual Learning Environment) και προσφέρει διάφορες υπηρεσίες ασύγχρονης τηλεκπαίδευσης.

Όπως είδαμε το όνομα του Moodle προέρχεται από διάφορες λέξεις, τις οποίες θα εξηγήσουμε παρακάτω.

Η λέξη **Modular** χρησιμοποιείται επειδή το περιβάλλον της πλατφόρμας αποτελείται από αρθρώματα (quiz, forum, ηλεκτρονικό ταχυδρομείο, κλπ.) που πραγματοποιούν συγκεκριμένες λειτουργίες. Τα μέλη της κοινότητας του Moodle δημιουργούν ολοένα και περισσότερα αρθρώματα τα οποία προσφέρουν στο ευρύ κοινό.

Ο όρος **Object Oriented** χρησιμοποιείται επειδή το περιβάλλον του λογισμικού είναι αντικειμενοστραφές, δηλαδή οι χρήστες ασκούν ενέργειες σε αντικείμενα του περιβάλλοντος. Έτσι, το σύστημα μπορεί να χρησιμοποιηθεί με πολύ εύκολο τρόπο.

Με τη λέξη **Dynamic** ορίζεται ότι το περιβάλλον του Moodle είναι δυναμικό και συνεχώς εξελισσόμενο. Είναι δυνατή η είσοδος και η αποθήκευση των στοιχείων του χρήστη όπως είναι το προφίλ, οι βαθμοί,

τα μαθήματα, κλπ. Μέσω του περιβάλλοντος αυτού παρουσιάζονται διαφορετικά στοιχεία για κάθε χρήστη λόγω της ύπαρξης μιας μεγάλης Βάσης Δεδομένων. Συμπεραίνεται ότι οι ιστοσελίδες δεν είναι στατικές αλλά δυναμικές και είναι προσαρμοσμένες σε κάθε χρήστη.

2.6 Ιστορία του Moodle

Η πλατφόρμα ασύγχρονης τηλεκπαίδευσης Moodle αναπτύχθηκε το 1999 από τον Αυστραλό Martin Dougiamas, στα πλαίσια της διδακτορικής του διατριβής, ο οποίος είχε ως στόχο να βοηθήσει τους εκπαιδευτικούς να δημιουργήσουν online μαθήματα αποσκοπώντας έτσι στην αλληλεπίδραση καθώς επίσης και την συνεργατική κατασκευή περιεχομένου.

Η πρώτη έκδοση του Moodle κυκλοφόρησε στις 20 Αυγούστου 2002 και η τελευταία του έκδοση είναι η 3.0.3 και κυκλοφόρησε στις 14 Μαρτίου 2016.

Η πλατφόρμα προσφέρεται δωρεάν ως λογισμικό ανοικτού κώδικα βασισμένο στη Γενική Άδεια Δημόσιας Χρήσης GNU (GNU Public License). Αυτό συνεπάγεται ότι έχουμε τη δυνατότητα λήψης (download) του κώδικα δωρεάν από το διαδίκτυο, χρήσης του καθώς επίσης μπορούμε να προτείνουμε τυχόν διορθώσεις σε αυτόν.

Σε αυτό το σημείο μπορούμε να αναφέρουμε και κάποια στατιστικά στοιχεία για το Moodle τα οποία είναι τα εξής:

- Έχει 74.117 ενεργές εγκαταστάσεις σε websites.
- Είναι εγκατεστημένο σε 228 χώρες.
- Έχουν δημιουργηθεί 9.565.924 μαθήματα.
- Έχει 86.040.330 ενεργούς χρήστες.
- Έχει 252.956.099 εγγραφές.
- Είναι διαθέσιμο σε περισσότερες από 78 γλώσσες.

2.7 Χαρακτηριστικά του Moodle

Σε αυτή την ενότητα θα μελετήσουμε ορισμένα από τα χαρακτηριστικά που έχει η πλατφόρμα Moodle. Τα χαρακτηριστικά της πλατφόρμας είναι τα εξής:

- Το Moodle είναι ευρέως γνωστό. Το χρησιμοποιούν διάφορα Πανεπιστήμια ανά τον κόσμο όπως είναι το Πανεπιστήμιο Yale (http://www.yale.edu/) και το MIT (http://web.mit.edu/) της Μασαχουσέτης. Στην Ελλάδα το Moodle έχει περισσότερες από 45 ενεργές εγκαταστάσεις σε φορείς εκπαίδευσης, όπως είναι το Εθνικό Μετσόβιο Πολυτεχνείο (http://www.ntua.gr/), το Ελληνικό Ανοιχτό Πανεπιστήμιο (http://www.eap.gr/el/) και το Πανελλήνιο Σχολικό Δίκτυο (http://www.sch.gr/).
- Η πλατφόρμα Moodle χορηγείται δωρεάν σαν ελεύθερο λογισμικό κάτω από την Γενική Άδεια Δημόσιας Χρήσης (GPL GNU). Ο κώδικας υπάρχει διαθέσιμος στο Διαδίκτυο και μπορεί ελεύθερα να χρησιμοποιηθεί, να διορθωθεί και να προστεθούν νέα τμήματα κώδικα στον ήδη υπάρχον.
- Το Moodle διαθέτει μια πολύ ενεργή κοινότητα χρηστών (community) στην οποία υπάρχει μια ομάδα που ασχολείται με τη διόρθωση σφαλμάτων στον κώδικα, την κατασκευή νέων εργαλείων, την επίλυση προβλημάτων και αποριών που αναφέρονται σε ομάδες συζητήσεων (forum). Οι χρήστες του Moodle χρησιμοποιούν τα νέα χαρακτηριστικά και δίνουν feedback στους προγραμματιστές της πλατφόρμας με αποτέλεσμα τη μεταξύ τους συνεργασία.
- Το LMS Moodle είναι επικεντρωμένο στο πόσο αποτελεσματική είναι η εκπαίδευση (learning centered) και είναι βασισμένο σε διάφορες παιδαγωγικές αρχές, σε αντίθεση με άλλες εμπορικές πλατφόρμες LMS οι οποίες στηρίζονται στα εργαλεία που διαθέτουν (tool centered).

- Οι εκπαιδευόμενοι μπορούν να υποβάλλουν την εργασία τους ηλεκτρονικά και έχουν τη δυνατότητα καθορισμού της προθεσμίας υποβολής.
- Το εκπαιδευτικό υλικό μπορεί να οργανωθεί ανά εβδομάδα ή ανά θεματική ενότητα.
- Δίνει τη δυνατότητα αξιολόγησης μέσω κατάλληλων φορμών που έχουν αναπτύξει οι εκπαιδευτικοί. Η βαθμολόγηση των διαγωνισμάτων γίνεται αυτοματοποιημένα και ενημερώνεται άμεσα ο εκπαιδευόμενος.
- Οι εκπαιδευόμενοι εγγράφονται αυτόματα, έπειτα δημιουργούν το προσωπικό τους προφίλ και εγγράφονται στα μαθήματα που επιθυμούν.

2.8 Λειτουργίες του Moodle

Το Moodle έχει διάφορες λειτουργίες που σχετίζονται τόσο με τους εκπαιδευτικούς όσο και με τους εκπαιδευόμενους. Παρακάτω αναλύονται ορισμένες από αυτές:

- **Απουσιολόγια**: Ο εκπαιδευόμενος μπορεί να δει τις απουσίες του, οι οποίες προκύπτουν με βάση τη δραστηριότητα ή τη συμμετοχή του σε ένα μάθημα. Η καταχώριση των απουσιών γίνεται με δύο τρόπους. Ο πρώτος τρόπος είναι να τις καταχωρίσει ο εκπαιδευτικός και ο δεύτερος είναι να καταχωρηθούν αυτόματα μέσα σε 24 ώρες μέσω των αρχείων καταγραφής.
- **Απορίες**: Όλοι οι εκπαιδευόμενοι μπορούν να εκφράσουν μια απορία δίνοντας τίτλο, περιγραφή, λέξη-κλειδί και λαμβάνουν την απάντησή της είτε από κάποιες έτοιμες απαντήσεις που υπάρχουν, είτε λαμβάνουν μια προσωπική απάντηση από τον καθηγητή.
- **Ασκήσεις**: Ο εκπαιδευτικός αναθέτει την άσκηση στον εκπαιδευόμενο και του δίνει τη δυνατότητα να αξιολογήσει την άσκηση πριν την υποβάλλει. Ο εκπαιδευτικός έχει τη δυνατότητα να ζητήσει από τον μαθητή να βελτιώσει την άσκηση και να την

υποβάλλει εκ νέου. Ο τελικός βαθμός της προκύπτει από την βαθμολόγηση που έδωσε ο εκπαιδευόμενος για τον εαυτό του, για την άσκηση και πόσο σωστή ήταν η άσκηση.

- **Βιβλίο**: Το βιβλίο είναι εκπαιδευτικό υλικό σε ηλεκτρονική μορφή. Οι εκπαιδευόμενοι μπορούν μόνο να διαβάσουν το βιβλίο και όχι να το επεξεργαστούν.
- **Διάλογοι**: Ένας εκπαιδευτικός μπορεί να συνομιλήσει με έναν, δύο ή περισσότερους μαθητές. Ένας μαθητής μπορεί να ανοίξει διάλογο με ένα καθηγητή.
- **Επιλογές**: Οι εκπαιδευόμενοι μπορούν να εκφράσουν τη γνώμη τους για ένα θέμα που θα ορίσει ο εκπαιδευτής. Οι επιλογές λειτουργούν σαν ψηφοφορία και έτσι ο εκπαιδευόμενος αποφασίζει για ένα θέμα που τον αφορά. Ο εκπαιδευτής στην ερώτηση που θα θέσει μπορεί να δώσει επιλογή από απαντήσεις και να δει την άποψή τους για κάποιο θέμα.
- **Εργασίες ή Αναθέσεις**: Ο εκπαιδευτικός ορίζει μια εργασία στους εκπαιδευόμενους και αναμένει τη δημιουργία ενός αρχείου, που θα περιέχει την εργασία, το οποίο θα «ανεβάσουν» στη σελίδα.
- **Εργαστήρια**: Μέσω των εργαστηρίων οι εκπαιδευόμενοι μπορούν να αξιολογήσουν τις εργασίες τους ή τα δείγματα εργασιών που τους έχει δώσει ο καθηγητής. Τα εργαστήρια είναι ένα είδος αξιολόγησης με πολλές επιλογές.
- **Έρευνες**: Ο μαθητής μέσω κάποιων ερευνών μπορεί να πει την άποψή του για το μάθημα, τη διδασκαλία και την ύλη. Από τις έρευνες αυτές οι εκπαιδευτικοί λαμβάνουν feedback από τους μαθητές και βελτιώνονται.
- **Quiz**: Το quiz είναι ενός είδους test το οποίο περιέχει ερωτήσεις με τη μορφή: πολλαπλής επιλογής, Σωστό/Λάθος, κλπ. Το quiz αποθηκεύεται σε μια Βάση Δεδομένων με σκοπό την επαναχρησιμοποίησή του.
- **Λεξικά**: Ο εκπαιδευτής μπορεί να δημιουργήσει ένα λεξικό όρων και ο εκπαιδευόμενος να το χρησιμοποιήσει. Το μάθημα μπορεί να έχει ένα «βασικό» λεξικό και κάποια δευτερεύοντα. Ο μαθητής

μπορεί να καταχωρήσει όρους στα δευτερεύοντα λεξικά, να τους επεξεργαστεί και να τους διαγράψει. Ο εκπαιδευόμενος μπορεί να πραγματοποιήσει μόνο αναζήτηση στο βασικό λεξικό, ωστόσο ο εκπαιδευτής μπορεί να μεταφέρει τις εγγραφές του δευτερεύοντος λεξικού στο βασικό.

- **Μαθήματα**: Ο εκπαιδευόμενος έχει τη δυνατότητα να δει τμηματικά κάποιο μάθημα. Στο τέλος κάθε τμήματος υπάρχουν και ορισμένες ερωτήσεις όπου ανάλογα με την απάντηση που θα δώσει ο μαθητής θα τον οδηγήσουν και στο αντίστοιχο τμήμα μαθήματος.
- **Ομάδες συζητήσεων**: Δίνεται η δυνατότητα στους εκπαιδευόμενους να λαμβάνουν μέρος σε ομάδες συζητήσεων (forum) ανταλλάσσοντας απόψεις με άλλους εκπαιδευόμενους.
- **Συνομιλίες πραγματικού χρόνου**: Ο κάθε εκπαιδευόμενος μπορεί μέσω συνομιλιών πραγματικού χρόνου (chat) να μιλάει με άλλους συμμετέχοντες σε πραγματικό χρόνο.
- **SCORM**: Το SCORM προέρχεται από τις λέξεις Shareable Content Object Reference Model και είναι ένα σύστημα χρήσης μαθησιακού περιεχομένου και βρίσκεται στο διαδίκτυο ως αντικείμενο εκμάθησης του e-learning. Το πακέτο SCORM αποτελείται από γραφικά, ιστοσελίδες, προγράμματα JavaScript, οτιδήποτε γενικότερα λειτουργεί σε έναν Web Browser.
- **Wikis**: Συλλογική συγγραφή αρχείων με τη χρήση μιας γλώσσας προγραμματισμού χρησιμοποιώντας Web Browser.

2.9 Αρχιτεκτονική του Moodle

Σε αυτή την ενότητα θα μελετήσουμε και θα αναλύσουμε την αρχιτεκτονική της πλατφόρμας Moodle. Η πλατφόρμα που μελετάμε τρέχει σε οποιοδήποτε σύστημα υποστηρίζει τη γλώσσα προγραμματισμού PHP και είναι συμβατή με διάφορες Βάσεις Δεδομένων (MySQL, PostgreSQL, MSSQL, Oracle, SQLite). Η ιδανική λύση είναι να τρέχει σε ένα σύστημα με Apache Web Server και MySQL.

Το Moodle αποτελείται από τον κατάλογο εφαρμογών, τον κατάλογο δεδομένων και από τη Βάση Δεδομένων.

- **Κατάλογος Εφαρμογών**: Ο Κατάλογος Εφαρμογών του Moodle είναι ένας κατάλογος που αποτελείται από υποκαταλόγους για την αποθήκευση διαφόρων αρθρωμάτων.
- **Κατάλογος Δεδομένων**: Ο Κατάλογος Δεδομένων του Moodle περιέχει όλα τα αρχεία (εργασίες, κείμενα, κ.α.) που ανεβάζουν στην πλατφόρμα οι εκπαιδευτικοί και οι εκπαιδευόμενοι.
- **Βάση Δεδομένων**: Η Βάση Δεδομένων του Moodle περιλαμβάνει όλο το υλικό των μαθημάτων όπως είναι οι εργασίες, τα εργαστήρια, τα μαθήματα, οι ασκήσεις, τα στοιχεία των καθηγητών και των μαθητών.

Παρακάτω θα δούμε, από την πλευρά του διαχειριστή, ποια είναι τα κριτήρια με βάση τα οποία σχεδιάστηκε το Moodle.

I. **Το Moodle πρέπει να είναι συμβατό με πολλές πλατφόρμες**. Η πλατφόρμα της δικτυακής εφαρμογής που τρέχει στα πιο πολλά συστήματα έχει υλοποιηθεί με PHP και MySQL. Στο περιβάλλον αυτό (PHP + MySQL) έχει υλοποιηθεί και το Moodle το οποίο διαθέτει ένα επιπλέον χαρακτηριστικό καθώς εκτός από τη MySQL υποστηρίζει μια πληθώρα από Βάσεις Δεδομένων.
II. **Πρέπει η αναβάθμιση της πλατφόρμας από την μία έκδοση στην άλλη να γίνεται εύκολα**. Το Moodle διαθέτει ένα μηχανισμό μέσω του οποίου πραγματοποιεί αυτόματα αναβάθμιση της έκδοσης.
III. **Πρέπει η πλατφόρμα να αποτελείται από υπομονάδες για να επεκτείνεται**. Τα χαρακτηριστικά του Moodle (δραστηριότητες, θέματα, διατάξεις μαθημάτων) αποτελούνται από υπομονάδες. Έτσι, μπορεί οποιοσδήποτε να τα επεκτείνει.
IV. **Πρέπει η πλατφόρμα να συνδέεται με άλλα συστήματα και να χρησιμοποιείται με αυτά**.
V. **Το Moodle εγκαθίσταται, τροποποιείται και χρησιμοποιείται εύκολα**.

2.9.1 Κατάλογος Εφαρμογών του Moodle

Ο Κατάλογος Εφαρμογών περιέχει διάφορους υποκαταλόγους το όνομα των οποίων αντιστοιχεί στο περιεχόμενό τους. Κάποιοι από τους κυριότερους υποκαταλόγους είναι ο admin που περιέχει τον PHP κώδικα των σελίδων διαχείρισης, ο mod που περιέχει τα αρθρώματα (modules), ο lang που περιέχει τις μεταφράσεις του interface και ο course που εμπεριέχει τα μαθήματα που θα δει ένας επισκέπτης. Εκτός από υποκαταλόγους στον Κατάλογο Εφαρμογών του Moodle περιέχονται και διάφορα αρχεία. Το αρχείο index.php έχει τον PHP κώδικα της Αρχικής Σελίδας του Moodle της ιστοσελίδας μας.

2.9.2 Κατάλογος Δεδομένων του Moodle

Στον Κατάλογο Δεδομένων αποθηκεύονται τα αρχεία που ανεβάζουν στο Moodle οι εκπαιδευτές και οι εκπαιδευόμενοι. Για λόγους ασφαλείας ο κατάλογος αυτός τοποθετείται εκτός του φακέλου που περιέχει τα έγγραφα του Server.

2.9.3 Βάση Δεδομένων του Moodle

Η Βάση Δεδομένων είναι υπεύθυνη για τη διατήρηση των περισσότερων πληροφοριών της ιστοσελίδας. Μέσω του Moodle μπορούμε να δημιουργήσουμε ιστοσελίδες για τα μαθήματα, οπότε ο HTML κώδικας, τα Links, τα wikis, οι ομάδες συζητήσεων (forums) και όλες οι ρυθμίσεις αποθηκεύονται στη Βάση Δεδομένων.

2.10 Λόγοι Επιλογής του Moodle

Παρακάτω θα δούμε τους λόγους για τους οποίους επιλέξαμε την πλατφόρμα Ασύγχρονης Τηλεκπαίδευσης Moodle.

Το Moodle είναι μια ιδιαίτερα ασφαλές πλατφόρμα καθώς ο ρόλος που έχει ο κάθε χρήστης είναι συγκεκριμένος και ελέγχεται. Ακόμη, η πλατφόρμα είναι χρηστική διότι έχει πάρα πολλές δυνατότητες οι οποίες είναι εύκολα αξιοποιήσιμες από τον χρήστη. Ένας ακόμη λόγος που επιλέξαμε το Moodle είναι το γεγονός ότι είναι μια ανοικτού κώδικα και ευέλικτη πλατφόρμα σε σύγκριση με άλλες εμπορικές.

Η υποστήριξη που παρέχει το συγκεκριμένο Σύστημα Διαχείρισης Μάθησης στους χρήστες και τους διαχειριστές του είναι πολύ καλά οργανωμένη λόγω του ότι έχει πολύ ενεργή κοινότητα. Τέλος, είναι το πιο διαδεδομένο LMS για τη δημιουργία ιστοσελίδων με εκπαιδευτικό περιεχόμενο, επειδή παρέχει πολλές πρόσθετες εκπαιδευτικές δραστηριότητες.

3

ΤΟ PHP FRAMEWORK LARAVEL

3.1 Η Γλώσσα Προγραμματισμού PHP

Η PHP είναι μια scripting (ερμηνεύεται κατά την εκτέλεση) γλώσσα προγραμματισμού που χρησιμοποιείται για τη δημιουργία δυναμικών Web σελίδων. Με τον όρο δυναμική σελίδα εννοούμε ότι το περιεχόμενο της σελίδας που βλέπει ο κάθε χρήστης είναι διαφορετικό και εξαρτάται από παράγοντες όπως είναι οι κινήσεις του χρήστη στη σελίδα, το Λειτουργικό του Σύστημα, κλπ. Μπορούμε εύκολα να ενσωματώσουμε PHP κώδικα σε μια σελίδα που περιέχει HTML κώδικα. Ο PHP κώδικας θα εκτελείται κάθε φορά που θα επισκεπτόμαστε τη σελίδα, μεταφράζεται στον Server και δημιουργεί HTML.

Η αρχική ονομασία της γλώσσας ήταν PHP - Personal Home Page, ενώ αργότερα μετονομάστηκε σε PHP - Hypertext Preprocessor. Για τη δημιουργία της PHP ο Rasmus Lerdorf βασίστηκε στις γλώσσες Pearl και C. Σήμερα βρισκόμαστε στη βασική έκδοση PHP 7.

Η PHP εκτελείται στον Server (Server - Side) σε αντίθεση με άλλες γλώσσες προγραμματισμού, όπως είναι η JavaScript, οι οποίες εκτελούνται στον Browser του χρήστη (Client - Side). Η PHP εκτός από τις Server – Side εφαρμογές, μπορεί να χρησιμοποιηθεί και στην αποστολή και λήψη cookies, στη συλλογή δεδομένων, στην παραγωγή δυναμικού περιεχομένου στις ιστοσελίδες.

3.2 Πλεονεκτήματα της PHP

Εκτός από την PHP υπάρχουν και κάποιες ακόμη αντίστοιχου τύπου γλώσσες προγραμματισμού όπως είναι οι Microsoft Active Server Pages (ASP), Java Server Pages (JSP), Perl. Παρακάτω θα δούμε τα πλεονεκτήματα της PHP συγκριτικά με τις παραπάνω γλώσσες προγραμματισμού.

- **Συνεργάζεται με πληθώρα Συστημάτων Βάσεων Δεδομένων**

  Εκτός από τη MySQL η PHP μπορεί να συνδεθεί και με άλλες Βάσεις Δεδομένων τόσο εμπορικές όσο και open source: PostgreSQL, MSSQL, Sybase, Oracle, InterBase, IBM DB2, DBase, Solid, κ.α. Επίσης χρησιμοποιώντας το Open Database Connectivity Standard (ODBC) μπορούμε να συνδεθούμε σε οποιαδήποτε Βάση Δεδομένων χρησιμοποιεί ένα ODBC driver.

- **Έχει ενσωματωμένες βιβλιοθήκες**

  Η PHP έχει πολλές ενσωματωμένες βιβλιοθήκες που εκτελούν λειτουργίες σχετικές με το Web, διότι σχεδιάστηκε για να χρησιμοποιείται στο Web. Μπορούμε να δημιουργήσουμε έγγραφα PDF, εικόνες GIF, να συνδεθούμε με άλλες υπηρεσίες δικτύων και πολλά ακόμα τα οποία μπορούμε να δούμε στο official documentation της PHP (http://www.php.net).

- **Ευκολία στη χρήση**

  Η PHP είναι απλή στη χρήση καθώς η σύνταξή της στηρίζεται πάνω στη C και την Perl. Αν γνωρίζετε ήδη να προγραμματίζετε σε C/C++ ή Java τότε η PHP θα είναι πολύ εύκολη στη χρήση.

- **Μηδενικό κόστος**

  Για να δουλέψουμε με την PHP αρκεί μόνο να κατεβάσουμε δωρεάν την τελευταία της έκδοση από το http://www.php.net.

- **Ο κώδικας προέλευσης είναι διαθέσιμος**

  Έχουμε τη δυνατότητα να προσθέσουμε, να διορθώσουμε ή να αλλάξουμε ό,τι θέλουμε στη γλώσσα σε αντίθεση με άλλα εμπορικά προγράμματα.

- **Ανεξάρτητη από την πλατφόρμα**

  Η PHP εκτελείται με τον ίδιο τρόπο ανεξάρτητα από το Λειτουργικό Σύστημα του υπολογιστή όπου εκτελείται.

- **Χαλαρό σύστημα τύπων**

  Στην PHP δε χρειάζεται να δηλώσουμε αν ένας αριθμός είναι short ή long.

3.3 Εισαγωγή στα PHP Frameworks

Όπως είδαμε και στην προηγούμενη ενότητα η PHP είναι μια γλώσσα διαδικτυακού προγραμματισμού η οποία χρησιμοποιείται κατά κόρον στην ανάπτυξη ιστοσελίδων και διαδικτυακών εφαρμογών, καθώς είναι πιο δημοφιλής από τις Active Server Pages (ASP), Java Server Pages (JSP), Ruby, κ.α.

Στις μέρες μας είναι δύσκολο και χρονοβόρο να δημιουργήσουμε πολύπλοκες ιστοσελίδες ή εφαρμογές γράφοντας από το μηδέν κώδικα σε PHP. Πλέον με τη βοήθεια των PHP Framewokrs μπορούμε εύκολα και γρήγορα να αναπτύξουμε ιστοσελίδες ή εφαρμογές.

3.3.1 Γιατί να χρησιμοποιήσουμε PHP Framework

Παρακάτω θα δούμε τους λόγους για τους οποίους μπορούμε να χρησιμοποιήσουμε κάποιο PHP framework.

- Γράφουμε λιγότερο κώδικα σε πιο σύντομο χρονικό διάστημα.

- Τα PHP frameworks ακολουθούν το αρχιτεκτονικό πρότυπο MVC (Model-View-Controller) στο οποίο διαχωρίζεται το επίπεδο της παρουσίασης από αυτό της λογικής.
- Αναπτύσσουμε web εφαρμογές ή ιστοσελίδες με εργαλεία Object-Oriented προγραμματισμού.
- Μας προτρέπουν για καλύτερη δομή της εφαρμογής με τελικό σκοπό ο προγραμματιστής να μπορεί να τη διαχειριστεί και να τη συντηρεί ευκολότερα όσο μεγαλώνει.
- Τα PHP frameworks παρέχουν ήδη υλοποιημένες (Pre Build) βιβλιοθήκες και εργαλεία για χειρισμό των συνεδριών και των cookies (Session and Cookie Handling), για επικύρωση φορμών (Form Validation).
- Είναι κατάλληλα για άτομα που δουλεύουν σε μια ομάδα.
- Παρέχουν οργανωμένες και ενεργές κοινότητες με πολύ καλή υποστήριξη σε οποιοδήποτε πρόβλημα προκαλείται στην πλατφόρμα.
- Τα PHP frameworks περιλαμβάνουν χρήσιμες συναρτήσεις για υλοποίηση φορμών, URLs, ερωτημάτων SQL, κ.α.
- Προσφέρουν συνεχή αναβάθμιση και συντήρηση των εκδόσεων.
- Βοηθούν στην ασφάλεια των εφαρμογών.

3.4 Το PHP Framework Laravel

Το Laravel είναι ένα ανοικτού κώδικα PHP framework που δημιουργήθηκε από τον Taylor Otwell το 2011, για την ανάπτυξη Web εφαρμογών ακολουθώντας το αρχιτεκτονικό πρότυπο Model – View – Controller (MVC). Με το αρχιτεκτονικό πρότυπο MVC θα ασχοληθούμε αναλυτικά σε επόμενη ενότητα. Το Laravel είναι ένα απλό, κομψό και καλά τεκμηριωμένο PHP Framework.

Είναι απλό διότι οι περιέχει απλές στην κατανόηση και υλοποίηση λειτουργίες. Όποιος έχει προηγούμενη εμπειρία με άλλα MVC frameworks είναι εύκολη η μετάβασή του στο Laravel.

Είναι κομψό καθώς βασίζεται στα standard της εποχής τα οποία θέλουν απλό κώδικα για την εκτέλεση απλών λειτουργιών.

Διαθέτει πλήρη και ενημερωμένη τεκμηρίωση. Στο internet υπάρχουν πάρα πολλοί οδηγοί και πηγές για κάποιον που θέλει να μάθει το Laravel.

3.4.1 Το Laravel από το 2011 στο 2016

Ο Taylor Otwell δημιούργησε το Laravel σαν μια εναλλακτική λύση για το PHP framework CodeIgniter, το οποίο δεν παρείχε ενσωματωμένη υποστήριξη για τον έλεγχο ταυτότητας και την εξουσιοδότηση χρήστη (User Authentication & Authorization). Η πρώτη έκδοση του Laravel έγινε διαθέσιμη στις 9 Ιουνίου 2011. Το Laravel 1 περιείχε ενσωματωμένη υποστήριξη για έλεγχο ταυτότητας χρήστη (user authentication), για μοντέλα (models), views, συνεδρίες (sessions), routing αλλά και για άλλους μηχανισμούς. Ωστόσο, η πρώτη έκδοση δεν υποστήριζε ελεγκτές (controllers) που την απέτρεπαν από το να είναι ένα MVC framework.

Λίγους μήνες αργότερα τον Σεπτέμβριο του 2011 κυκλοφόρησε η δεύτερη έκδοση του PHP framework, η οποία είχε βελτιώσεις από τον Taylor Otwell και από την κοινότητα του Laravel. Το Laravel 2 υποστήριζε τους ελεγκτές (controllers) γεγονός το οποίο έκανε το Laravel 2 να είναι πλέον ένα MVC framework. Ακόμη περιείχε ένα σύστημα με templates (Blade Templating System) το οποίο είναι υπεύθυνο για την εμφάνιση των σελίδων. Μειονέκτημα θεωρείται ότι δεν υποστήριζε πακέτα από τρίτους κατασκευαστές.

Τον Φεβρουάριο του 2012 κυκλοφόρησε η έκδοση 3 του Laravel η οποία αποτελούνταν από διάφορα νέα χαρακτηριστικά συμπεριλαμβανόμενου του Artisan Command Line Interface. Ακόμη, το Laravel 3 υποστήριζε περισσότερα Συστήματα Διαχείρισης Βάσης Δεδομένων, χειρισμό events, Database Migrations (Μεταναστεύσεις Βάσεων Δεδομένων) και ένα σύστημα πακέτων που ονομάζεται Bundles.

Το Laravel 4 κυκλοφόρησε τον Μάιο του 2013 και είχε την κωδική ονομασία Illuminate. Τα χαρακτηριστικά που συνόδευαν το Laravel 4 ήταν η προσθήκη ενός τρόπου να προσθέτεις στοιχεία στη Βάση Δεδομένων (Database Seeding), η ενσωματωμένη υποστήριξη για αποστολή διαφόρων τύπων email και υποστήριξη για καθυστερημένη διαγραφή των εγγραφών της Βάσης Δεδομένων.

Η έκδοση 5 του Laravel κυκλοφόρησε τον Φεβρουάριο του 2015. Τα νέα χαρακτηριστικά του Laravel 5 περιλάμβαναν την υποστήριξη προγραμματισμό διεργασιών μέσα από ένα πακέτο που ονομάζεται Scheduler, βελτιωμένο χειρισμό στοιχείων μέσω του πακέτου Elixir και χειρισμό του ελέγχου ταυτότητας χρήστη (Authentication) μέσω του πακέτου Socialite. Τέλος, στην έκδοση 5 εισάχθηκε μια νέα δομή των καταλόγων για τις υλοποιημένες εφαρμογές.

Τον Ιούνιο του ίδιου χρόνου κυκλοφόρησε η πρώτη έκδοση του Laravel με μακροπρόθεσμη υποστήριξη (Long Term Support LTS). Αυτή είναι η έκδοση 5.1. Οι LTS εκδόσεις έχουν προγραμματιστεί να κυκλοφορούν κάθε δύο χρόνια.

Πλέον, βρισκόμαστε στην έκδοση 5.2 του Laravel framework η οποία συνεχίζει τις βελτιώσεις που έγιναν στο Laravel 5.1 προσθέτοντας την υποστήριξη πολλαπλών οδηγών αυθεντικοποίησης, βελτιώσεις στην επικύρωση των πινάκων, κ.α.

3.4.2 Χαρακτηριστικά του Laravel

Σε αυτή την ενότητα θα δούμε κάποια από τα κυριότερα χαρακτηριστικά του Laravel τα οποία το καθιστούν το πιο δημοφιλές PHP framework για Web εφαρμογές.

- Το Laravel έχει δική του μηχανή προτύπων που ονομάζεται **Blade Templating Engine**. Το πλεονέκτημα που διαθέτει η Blade Templating Engine είναι ότι μπορούμε να γράψουμε απλό PHP κώδικα.

- Χαρακτηρίζεται από την **κομψή** και εύκολη στο διάβασμα **σύνταξή** του.
- **Composer Ready**. Ο Composer είναι ένα εργαλείο μέσω του οποίου μπορούμε εύκολα να εγκαταστήσουμε στην εφαρμογή μας πακέτα από τρίτους (third-party packages).
- **Eloquent Object Relational Mapping** (ORM). Μέσω του Eloquent ORM η πρόσβαση στη Βάση Δεδομένων MySQL γίνεται πιο εύκολα. Παρέχει τις εσωτερικές μεθόδους για τους περιορισμούς στις σχέσεις μεταξύ των αντικειμένων της Βάσης Δεδομένων.
- Οι **RESTfull Controllers** διαχωρίζουν τη λογική πίσω από τα HTTP GET και HTTP POST αιτήματα.
- Διαθέτει **έξυπνο σχεδιασμό** ο οποίος προσφέρει ευελιξία στους προγραμματιστές που το χρησιμοποιούν.
- Έχει μια πολύ **μεγάλη** και ενεργή **κοινότητα** η οποία δίνει βοήθεια σε διαφόρων ειδών θέματα.
- **Query Builder**. Το Query Builder παρέχει διάφορες κλάσεις και μεθόδους οι οποίες παίρνουν τη θέση των SQL ερωτημάτων (Raw SQL Queries).
- Η λογική της εφαρμογής του Laravel (**Application Logic**) είναι άρρηκτα συνδεδεμένη με την ανάπτυξη των εφαρμογών. Υλοποιείται χρησιμοποιώντας ελεγκτές (Controllers) ή μέσω του Routing.
- **Form Requests**. Τα Form Requests αποσκοπούν στην επικύρωση των δεδομένων που εισάγει ο χρήστης σε μια φόρμα.
- Το **Socialite** είναι ένα προαιρετικό πακέτο του Laravel που έχει μηχανισμούς για τον έλεγχο ταυτότητας χρήστη (Authentication) με παρόχους όπως το GitHub, το Facebook, το Twitter και το Google.
- **Database Seeding**. Προσφέρει έναν απλό τρόπο να συμπληρώσουμε τους πίνακες της Βάσης Δεδομένων με επιλεγμένα

default δεδομένα τα οποία χρησιμοποιούνται για τον έλεγχο της εφαρμογής.

- **Elixir**. Το Elixir σχετίζεται με το Front-End καθώς υποστηρίζει τα CSS, τη JavaScript και εργαλεία για δοκιμή.

3.5 Λόγοι επιλογής του Laravel

Στην παρακάτω ενότητα θα δούμε κάποιους από τους σημαντικότερους λόγους οι οποίοι μας οδήγησαν στην επιλογή του Laravel framework. Σύμφωνα με το sitepoint.com το Laravel είναι το πιο δημοφιλές PHP Framework, τόσο σε εταιρικά projects όσο και σε ατομικά. Επιπλέον, με την πάροδο του χρόνου αυξάνεται το ενδιαφέρον των προγραμματιστών ως προς το Laravel Framework. Άλλος ένας λόγος είναι το πολύ καλά οργανωμένο documentation που διαθέτει και η μεγάλη βιβλιογραφία που βρίσκεται πίσω από αυτό.

Το Laravel διαθέτει μια πλήρες «συλλογή» από πακέτα τα οποία είναι το Laravel Homestead, το Forge, το Enoyer, το Lumen και το Laracasts.

Το Laravel Homestead είναι ένα πακέτο με περιβάλλον για ανάπτυξη εφαρμογών χωρίς να χρειάζεται να εγκαταστήσουμε την PHP, τη MySQL, κάποιον Web Server, ή οποιοδήποτε άλλο λογισμικό.

Με το Laravel Forge έχουμε δυνατότητες δημιουργίας και διαχείρισης ενός Web Server και μπορούμε να φιλοξενούμε την εφαρμογή μας σε αυτόν τον Server.

Το Laravel Envoyer είναι μια υπηρεσία ανάπτυξης PHP εφαρμογών με μηδενικό downtime. Έτσι, οι προγραμματιστές μπορούν να κάνουν αλλαγές στην εφαρμογή χωρίς να επηρεάζονται οι τελικοί χρήστες.

Το Laravel Lumen δημιουργήθηκε και αυτό από τον δημιουργό του Laravel και είναι ένα Micro-Framework. Δηλαδή είναι μικρότερο, γρηγορότερο και αποτελεί μια απλούστερη έκδοση του Laravel Framework.

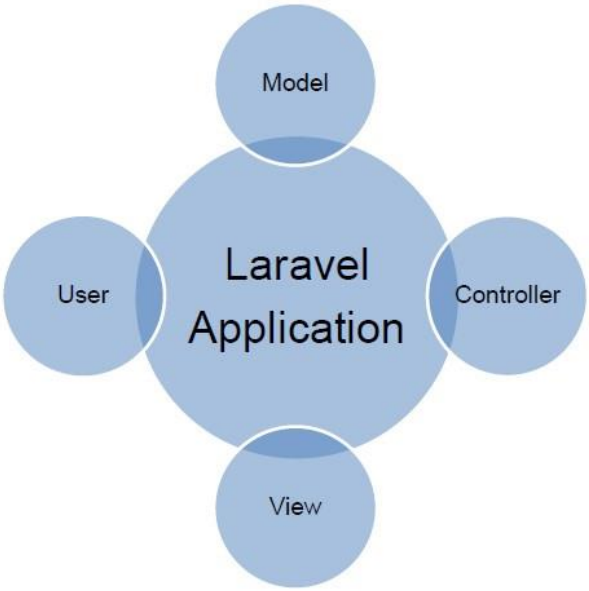
Τέλος, το Laracasts είναι μια πλατφόρμα του Jeffrey Way η οποία προσφέρει πληθώρα δωρεάν ή επί πληρωμή μαθημάτων (Laravel, JavaScript, PHP, κ.α.) με τη μορφή video.

3.6 Αρχιτεκτονική του Laravel (MVC)

Σε αυτή την ενότητα θα αναλύσουμε το μοντέλο αρχιτεκτονικής Model – View – Controller (MVC). Στο μοντέλο MVC διαχωρίζεται η λογική της εφαρμογής από τη λογική της παρουσίασης που συνδέεται με ένα γραφικό περιβάλλον χρήστη (Graphical User Inerface).

3.6.1 Model

Το πρώτο συστατικό γύρω από το οποία χτίζεται μια Web εφαρμογή είναι το Model. Το Model είναι ένα μόνιμο χαρακτηριστικό και αποθηκεύεται σε μια Βάση Δεδομένων. Είναι το επίπεδο στο οποίο κατασκευάζουμε σύνθετους αλγόριθμους, πολύπλοκες πράξεις και πραγματοποιούμε ενέργειες στη Βάση Δεδομένων όπως ενημέρωση στοιχείων, εισαγωγή, διαγραφή, κλπ.



Εικόνα 1 Μοντέλο αρχιτεκτονικής Model View Controller

3.6.2 View

Το View είναι το συστατικό με το οποίο ασχολούμαστε με τον τρόπο παρουσίασης των πληροφοριών στο χρήστη. Το επίπεδο παρουσίασης είναι υπεύθυνο για τη δημιουργία ενός περιβάλλοντος χρήστη βασισμένου στα δεδομένα του Model.

3.6.3 Controller

Το τρίτο και τελευταίο συστατικό της MVC αρχιτεκτονικής είναι ο Controller ο οποίος βρίσκεται μεταξύ του Model και του View και τα συντονίζει. Ο ελεγκτής (Controller) επεξεργάζεται τα αιτήματα και πραγματοποιεί τις κατάλληλες ενέργειες. Όταν δεχθεί ένα αίτημα για το οποίο απαιτούνται να γίνουν σύνθετες διεργασίες ο Controller επικοινωνεί αρχικά με το Model και στη συνέχεια προωθεί το αποτέλεσμα στο View για να το εμφανίσει.

3.7 Composer

Ο Composer είναι ένα εργαλείο για να διαχειριζόμαστε τα διάφορα πακέτα της PHP. Η κύρια λειτουργία του Composer είναι να κατεβάσει και να ενημερώσει τα PHP πακέτα που χρησιμοποιούμε στην εφαρμογή μας.

3.8 Artisan CLI

Το Artisan Command-Line Interface (CLI) είναι ένα βοηθητικό πρόγραμμα γραμμής εντολών (Command-Line) μέσω του οποίου χειριζόμαστε το περιβάλλον του project μας και είναι ενσωματωμένο στο Laravel Framework. Έτσι, ο προγραμματιστής χρησιμοποιώντας το Artisan CLI αλληλοεπιδρά με το Laravel Framework. Με το Artisan μπορούμε να γράψουμε εντολές που σχετίζονται με τη Βάση Δεδομένων, με τους Controllers, τα Models, τα Views, κ.α.

4

«ΕΙΣΑΓΩΓΗ ΣΤΟ LARAVEL»

4.1 Το Μάθημα

4.1.1 Εισαγωγή

Στο τελευταίο κεφάλαιο θα σας παρουσιάσω αναλυτικά της ενότητες του μαθήματος «Εισαγωγή στο Laravel». Ωστόσο εκτός από το θεωρητικό κομμάτι που υπάρχει στις παρακάτω ενότητες υπάρχει και το κομμάτι με τα αντίστοιχα υλοποιημένα παραδείγματα σε παράρτημα που ακολουθεί.

4.1.2 Σκοπός του Μαθήματος

Σκοπός του μαθήματος είναι η εκμάθηση των δυνατοτήτων του PHP Framework Laravel στους εκπαιδευόμενους μέσα από μια σειρά 11 εκπαιδευτικών ενοτήτων οι οποίες καλύπτουν όλα όσα πρέπει να γνωρίζει κάποιος για να ξεκινήσει την ανάπτυξη διαδικτυακών εφαρμογών που βασίζονται στο Laravel.

Η οργάνωση της διδασκαλίας του μαθήματος έχει γίνει με τη θεώρηση πως οι εκπαιδευόμενοι είτε έχουν πρότερη επαφή με τις Web τεχνολογίες (HTML, PHP, CSS), είτε όχι καθώς η διδασκαλία του Laravel Framework ξεκινάει από το μηδέν. Κατά τη διάρκεια των εκπαιδευτικών ενοτήτων γίνεται και αναφορά σε βασικές λειτουργικότητες των Web τεχνολογιών. Βέβαια για να είναι οι εκπαιδευόμενοι σε θέση να παρακολουθήσουν το μάθημα πρέπει να έχουν βασικές γνώσεις χειρισμού των Windows.

Η εκπαίδευση σχεδιάστηκε για τη γρήγορη εκμάθηση του PHP Framework Laravel των χαρακτηριστικών και των δυνατοτήτων του. Κατά τη διάρκεια του σεμιναρίου παρουσιάζονται 2 projects. Στο 1ο project παρουσιάζεται η ανάπτυξη ενός website με δωρεές συνδυάζοντας διάφορες γνώσεις που θα έχει λάβει ο εκπαιδευόμενος από τις προηγούμενες ενότητες. Στο 2ο και τελικό project παρουσιάζεται η υλοποίηση μιας πλατφόρμας με Εγγραφή χρήστη (Registration) και Είσοδο χρήστη (Login) συνδυάζοντας τις γνώσεις όλων των ενοτήτων. Τα projects έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να δουν οι εκπαιδευόμενοι όχι μόνο σε θεωρητικό, αλλά και σε πρακτικό επίπεδο πως μπορούν να συνδυάσουν τις λειτουργικότητες που παρέχει το Laravel. Στο μάθημα χρησιμοποιείται η έκδοση 5.1 του Laravel Framework.

4.1.3 Μετά το Μάθημα

Μετά το μάθημα ο εκπαιδευόμενος θα έχει αποκτήσει τις απαραίτητες γνώσεις και δεξιότητες έτσι ώστε :

- Να γνωρίζει τις βασικές λειτουργίες του Laravel Framework.
- Να γνωρίζει τις σύγχρονες τεχνολογίες για την ανάπτυξη Web εφαρμογών.
- Να γνωρίζει τη λειτουργία της MVC αρχιτεκτονικής στην οποία βασίζονται πολλά frameworks (Model-View-Controller).
- Να είναι κατάλληλα προετοιμασμένος για την απαιτητική αγορά εργασίας.
- Να εμπλουτίσει τις γνώσεις του στην PHP.

4.2 Εισαγωγή

4.2.1 Εισαγωγή στο Laravel

Το Laravel είναι ένα ανοιχτού κώδικα (open source) PHP Framework που δημιουργήθηκε με σκοπό την ανάπτυξη διαδικτυακών εφαρμογών ακολουθώντας το αρχιτεκτονικό πρότυπο MVC (Model-View-Controller). Το Laravel μας προσφέρει αρκετές δυνατότητες όπως είναι:

- Επικύρωση Φορμών (Form Validation)
- Εγγραφή Χρήστη (User Registration)
- Πιστοποίηση (Authentication)
- Χειρισμός Συνεδριών (Session Handling)
- Database Migrations & Seeding

Εκτός από το Laravel υπάρχουν και άλλα PHP Frameworks όπως είναι το CakePHP, το CodeIgniter, το ZEND Framework, το Fuelphp, κ.α.

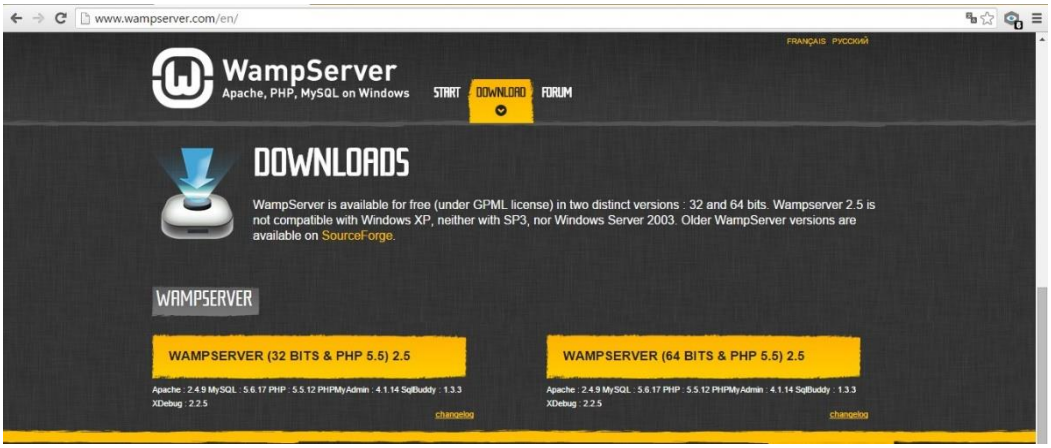
4.2.2 Εγκατάσταση τοπικού Server (Wamp Server)

Για να μπορούμε να υλοποιήσουμε διάφορα παραδείγματα και τα 2 projects στο Laravel πρέπει να μετατρέψουμε τον υπολογιστή μας σε έναν τοπικό Web Server και στη συνέχεια να εγκαταστήσουμε το Laravel. Για τον λόγο αυτό θα χρησιμοποιήσουμε το WAMP.

Το WAMP είναι το ακρωνύμιο των Windows, Apache, MySQL, PHP και είναι ένα πακέτο που περιλαμβάνει την τελευταία έκδοση του Apache, της MySQL και της PHP. Αν δε διαθέτουμε Λειτουργικό Σύστημα Windows αλλά Linux ή MAC OS τότε θα εγκαταστήσουμε το αντίστοιχο LAMP ή MAMP.

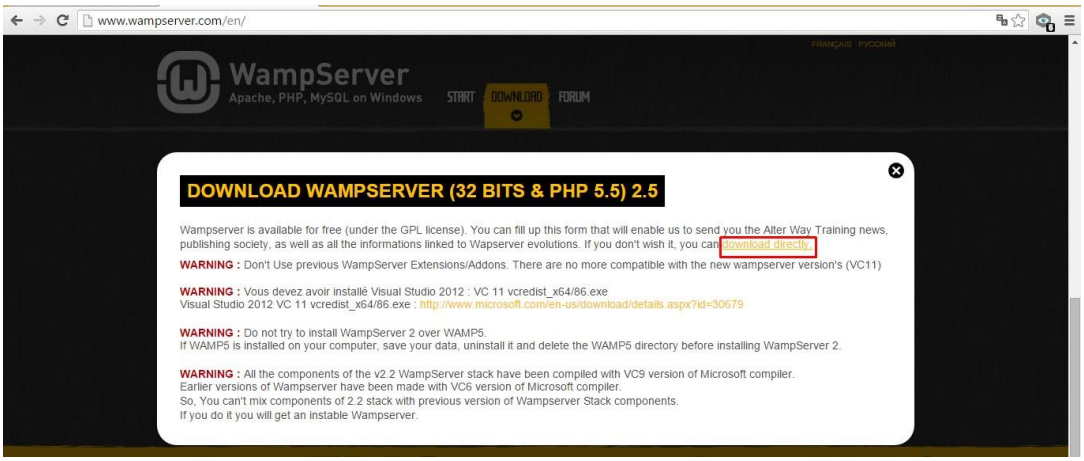
Ο WampServer διανέμεται δωρεάν από τη σελίδα http://wampserver.com/en/. Αφού πατήσουμε στην καρτέλα

“Download” επιλέγουμε την κατάλληλη έκδοση, ανάλογα με την αρχιτεκτονική του συστήματός μας.



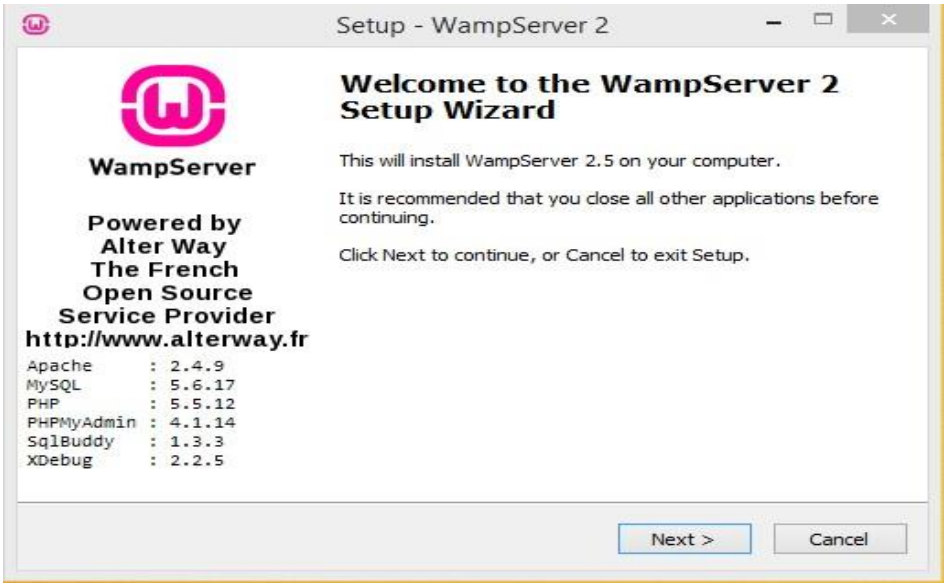
Εικόνα 2 Αρχική Σελίδα Wamp Server

Στη συνέχεια ανοίγει ένα pop-up παράθυρο στο οποίο θα κάνουμε κλικ στον σύνδεσμο “download directly”.



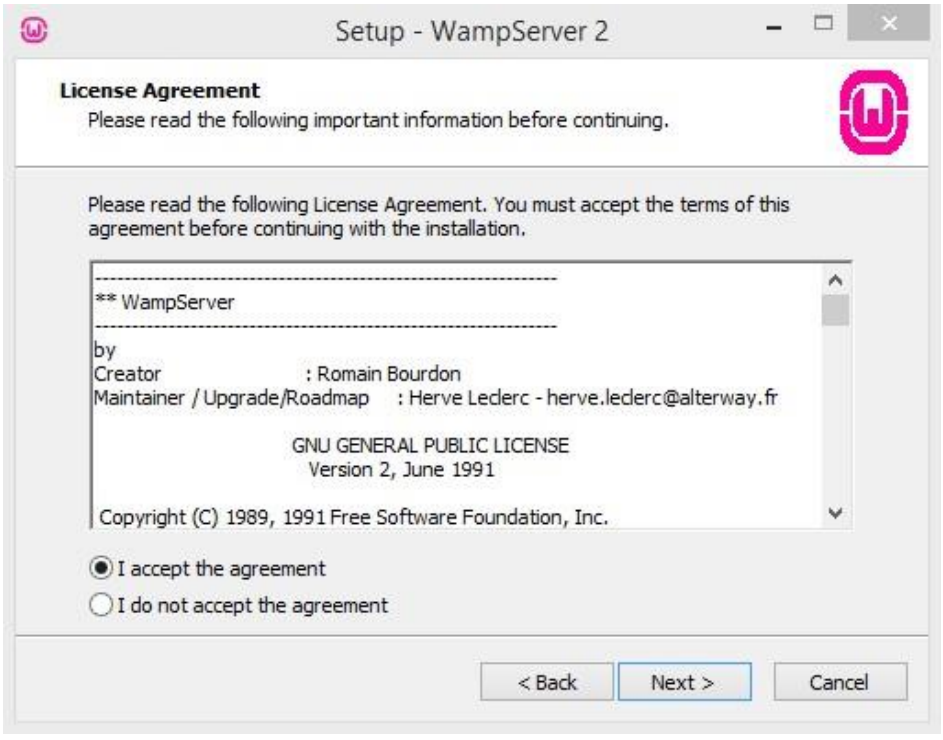
Εικόνα 3 Donwload Page του Wamp Server

Ανοίγουμε το .exe που κατεβάσαμε και πατάμε Next.



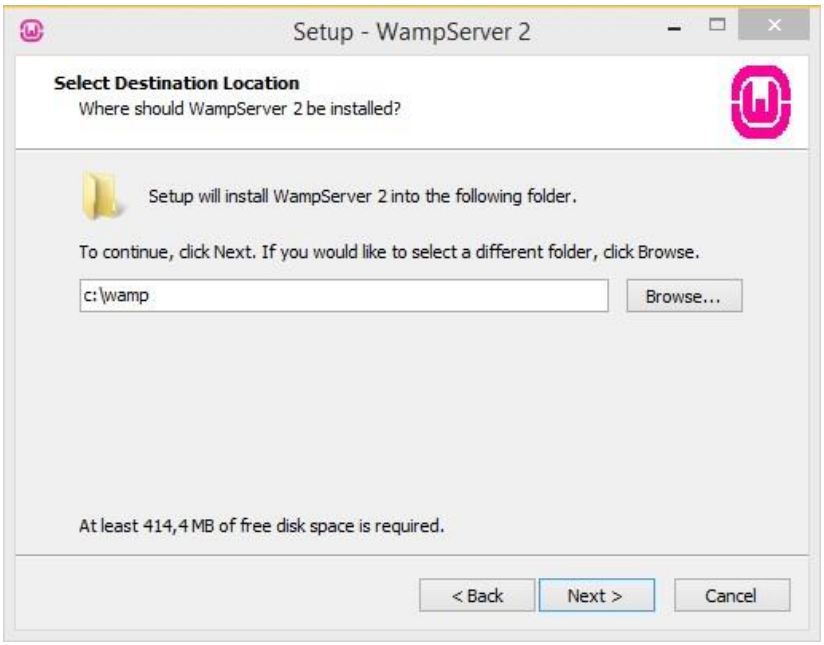
Εικόνα 4 Βήμα 1 εγκατάστασης

Δεχόμαστε τους «Όρους» και πατάμε πάλι Next.



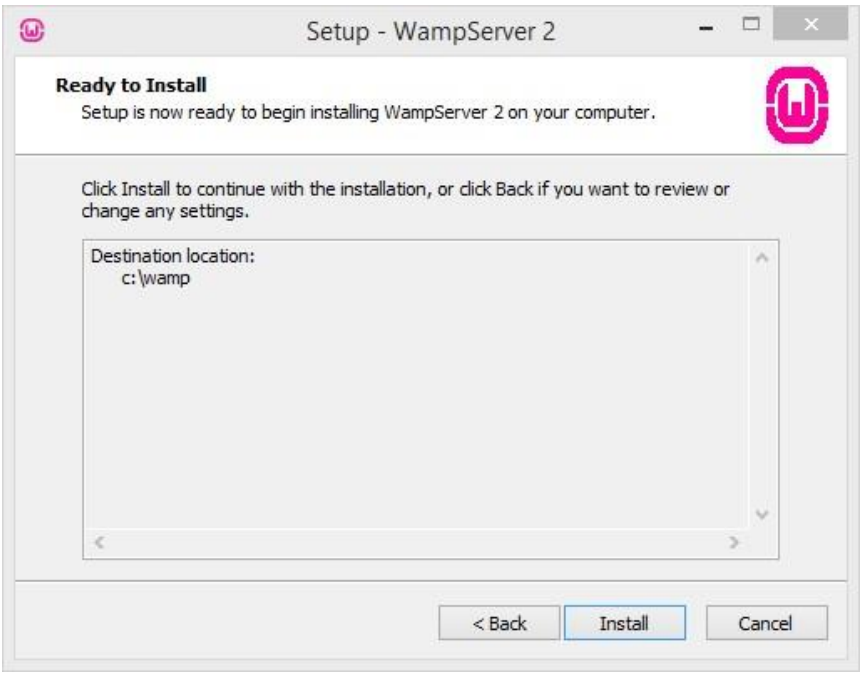
Εικόνα 5 Βήμα 2 εγκατάστασης

Σε αυτό το στάδιο επιλέγουμε σε ποιο σημείο του υπολογιστή μας θέλουμε να εγκατασταθεί ο WampServer και πατάμε Next.



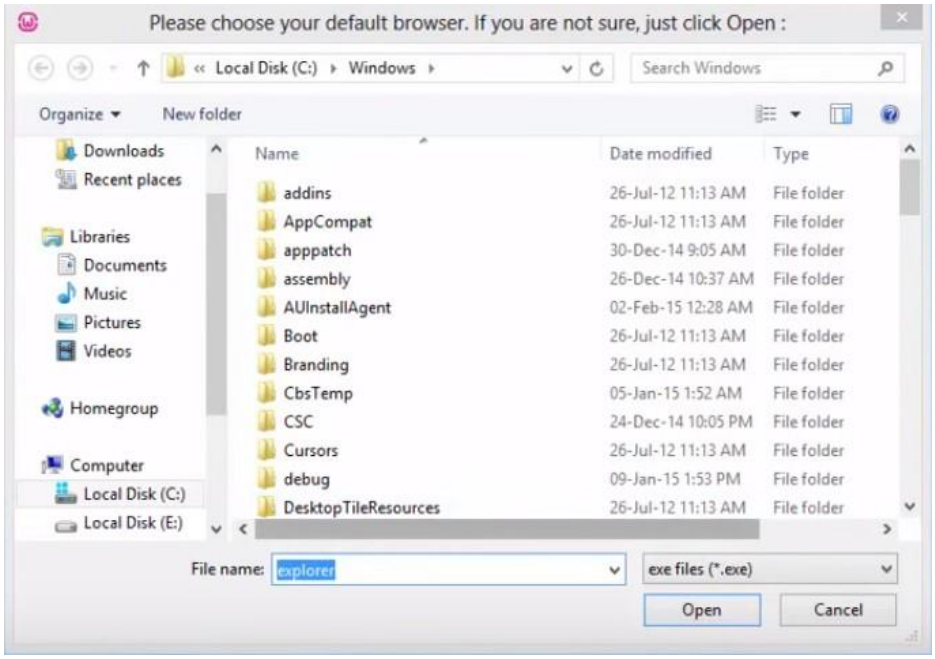
Εικόνα 6 Βήμα 3 εγκατάστασης

Ελέγχουμε όλες τις ρυθμίσεις και εφόσον είναι οι επιθυμητές κάνουμε κλικ στο Install.



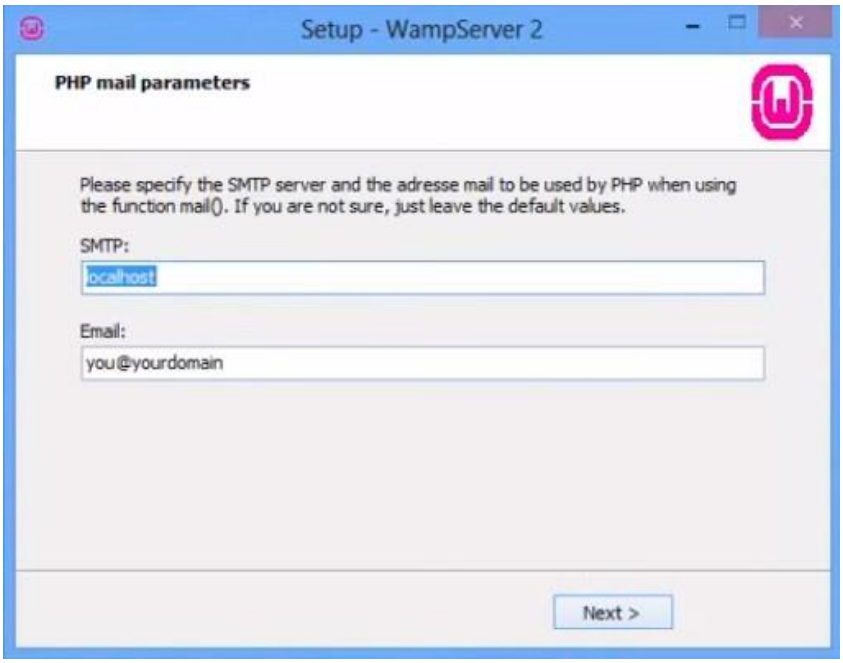
Εικόνα 7 Βήμα 4 εγκατάστασης

Αφού πατήσουμε Install, λίγο αργότερα, εμφανίζεται το παρακάτω pop-up παράθυρο όπου ορίζουμε τον Default Browser και πατάμε Open.



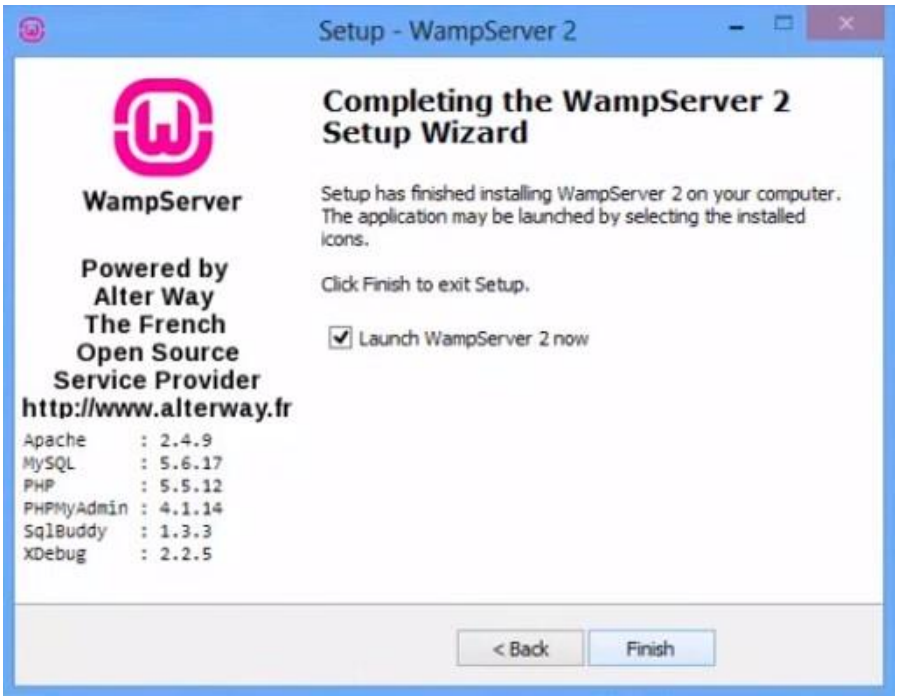
Εικόνα 8 Βήμα 5 εγκατάστασης

Στη συνέχεια αφήνουμε ως έχει τις παραμέτρους PHP mail και πατάμε Next.



Εικόνα 9 Βήμα 6 εγκατάστασης

Τέλος, έχουμε επιλεγμένο το checkbox “Launch WampServer 2 now” και πατάμε Finish.

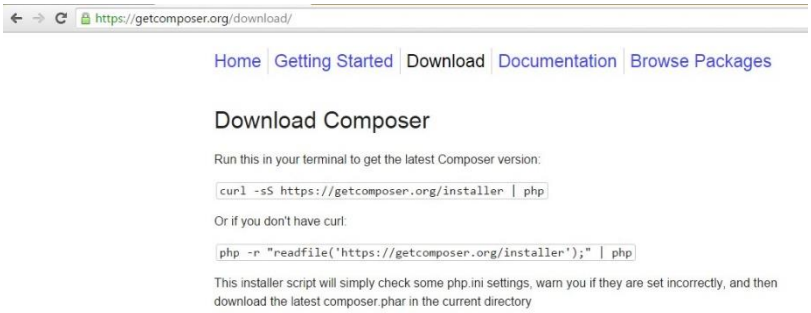


Εικόνα 10 Βήμα 7 εγκατάστασης

4.2.3 Εγκατάσταση Composer & Laravel

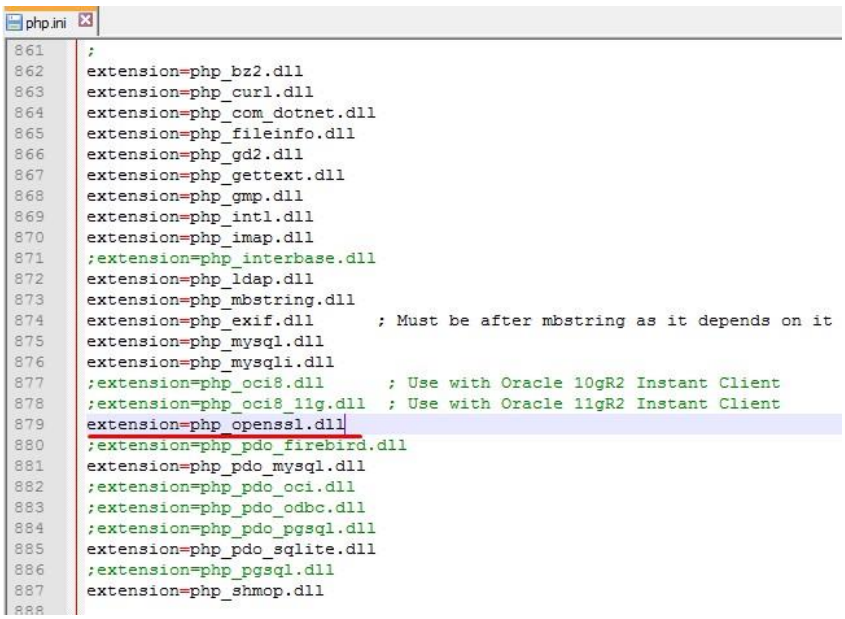
Ο Composer είναι ένα εργαλείο για να διαχειριζόμαστε τα διάφορα πακέτα της PHP. Η κύρια λειτουργία του Composer είναι να κατεβάσει και να ενημερώσει τα PHP πακέτα που χρησιμοποιούμε στην εφαρμογή μας. Για να εγκαταστήσουμε τον Composer θα ακολουθήσουμε τα επόμενα βήματα.

Αρχικά πηγαίνουμε στο https://getcomposer.org και στη συνέχεια στην καρτέλα Download.



Εικόνα 11 Download Page του Composer

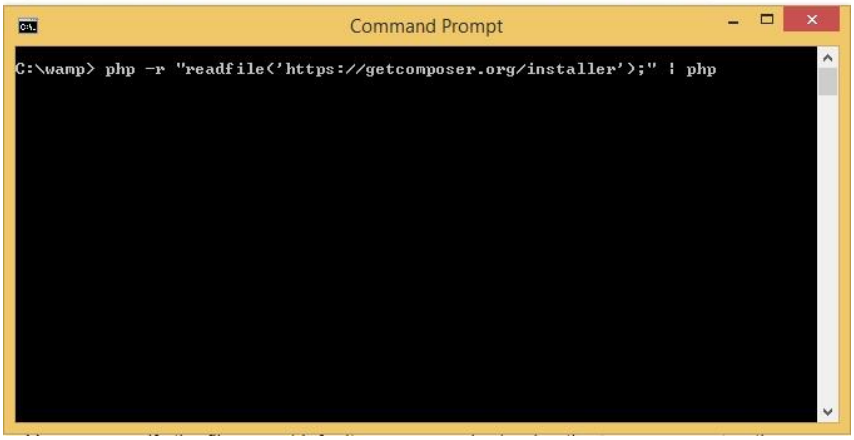
Ανοίγουμε το αρχείο php.ini και σιγουρευόμαστε ότι το SSL είναι ενεργό. Επειδή στην αρχή της γραμμής που γράφει για το open SSL δεν υπάρχει το αγγλικό ερωτηματικό (;) σημαίνει ότι η γραμμή δεν είναι σχόλιο.



Εικόνα 12 Έλεγχος για Open ssl στο αρχείο php.ini

Για την εγκατάσταση του Composer ανοίγουμε το cmd και γράφουμε την παρακάτω εντολή και πατάμε Enter.

```
php -r "readfile('https://getcomposer.org/installer');" | php
```



Εικόνα 13 Εγκατάσταση του Composer

Αφού εγκαταστήσαμε τον Composer θα εγκαταστήσουμε το PHP Framework Laravel. Ανοίγουμε το Command (cmd) κατευθυνόμαστε στον φάκελο του WampServer, γράφουμε την παρακάτω εντολή, πατάμε Enter και εγκαθίσταται το Laravel.

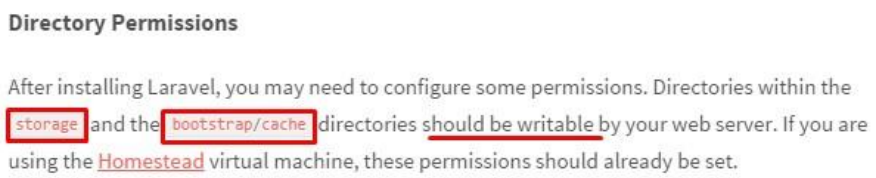
Via Composer Create-Project

You may also install Laravel by issuing the Composer `create-project` command in your terminal:

```
composer create-project laravel/laravel --prefer-dist
```

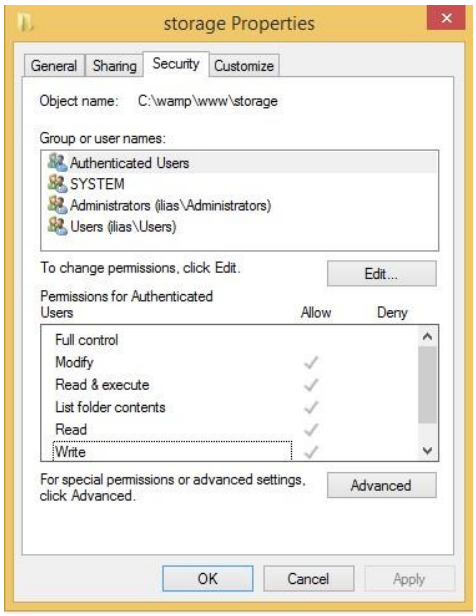
Εικόνα 14 Εγκατάσταση Laravel Framework

Στη συνέχεια για να δώσουμε δικαιώματα εγγραφής στους φακέλους storage & bootstrap/cache πρέπει να πάμε στο http://laravel.com/docs/5.1#configuration.



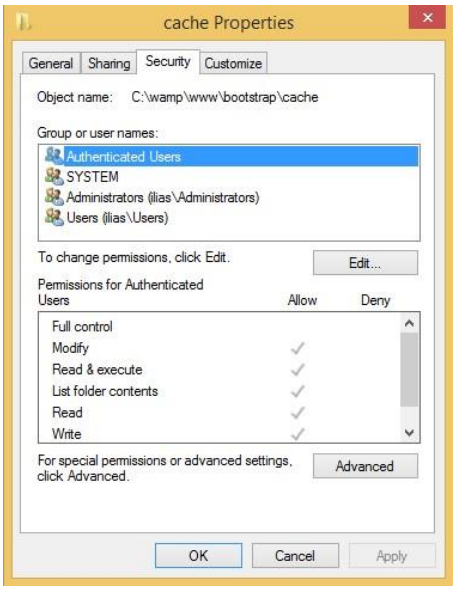
Εικόνα 15 Δικαιώματα εγγραφής σε φακέλους storage & bootstrap/cache

Έπειτα οδηγούμαστε στον φάκελο storage που βρίσκεται μέσα στον φάκελο www. Wamp ->www-> storage ->Δεξί κλικ -> Ιδιότητες -> Ασφάλεια -> Write √.



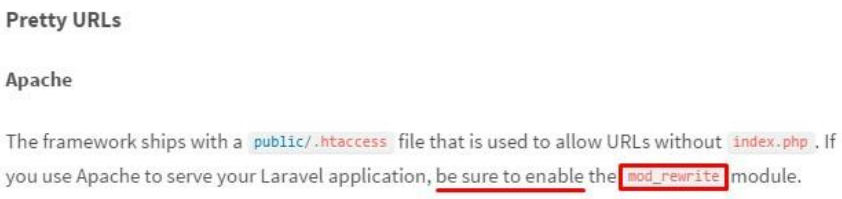
Εικόνα 16 Φάκελος storage

Αφού ελέγξαμε ότι έχουμε δικαιώματα εγγραφής στον φάκελο storage, έτσι θα ελέγξουμε ότι έχουμε τα ίδια δικαιώματα και στον φάκελο bootstrap/cache. Wamp -> www -> Bootstrap -> Cache -> Δεξί κλικ -> Ιδιότητες -> Ασφάλεια -> Write √.



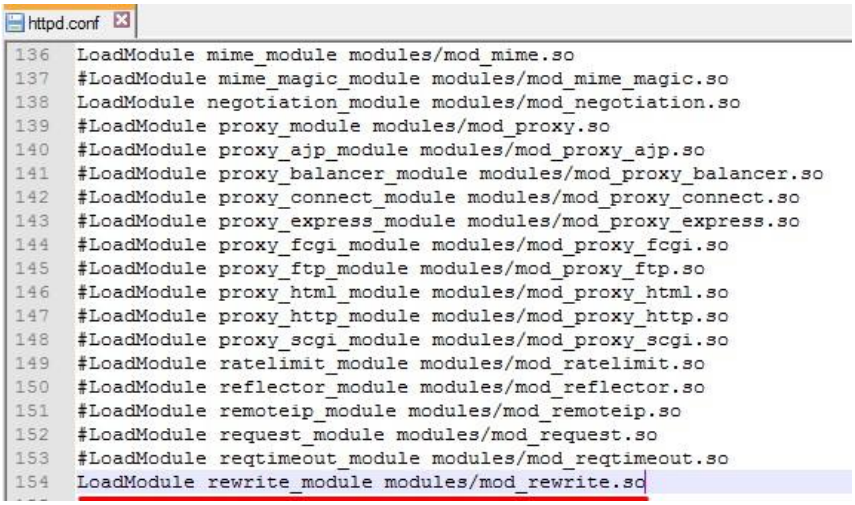
Εικόνα 17 Φάκελος bootstrap/cache

Επόμενο βήμα είναι να ενεργοποιήσουμε το module Rewrite του Apache http://laravel.com/docs/5.1#basic-configuration.



Εικόνα 18 Ενεργοποίηση module Rewrite του Apache

Ανοίγουμε το httpd.conf για να ενεργοποιήσουμε το module Rewrite. Το mod_rewrite είναι ενεργό, διότι λείπει από τη γραμμή η δίεση (#). Στην περίπτωση που η γραμμή αυτή είχε τη δίεση θα ήταν σχόλιο.



Εικόνα 19 Ενεργοποίηση module Rewrite του Apache στο αρχείο httpd.conf

Αφού έχουν ολοκληρωθεί η εγκατάσταση του Laravel Framework και οι ρυθμίσεις μένει να ανοίξουμε τον Web Browser και να πατήσουμε localhost.

Εικόνα 20 Σελίδα localhost μετά την επιτυχή εγκατάσταση του Laravel Framework

4.3 Routing

4.3.1 Βασικά στο Routing

Το Routing είναι ένα από τα κυριότερα συστατικά μιας διαδικτυακής εφαρμογής. Κάθε φορά που κάνουμε ένα αίτημα για ένα URL, το route είναι αυτό που εμφανίζεται στον χρήστη.

Τα routes για τις διαδικτυακές εφαρμογές μας τα γράφουμε στο αρχείο routes.php το οποίο βρίσκεται στην παρακάτω διαδρομή wamp/app/Http/routes.php.

4.3.2 Παράμετροι Routing

Υπάρχουν δύο κατηγορίες route παραμέτρων. Στην πρώτη κατηγορία ανήκει η Βασική route παράμετρος και στη δεύτερη είναι η Προαιρετική (optional) route παράμετρος.

Στην περίπτωση που έχουμε πολλά παρόμοια routes μπορούμε να δημιουργήσουμε ένα γενικό route. Αυτό γίνεται θέτοντας την παράμετρο ως μεταβλητή στη συνάρτηση και να έχουμε ένα συγκεκριμένο αποτέλεσμα. Οπότε τη μεταβλητή που βάλαμε ως είσοδο την έχουμε ως έξοδο στο return.

Μια route παράμετρος γίνεται προαιρετική (optional) προσθέτοντας στο τέλος της το αγγλικό ερωτηματικό (?) και καθορίζοντας την προεπιλεγμένη τιμή της.

4.4 Views

4.4.1 Βασικά στο Blade Templating Engine

Τα Views περιέχουν κώδικα γραμμένο σε HTML (Hyper Text Markup Language) ο οποίος ευθύνεται για την εμφάνιση της

διαδικτυακής εφαρμογής. Τα Views βρίσκονται στον φάκελο app/resources/views.

Το Blade Templating Engine είναι μια μηχανή προτύπων που παρέχει το Laravel Framework και έρχεται να αντικαταστήσει τη χρήση PHP κώδικα. Χρησιμοποιώντας το Blade Templating Engine έχουμε στη διάθεσή μας τις γνωστές Δομές Επιλογής (if/else/elseif/unless) και Δομές Επανάληψης (for/foreach/while). Τέλος, είναι πολύ σημαντικό στα αρχεία που χρησιμοποιούμε τη σύνταξη του Blade να έχουν την κατάληξη .blade.php.

Η Blade σύνταξη είναι η εξής: `{{ Welcome to Laravel }}`

4.4.2 Passing Data to Views (Routes & Views)

Σε αυτή την ενότητα θα δούμε τρείς τρόπους με τους οποίους μπορούμε να εμφανίσουμε μια μεταβλητή από το αρχείο routes.php στο views.php.

Ο πρώτος τρόπος είναι με χρήση της μεθόδου with(), η οποία παίρνει ως πρώτη παράμετρο τη μεταβλητή και ως δεύτερη την τιμή της μεταβλητής που θέλουμε να εμφανίσουμε. Είναι σημαντικό να τονίσουμε ότι η πρώτη παράμετρος της μεθόδου πρέπει να έχει το ίδιο όνομα με τη μεταβλητή στο View.

Ο δεύτερος τρόπος είναι με τη χρήση της μεθόδου view(), η οποία θα έχει ως πρώτη παράμετρο το welcome view και ως δεύτερη παράμετρο βάζουμε έναν πίνακα και τη μεταβλητή του που υπάρχει στο view.

Τέλος, έχουμε τη δυνατότητα να χρησιμοποιήσουμε μια μέθοδο που ονομάζεται Magic Method και έχει ως παράμετρο τη μεταβλητή που θέλουμε να εμφανίσουμε σε κάθε περίπτωση.

4.4.3 Δομές Επιλογής και Δομές Επανάληψης

Το Laravel είναι ένα PHP Framework γεγονός που δηλώνει ότι περιέχει τις δομές επιλογής, που συναντάμε και στην PHP if, elseif και else με τη μόνη προσθήκη της unless, αλλά και τις δομές επανάληψης for, foreach, while.

Σύνταξη των Δομών Επιλογής

```
@if(συνθήκη)
Αποτέλεσμα που επιστρέφεται αν ισχύει η συνθήκη
@endif

@elseif(συνθήκη)
Αποτέλεσμα που επιστρέφεται αν ισχύει η συνθήκη
@else
Αποτέλεσμα που επιστρέφεται αν ισχύει η συνθήκη
@endif

@unless (συνθήκη)
Αποτέλεσμα που επιστρέφεται αν δεν ισχύει η συνθήκη
@endunless
```

Σύνταξη των Δομών Επανάληψης

```
@for (δήλωση 1; δήλωση 2; δήλωση 3)
Κώδικας που εκτελείται
@endfor

@foreach (a as b)
Κώδικας που εκτελείται
@endforeach
```

```
@while (συνθήκη)
Κώδικας που εκτελείται
@endwhile
```

4.4.4 Επεκτείνοντας τα Views

Σε αυτή την ενότητα θα μελετήσουμε ένα από τα πιο σημαντικά και χρήσιμα συστατικά για την εμφάνιση μιας διαδικτυακής εφαρμογής.

Με τη χρήση του Blade Templating Engine έχουμε τη δυνατότητα να επεκτείνουμε Views από άλλα Views. Αυτό γίνεται δημιουργώντας μέσα στον φάκελο που έχουμε τα Views μας, ένα νέο φάκελο με όνομα Layouts. Εν συνεχεία μέσα στον φάκελο Layouts που δημιουργήσαμε, δημιουργούμε ένα νέο View με όνομα master.blade.php. Το master.blade.php είναι το parent view το οποίο κληρονομούν τα child views, όπως το contact.blade.php, about.blade.php, κ.α.

Παρακάτω θα αναλύσουμε κάποιες από τις βασικότερες «εντολές» που χρησιμοποιούμε στα Views.

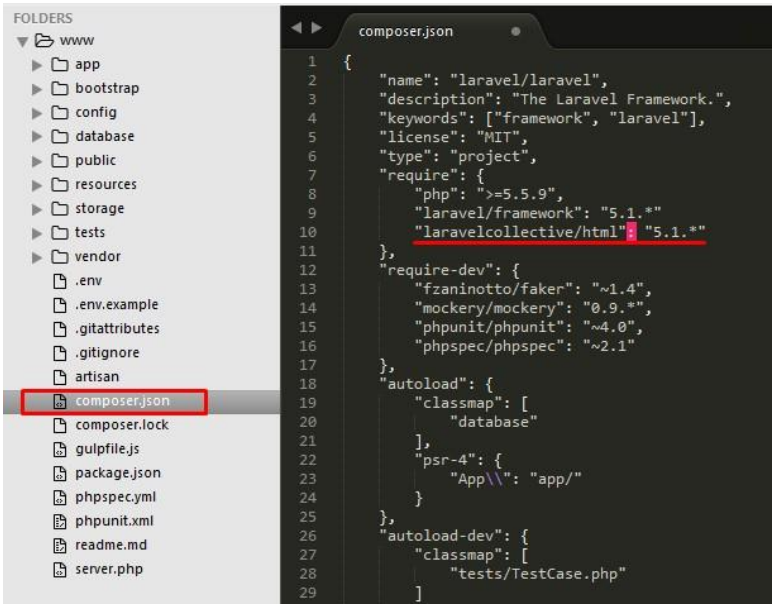
- @section: Τη χρησιμοποιούμε στο child view για να επεκτείνουμε το layout.
- @yield: Τη χρησιμοποιούμε στο parent view για να εμφανίσουμε τα αντίστοιχα περιεχόμενα του child view.
- @extends: Τη χρησιμοποιούμε στο child view διότι επεκτείνουμε το layout από το parent view. Μέσα στο @extends ορίζεται το layout που θα κληρονομήσει το child view.
- @parent: Τη χρησιμοποιούμε στο child view καθώς όταν ένα @section έχει οριστεί στο parent view και το ορίσουμε ξανά στο child view, το περιεχόμενο του @section στο child view υπερκαλύπτει το περιεχόμενο του @section στο parent view.

4.4.5 Εγκατάσταση του πακέτου HTML

Το πακέτο HTML το εγκαθιστούμε χρησιμοποιώντας τον Composer. Με το πακέτο HTML δημιουργούμε φόρμες με απλούστερο τρόπο από τον κλασσικό τρόπο δημιουργίας. Για την εγκατάστασή του θα ακολουθήσουμε τα παρακάτω βήματα:

Αρχικά, για να το εγκαταστήσουμε πηγαίνουμε στο official site https://laravelcollective.com/docs/5.1/html.

Στη συνέχεια εισάγουμε τη γραμμή “laravelcollective/html”:”5.1.*” στο αρχείο composer.json που βρίσκεται στον φάκελο www.



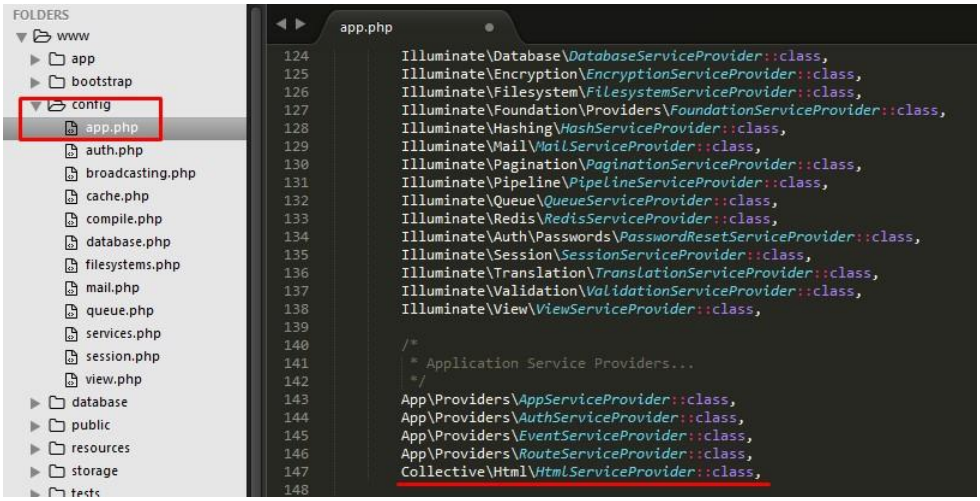
Εικόνα 21 Αρχείο composer.json

Το επόμενο βήμα είναι να κάνουμε μια ενημέρωση στον Composer ανοίγοντας το cmd και πηγαίνοντας στον φάκελο του project μας.

Εικόνα 22 Ενημέρωση του Composer

Πλέον μένουν δυο τελευταίες ρυθμίσεις και το πακέτο HTML είναι έτοιμο.

Πηγαίνουμε στο αρχείο app.php (www/config/app.php) για να προσθέσουμε την ακόλουθη γραμμή στους Providers Collective\Html\HtmlServiceProvider::class.

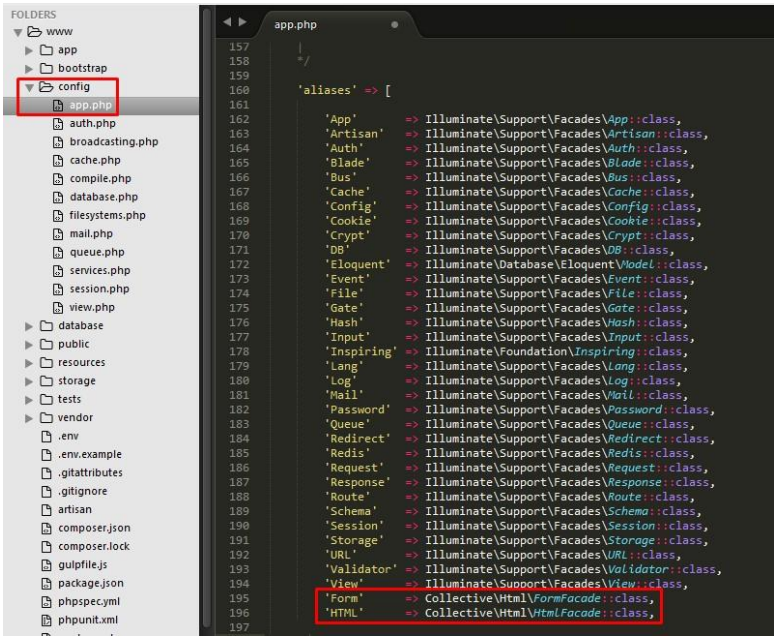


Εικόνα 23 Ρυθμίσεις αρχείο app.php (1)

Στο ίδιο αρχείο προσθέτουμε στα aliases τις γραμμές

‘Form’ => Collective\Html\FormFacade::class

‘HTML’ => Collective\HTML\HtmlFacade::class



Εικόνα 24 Ρυθμίσεις αρχείο app.php (2)

4.4.6 HTML Builder

Το HTML Builder είναι ένα εργαλείο που περιέχει μεθόδους αντίστοιχες με τα tags της HTML. Χρησιμοποιώντας το εργαλείο αυτό κερδίζουμε χρόνο και γραμμές κώδικα στα projects μας. Εμείς θα ασχοληθούμε με τέσσερεις από τις μεθόδους που μας παρέχει. Θα δούμε πως εισάγουμε ένα CSS αρχείο, ένα JavaScript αρχείο, πως δημιουργούμε ένα σύνδεσμο και μια μη αριθμημένη λίστα.

4.5 Forms – Φόρμες

4.5.1 GET & POST Requests

Όταν θέλουμε να δημιουργήσουμε σελίδες που να υπάρχει διάδραση μεταξύ χρήστη και σελίδας ή θέλουμε να συλλέξουμε πληροφορίες από τον χρήστη, χρησιμοποιούμε φόρμες. Για τη δημιουργία μιας φόρμας είναι απαραίτητη η χρήση tags και χαρακτηριστικών (attributes) τα οποία θα αναλύσουμε παρακάτω.

- <form>: Το <form> ορίζει την αρχή μιας φόρμας.
  - action: Στο attribute action ορίζουμε τη διεύθυνση που θα αποσταλούν τα δεδομένα της φόρμας μόλις ο χρήστης πατήσει το κουμπί Submit.
  - Method: Στο attribute method γράφουμε τη μέθοδο που θα χρησιμοποιήσουμε για την αποστολή των δεδομένων (GET/POST).
- </form>: Το </form> ορίζει το τέλος μιας λίστας.
- <input type='text'>: Δημιουργεί ένα πεδίο εισαγωγής κειμένου.
  - name: Το name είναι ένα χαρακτηριστικό που ορίζει ένα όνομα για το στοιχείο.
- <input type='password'>: Δημιουργεί ένα πεδίο εισαγωγής συνθηματικού.
  - name: Το name είναι ένα χαρακτηριστικό που ορίζει ένα όνομα για το στοιχείο.
- <input type='submit'>: Δημιουργεί ένα κουμπί το οποίο όταν πατηθεί από το χρήστη αποστέλλει τα δεδομένα της φόρμας στην προκαθορισμένη διεύθυνση.
  - value: Το value καθορίζει το κείμενο που θα εμφανίζεται στο κουμπί.
- <label>: Το <label> δημιουργεί μια ετικέτα για ένα πεδίο της λίστας.

Στις φόρμες στο χαρακτηριστικό method ορίζουμε την επιθυμητή μέθοδο υποβολής ενός αιτήματος μέσω του πρωτοκόλλου HTTP.

- Μέθοδος GET:
  - Χρησιμοποιείται όταν έχουμε λίγα δεδομένα για αποστολή.
  - Είναι μια μη ασφαλής μέθοδος επειδή τα δεδομένα προσαρτώνται στο URL κατά την υποβολή της φόρμας.
  - Το αίτημα GET δεν προκαλεί μεταβολές στον Server.
- Μέθοδος POST:
  - Τα δεδομένα που αποστέλλουμε δεν φαίνονται στο URL.
  - Χρησιμοποιώντας τη μέθοδο POST αποστέλλουμε τα δεδομένα στον Server για να τα επεξεργαστεί.

4.5.2 Form Builder

Το Form Builder είναι ένα εργαλείο που περιέχει μεθόδους αντίστοιχες με τα tags και τα χαρακτηριστικά δημιουργίας φορμών. Χρησιμοποιώντας το εργαλείο αυτό κερδίζουμε χρόνο και γραμμές κώδικα στα projects μας. Εμείς θα ασχοληθούμε με έξι από τις μεθόδους που μας παρέχει. Θα δούμε πως ανοίγουμε μια φόρμα, πως δημιουργούμε μια ετικέτα, ένα πεδίο κειμένου, ένα πεδίο επιλογής, radio button και το τελικό κουμπί αποστολής των δεδομένων.

4.6 Database – Βάσεις Δεδομένων

4.6.1 Σύνδεση Βάσης Δεδομένων με το Laravel

Στην ενότητα αυτή θα δούμε τρείς τρόπους που μπορούμε να δημιουργήσουμε μια Βάση Δεδομένων και πως θα τη συνδέσουμε με το Laravel. Ο πρώτος τρόπος είναι να χρησιμοποιήσουμε το phpMyAdmin, ο δεύτερος είναι να δημιουργήσουμε τη Βάση Δεδομένων μέσω του command με εντολές και ο τρίτος είναι χρησιμοποιώντας το MySQL Query Browser.

Ο κώδικας για τη σύνδεση της Βάσης με το Laravel είναι ο παρακάτω.

```
$name = DB::Connection()->getDatabaseName();
```

4.6.2 SQL Ερωτήματα

Σε αυτή την ενότητα θα εκτελέσουμε κάποια SQL ερωτήματα χρησιμοποιώντας τις γνωστές εντολές. Περισσότερα παραδείγματα θα δούμε στο παράρτημα 1.

4.6.3 Query Builder

Το Query Builder είναι ένα εργαλείο που περιέχει μεθόδους αντίστοιχες με τις εντολές της MySQL. Χρησιμοποιώντας το εργαλείο αυτό κερδίζουμε χρόνο στα projects μας. Εμείς θα ασχοληθούμε με έξι από τις μεθόδους που μας παρέχει. Θα δούμε πως ανοίγουμε μια φόρμα, πως δημιουργούμε μια ετικέτα, ένα πεδίο κειμένου, ένα πεδίο επιλογής, radio button και το τελικό κουμπί αποστολής των δεδομένων.

4.7 1ο Project – Website με δωρεές

Αρχικά θα δημιουργήσουμε τη Βάση Δεδομένων της ιστοσελίδας μας και στη συνέχεια μια φόρμα μέσω της οποίας ο χρήστης θα προσθέτει κάποιο project και στη συνέχεια θα εμφανίζεται το project.

Στο τελευταίο στάδιο υλοποίησης του πρώτου project δημιουργήσαμε ένα επιπλέον πεδίο επιλογής ποσού, ένα κουμπί για την πραγματοποίηση της δωρεάς και έναν μετρητή ο οποίος αυξάνει το ποσό της δωρεάς κάθε project. Με την ολοκλήρωση της δωρεάς εμφανίζεται το id του project που έγινε η δωρεά και το ποσό της δωρεάς.

4.8 MVC & REST Theory

4.8.1 MVC

Το Laravel Framework είναι βασισμένο στο αρχιτεκτονικό πρότυπο MVC (Model - View - Controller) καθώς τα Models, τα Views και οι Controllers βρίσκονται σε ξεχωριστά αρχεία και σε ξεχωριστούς καταλόγους. Στο πρότυπο MVC διαχωρίζεται η λογική της εφαρμογής από την λογική της παρουσίασης που συνδέεται με ένα γραφικό περιβάλλον χρήστη (Graphical User Interface).

Το Model είναι μόνιμο χαρακτηριστικό και αποθηκεύεται εκτός της εφαρμογής, συνήθως σε μια Βάση Δεδομένων. Στο Model κατασκευάζουμε πολύπλοκους αλγόριθμους, προγραμματίζουμε σύνθετες πράξεις και πραγματοποιούμε διάφορες ενέργειες σε Βάσεις Δεδομένων (ενημέρωση, εισαγωγή, κλπ..).

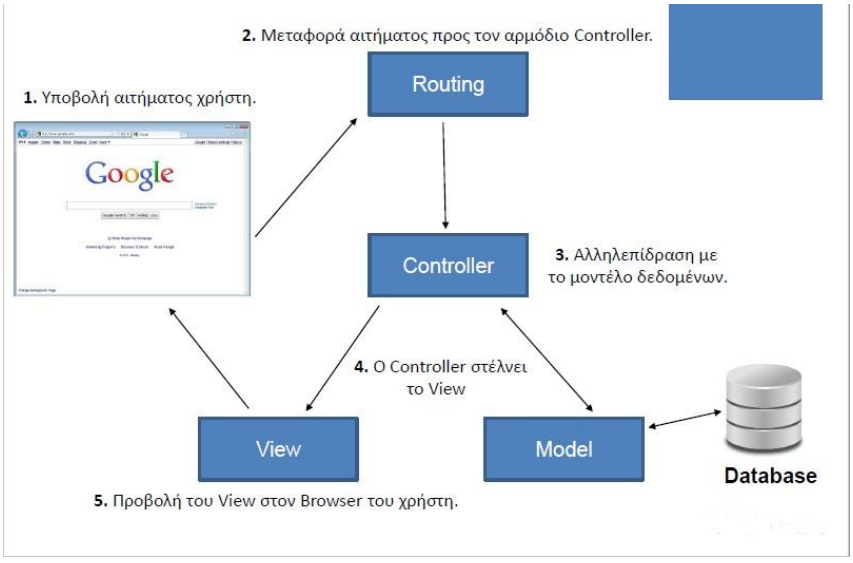
Το συστατικό στο οποίο ασχολούμαστε με τον τρόπο παρουσίασης των πληροφοριών στο χρήστη είναι το View. Το επίπεδο του view είναι υπεύθυνο για τη δημιουργία ενός περιβάλλοντος χρήστη (User Interface) βασισμένου στα δεδομένα του Model.

Το συστατικό το οποίο βρίσκεται μεταξύ του Model και του View και τα συντονίζει είναι ο Controller. Ο Controller είναι υπεύθυνος για την επεξεργασία των αιτημάτων, για το ποια ενέργεια να πραγματοποιήσει. Για παράδειγμα, όταν δεχθεί ένα αίτημα για το οποίο απαιτούνται να γίνουν σύνθετες διεργασίες επικοινωνεί αρχικά με το Model και στη συνέχεια προωθεί το αποτέλεσμα στο View για να το εμφανίσει.

Παρακάτω θα παρουσιάσουμε τη λειτουργία ενός αιτήματος που υποβάλει ο χρήστης.

Αρχικά, ο Browser στέλνει ένα αίτημα, το οποίο λαμβάνει ένας web server και το μεταφέρει στη Routing Engine του Laravel. Στη συνέχεια, ο router του Laravel λαμβάνει το αίτημα και το προωθεί στον κατάλληλο Controller.

Μόλις ο Controller λάβει το αίτημα από το Laravel Router, σε ορισμένες περιπτώσεις απαντάει αμέσως με ένα view, το οποίο είναι ένα πρότυπο που μετατρέπεται σε HTML και αποστέλλεται πίσω στον Browser. Τις περισσότερες όμως φορές, ο Controller πρώτα αλληλοεπιδρά με ένα στοιχείο της web εφαρμογής που είναι υπεύθυνο για την επικοινωνία με τη Βάση Δεδομένων (model) και στη συνέχεια στέλνει ένα view που επιστρέφει πίσω στον Browser ως ολοκληρωμένη Web Σελίδα (HTML, CSS, εικόνες).



Εικόνα 25 Λειτουργία user request

4.8.2 REST

Η θεμελιώδης έννοια σε κάθε RESTful web εφαρμογή είναι οι πόροι (resources). Ένας πόρος είναι μια αφηρημένη αναπαράσταση ενός αντικειμένου το οποίο σχετίζεται με την web εφαρμογή.

Ένα social media website σχετίζεται με πολλούς χρήστες (χρήστης=αντικείμενο) οι οποίοι αναπαρίστανται ως user resource.

Ένα website μιας βιβλιοθήκης σχετίζεται με πολλά βιβλία (βιβλίο=αντικείμενο) τα οποία αναπαρίστανται ως book resource.

Η λέξη REST είναι το ακρωνύμιο των λέξεων Representational State Transfer. Η αρχιτεκτονική στην οποία βασίζονται οι web εφαρμογές και χρησιμοποιείται για την κατασκευή web εφαρμογών είναι η REST. Οι εφαρμογές που στηρίζονται την αρχιτεκτονική REST ονομάζονται RESTful APIs (Application Programming Interface). Τα RESTful συστήματα συνήθως επικοινωνούν μέσω του HTTP πρωτοκόλλου χρησιμοποιώντας τα HTTP verbs. Τα HTTP verbs είναι το POST για τη δημιουργία νέου χρήστη, το GET για την ανάκτηση του χρήστη, το PUT για την ενημέρωση των στοιχείων του υπάρχοντος χρήστη και τέλος το DELETE για τη διαγραφή του υπάρχοντος χρήστη.

4.9 Controllers – Ελεγκτές

4.9.1 Basic Controllers

Στην περίπτωση που έχουμε ένα πολύ μεγάλο website με πολλά URLs, που πρέπει να χειριστούμε και πολλές φόρμες που οδηγούν σε άλλα URLs. Αυτή η υπόθεση συνεπάγεται να έχουμε ένα πολύ μεγάλο, σε έκταση, routes.php το οποίο θα χειριστούμε δύσκολα. Εδώ έρχονται οι Controllers για να λύσουν το πρόβλημα. Οι Controllers πρέπει να επεκτείνουν την κλάση Controller και βρίσκονται στο App\Http\Controllers. Τέλος, το όνομα του αρχείου του Controller πρέπει να είναι ίδιο με το όνομα της κλάσης.

4.9.2 RESTful Controllers

Στους RESTful Cotrollers τα ονόματα που δίνουμε στις μεθόδους μας αποτελούνται από δύο τμήματα. Το 1 ο τμήμα είναι το HTTP verb στο οποίο η μέθοδος θα απαντάει και το 2 ο τμήμα είναι το URL στο οποίο θα απαντάει η συνάρτηση. Παραδείγματος χάριν η μέθοδος getFilms() εκτελείται μόνο αν γίνεται αίτημα για το URL localhost/portfolio/films χρησιμοποιώντας τη μέθοδο GET.

4.9.3 Resource Controllers

Με τους Resource Controllers μπορούμε να δημιουργήσουμε έναν Controller να χειρίζεται τα HTTP αιτήματα για τους πόρους της εφαρμογής μας. Το 1 ο όρισμα της μεθόδου resource είναι το URL και το 2 ο όρισμα είναι το όνομα του Controller μέσω του οποίου θα χειριστούμε τα routes.

Οι Resource Controllers πρέπει να περιέχουν τις παρακάτω μεθόδους:

- index (): : Εκτελείται όταν γίνεται ένα GET Request για το /users και μας εμφανίζει όλους τους χρήστες από τη ΒΔ.
- create (): Εκτελείται όταν γίνεται ένα GET Request για το /users/create και μας εμφανίζει τη φόρμα δημιουργίας νέου χρήστη.
- store (Request $request): Εκτελείται όταν γίνεται ένα POST Request για το /users και αποθηκεύει ένα νέο χρήστη στη ΒΔ.
- show ($id): Εκτελείται όταν γίνεται ένα GET Request για το /users/user1 και μας εμφανίζει τον user1
- edit ($id): Εκτελείται όταν γίνεται ένα GET Request για το /users/user1/edit και μας εμφανίζει τη φόρμα για ενημέρωση ή διαγραφή των στοιχείων του user1.
- update (Request $request, $id): Εκτελείται όταν γίνεται ένα PUT Request και ενημερώνονται τα στοιχεία του user1 στη ΒΔ.

- destroy ($id): : Εκτελείται όταν γίνεται ένα DELETE Request και διαγράφει τον user1 από τη ΒΔ.

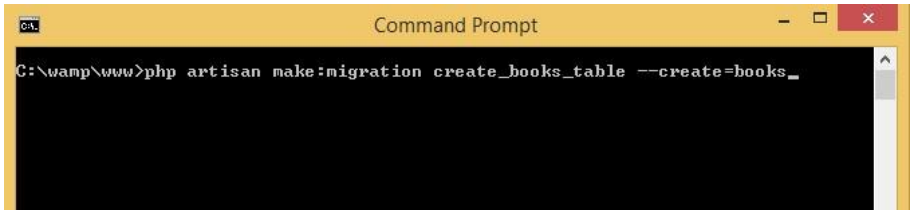
4.10 Προχωρημένα Θέματα Βάσεων Δεδομένων

4.10.1 Migrations & Schema Builder

Με τα migrations δημιουργούμε με απλό τρόπο πίνακες στη Βάση Δεδομένων.

Για να δημιουργήσουμε τον πίνακα books πηγαίνουμε στο command (cmd) και γράφουμε:

```
php artisan make:migration create_books_table -- create=books
```



Εικόνα 26 Δημιουργία πίνακα books με χρήση migrations

Για να κάνουμε migrate τη Βάση:

```
php artisan migrate
```

Εικόνα 27 Τρόπος που κάνουμε migrate τη Βάση Δεδομένων

Για να διαγράψουμε τον πίνακα books που δημιουργήσαμε με το migration πηγαίνουμε στο command (cmd) και γράφουμε:

```
php artisan migration:rollback
```

Εικόνα 28 Διαγραφή πίνακα books

Με την κλάση Schema του Laravel μπορούμε να επεξεργαστούμε τη Βάση Δεδομένων. Έχουμε δυνατότητα δημιουργίας και διαγραφής πινάκων, προσθήκη και επεξεργασία των στηλών ενός πίνακα, προσθήκη και επεξεργασία ευρετηρίων (indexes), χειρισμό των ξένων κλειδιών, κλπ.

Υπάρχουν κάποιες βασικές μέθοδοι τις οποίες θα αναλύσουμε.

- public function up(): Στη μέθοδο up () προσθέτουμε πίνακες ή στήλες ή ευρετήρια στη ΒΔ και εκτελείται όταν τρέχουμε το migration.
- public function down(): Στη μέθοδο down () αντιστρέφουμε αυτά που εκτελούμε στη μέθοδο up () και εκτελείται όταν κάνουμε rollback το migration.
- Schema::create('books', function (Blueprint $table): Καλούμε τη μέθοδο create () της κλάσης Schema. Η μέθοδος create () είναι υπεύθυνη για τη δημιουργία του πίνακα books.
- Schema::drop('books'): Διαγράφουμε τον πίνακα books από τη ΒΔ.
- $table->increments('id'): Δημιουργούμε ένα πεδίο που είναι AUTO_INCREMENT.
- $table->string('title'): Δημιουργούμε ένα πεδίο τύπου string για τίτλο των βιβλίων.
- $table->string('author'): Δημιουργούμε ένα πεδίο τύπου string για τον συγγραφέα.

- $text->text('summary')->nullable(): Δημιουργούμε ένα πεδίο, τύπου text για την περίληψη, το οποίο είναι null. By default τα πεδία που δημιουργούμε είναι not_null.
- $text->dropColumn('created_at'): Διαγράφουμε τη στήλη created_at που περιέχει την ημερομηνία και ώρα δημιουργίας.
- $text->dropColumn('updated_at'): Διαγράφουμε τη στήλη updated_at που περιέχει την ημερομηνία και ώρα ενημέρωσης.

4.10.2 Seeding

Για να γεμίσουμε έναν πίνακα με τιμές, θα δημιουργήσουμε μια κλάση η οποία θα επεκτείνει την κλάση Seeder και θα γεμίζει με τιμές τον πίνακα.

Το αρχείο DatabaseSeeder.php με το οποίο θα δουλέψουμε είναι App\Database\Seeds\DatabaseSeeder.php.

4.10.3 Eloquent ORM

Με το Eloquent δημιουργείται μια γέφυρα μεταξύ του Model (π.χ. Article Model) και του πίνακα στη ΒΔ (π.χ. articles table). Eloquent ORM είναι το ακρωνύμιο των λέξεων Object Relational Mapping και χρησιμοποιείται για να αλληλοεπιδρά με σχεσιακές Βάσεις Δεδομένων. Όταν χρησιμοποιούμε το Eloquent, τότε το όνομα του Model θα πρέπει να είναι στον ενικό αριθμό και το όνομα του πίνακα στον πληθυντικό αριθμό.

Για να δημιουργήσουμε ένα model Article πηγαίνουμε στο command (cmd) και γράφουμε:

```
php artisan make:model Article
```

Εικόνα 29 Δημιουργία Article model

4.10.4 Συσχετίσεις

Οι συσχετίσεις που υπάρχουν στο Laravel είναι οι έξι που ακολουθούν.

1. Ένα προς ένα (One to One)
2. Ένα προς πολλά (One to Many)
3. Πολλά προς πολλά (Many to Many)
4. Περιέχει πολλά (Has Many Through)
5. Πολυμορφικές συσχετίσεις (Polymorphic Relations)
6. Πολλές προς πολλές πολυμορφικές συσχετίσεις (Many to Many Polymorphic Relations)

4.11 Miscellaneous in Laravel – Διάφορα στο Laravel

4.11.1 Form Validation

Για να επικυρώσουμε τα στοιχεία που εισάγει ο χρήστης ορίζουμε ορισμένους κανόνες. Το επόμενο βήμα μετά τον ορισμό των επιθυμητών κανόνων είναι να επικυρώσουμε τα στοιχεία.

4.11.2 Artisan File & 404 Errors

Όταν επιθυμούμε να έχουμε αλληλοεπιδράσεις με την διαδικτυακή εφαρμογή που υλοποιούμε, τότε χρησιμοποιούμε το artisan file. Οι εντολές που έχει το artisan είναι πολλές και για να τις δούμε αρκεί να γράψουμε php artisan list.

Όσον αφορά τις error pages μπορούμε να δημιουργήσουμε custom templates για τις 404 σελίδες.

4.12 2ο Project – Πλατφόρμα με Registration & Login System

Το δεύτερο project περιλαμβάνει όλα όσα μάθαμε κατά τη διάρκεια των video διαλέξεων. Πρώτα οργανώνουμε τους πόρους της εφαρμογής μας και στη συνέχεια δημιουργούμε μια φόρμα εγγραφής χρήστη. Αφού εγγραφεί ο χρήστης στην πλατφόρμα δημιουργούμε τη σελίδα του προφίλ του, μέσω της οποίας ο κάθε χρήστης μπορεί να επεξεργαστεί τα στοιχεία του. Τέλος, κατά την εισαγωγή των απαραίτητων στοιχείων για το login πραγματοποιείται επικύρωση των στοιχείων του χρήστη.

Παράρτημα 1

Παραδείγματα Ενότητας 4.3.1

```
Route::get('/', function () {
      return 'Εισαγωγή στο Laravel';
});

Route::get('progrlang', function () {
      return 'Γλώσσες Προγραμματισμού..';
});

Route::get('progrlang/php', function () {
      return 'Η PHP είναι γλώσσα προγραματισμού!';
});
```

Παραδείγματα Ενότητας 4.3.2

```
Route::get('/', function () {
      return 'Εισαγωγή στο Laravel';
});

Route::get('progrlang/php', function () {
      return 'Η PHP είναι γλώσσα προγραματισμού!';
});
```

```
Route::get('progrlang/{web?}', function ($web = null) {
      if ($web == null)
      {
            return 'Γλώσσες Προγραμματισμού...';
      }
            return ' Η ' .$web. ' είναι γλώσσα προγραματισμού
web.';
});

Route::get('articles/{title?}', function ($title = null) {
      if ($title == null)
      {
            return 'Αυτή είναι μια λίστα με όλα τα άρθρα....';
      }
            return 'Αυτό είναι το κυρίως σώμα του άρθρου με
τίτλο:' .$title;
});

Route::get('a/{par1}/b/{par2?}', function ($par1, $par2 =
'default') {
      return 'Η Παράμετρος 1 = ' .$par1. ' και η Παράμετρος 2 =
' .$par2;
});
```

Παραδείγματα Ενότητας 4.4.1

Αρχείο routes.php

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

Αρχείο welcome.blade.php

```
<html>
    <body>
    <?php $var = 'hello!!'; ?>
    <h1>{{ $var }}</h1>
    </body>
</html>
```

Παραδείγματα Ενότητας 4.4.2

Αρχείο routes.php

```
Route::get('/', function () {
      // 1os tropos na emfanisoume mia metavliti
      $pet1 = 'Dog';
      return view('welcome')->with('var', $pet1);

      // 2os tropos na emfanisoume mia metavliti
      $pet1 = 'Dog';
      return view('welcome', ['var' => $pet1]);
});

Route::get('/', function () {
      // 1os tropos na emfanisoume perissoteres metavlites
      $pet1 = 'Dog';
      $pet2 = 'Cat';
      return  view('welcome')->with('var',  $pet1)->with('var2',
$pet2);
```

```
    // 2os tropos na emfanisoume perissoteres metavlites
    return view('welcome', ['var' => $pet1, 'var2' => $pet2]);

    // 3os tropos na emfanisoume perissoteres metavlites
    return view('welcome', compact('pet1', 'pet2');

    // 4os tropos na emfanisoume perissoteres metavlites
    $list = ['a', 'b', 'c'];
    return view('welcome', ['list' => $list]);

    //Magic Method
    return view('welcome')->withTest($pet1);
});
```

Αρχείο welcome.blade.php

```
<html>
    <body>
    <!--1os tropos na emfanisoume mia metavliti-->
    <h1>{{ $var }}</h1>

    <!-- 1os, 2os, 3os tropos na emfanisoume perissoteres
metavlites -->
    <h1>{{ $var }}</h1>
    <h1>{{ $var2 }}</h1>

    <!--4os tropos na emfanisoume perissoteres metavlites-->
    <h1>{{ $list[0] }}</h1>
    <h1>{{ $list[1] }}</h1>
    <h1>{{ $list[2] }}</h1>
```

```
      <!-- Magic Method gia na emfanisoume mia metavliti -->
      <h1>{{ $test }}</h1>
      </body>
</html>
```

Παραδείγματα Ενότητας 4.4.3

Αρχείο routes.php

```
Route::get('/', function () {
      $lang = ['PHP', 'Javascript', 'Java'];
      return view('welcome')->withWeb($lang);
});
```

Αρχείο welcome.blade.php

```
<html>
    <body>
      @if (count($web) === 1)
      Ξέρω μια Γλώσσα Προγραμματισμού web: {{ $web[0] }}
            @elseif (count($web) > 1)
                  @foreach ($web as $web)
                  Ξέρω την Γλώσσα Προγραμματισμού web: {{ $web }}
                  <br> <br>
                  @endforeach
                  @else
                  Δεν ξέρω καμία Γλώσσα Προγραμματισμού..
      @endif
      <br><br>
```

```
      @for ($i=0; $i<10; $i++)
      Η τρέχουσα τιμή του i είναι: {{ $i }}
      <br>
      @endfor

      @while(true)
      Ατέρμον Βρόχος
      @endwhile
      </body>
</html>
```

Παραδείγματα Ενότητας 4.4.4

Αρχείο routes.php

```
Route::get('/', function () {
      return view('home');
});

Route::get('about', function () {
      return view('about');
});

Route::get('contact', function () {
      return view('contact');
});
```

Αρχείο master.blade.php

```
<nav>
      @section('menu')
      <a href="/"> Αρχική </a> |
      @show
</nav>
<br><br>
<img src="http://laravel.com/assets/img/laravel-logo.png">
<br><br>
<section>
      @yield('content')
</section>
```

Αρχείο about.blade.php

```
@extends('layouts.master')
@section('menu')
      @parent
      <a href="contact"> Επικοινωνία </a>
@endsection

@section('content')
      Καλως ήρθατε στην Σελίδα "Λίγα Λόγια.." της εφαρμογής μας.
@endsection
```

Αρχείο home.blade.php

```
@extends('layouts.master')
@section('menu')
      @parent
      <a href="about"> Λίγα λόγια.. </a> |
      <a href="contact"> Επικοινωνία </a>
@endsection

@section('content')
      Καλως ήρθατε στην Αρχική Σελίδα της εφαρμογής μας.
@endsection
```

Αρχείο contact.blade.php

```
@extends('layouts.master')
@section('menu')
      @parent
      <a href="about"> Λίγα λόγια.. </a>
@endsection

@section('content')
      Καλως ήρθατε στην Σελίδα Επικοινωνίας της εφαρμογής μας.
@endsection
```

Αρχείο welcome.blade.php

```
<html>
      <body>
      @if (count($web) === 1)
      Ξέρω μια Γλώσσα Προγραμματισμού web: {{ $web[0] }}
            @elseif (count($web) > 1)
                  @foreach ($web as $web)
                  Ξέρω την Γλώσσα Προγραμματισμού web: {{ $web }}
                  <br><br>
                  @endforeach
                  @else
                  Δεν ξέρω καμία Γλώσσα Προγραμματισμού..
      @endif
      <br><br>
      @for ($i=0; $i<10; $i++)
            Η τρέχουσα τιμή του i είναι: {{ $i }}
      <br>
      @endfor

      @while(true)
            Ατέρμον Βρόχος
      @endwhile
    </body>
</html>
```

Παραδείγματα Ενότητας 4.4.6

Αρχείο style.css

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

Αρχείο script.js

```
alert('It Works!!');
```

Αρχείο routes.php

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

Αρχείο welcome.blade.php

```
<html>
      <head>
      <!-- Klassikos tropos na eisagoume script -->
      <script type="text/javascript" scr="js/script.js">
      </script>

      <!-- Me th methodo script() -->
      {!! Html::script('js/script.js') !!}
```

```
    <!-- Klassikos tropos na eisagoume css -->
    <link rel="stylesheet" type="text/css" href="style.css">

    <!-- Me th methodo style() -->
    {!! Html::style('css/style.css') !!}
    </head>
    <body>

    <!-- Klassikos tropos na eisagoume link -->
    <a href="http://dga.gr"> Digital Academy </a>

    <!-- Tropos na eisagoume link me th methodo link() -->
    {!! Html::link('http://google.com', 'Google') !!}

    <!-- Klassikos tropos na dhmiourghsoume mia ul list -->
    <ul>
    <li> a </li> <li> b </li> <li> c </li>
    </ul>

    <!-- Dimiourgia ul list me th methodo ul() -->
    {!! Html::ul(array('a', 'b', 'c')) !!}
    </body>
</html>
```

Παράδειγμα Ενότητας 4.5.1

Αρχείο routes.php

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

Route::get('verify', function() {
      return 'Τα στοιχεία έχουν καταχωρηθεί με επιτυχία!';
});

Route::post('verify', function() {
      var_dump($_POST);
});
```

Αρχείο welcome.blade.php

```
<html>
      <body>
      <form action="verify" method="GET">
            <br><br>
             <label>Όνομα Χρήστη: </label>
                 <input type="text" name="username">
             <br><br>
             <label>Κωδικός Χρήστη: </label>
                 <input type="password" name="password">
             <br><br>
             <input type="submit" value="Αποστολή">
```

```
    </form>


        /* Φόρμα με χρήση της μεθόδου POST */
        <form action="verify" method="POST">
            <input type="hidden" name="_token" value="{{
        scrf_token() }}">
         <br><br>
         <label>Όνομα Χρήστη: </label>
            <input type="text" name="username">
         <br><br>
         <label>Κωδικός Χρήστη: </label>
            <input type="password" name="password">
         <br><br>
            <input type="submit" value="Αποστολή">
        </form>
    </body>
</html>
```

Παράδειγμα Ενότητας 4.5.2

Αρχείο routes.php

```
Route::get('/', function () {
    $ages = array('15-18', '19-25', '25-35', '35+');
    return view('welcome')->with('age', $ages);
});

Route::post('/', function () {
    return 'Η φόρμα έχει καταχωρηθεί!';
});
```

Αρχείο welcome.blade.php

```
<html>
	<body>
	{!! Form::open(array('url' => '/', 'method' => 'POST'))
	!!}
	{!! Form::label('username', 'Όνομα Χρήστη:') !!}
	{!! Form::text('username', 'Δώσε ένα username...') !!}
	<br><br>
	{!! Form::label('email', 'Email:') !!}
	{!! Form::text('email', 'example@gmail.com') !!}
	<br><br>
	{!! Form::label('age', 'Ηλικία:') !!}
	{!! Form::select('age', $ages=array('15-18', '19-25', '25-
35', '35+')) !!}
	<br><br>
	{!! Form::label('fulo', 'Φύλο:') !!}
	Άνδρας {!! Form::radio('man', 'Άνδρας') !!}
	Γυναίκα {!! Form::radio('woman', 'Γυναίκα') !!}
	<br><br>
	{!! Form::submit('Αποστολή') !!}
	{!! Form::close() !!}
	</body>
</html>
```

Παράδειγμα Ενότητας 4.6.1

Αρχείο routes.php

```
Route::get('/', function () {
	$name = DB::Connection()->getDatabaseName();
	return 'Συνδεθήκατε στη Βάση Δεδομένων'. $name;
});
```

Παραδείγματα Ενότητας 4.6.2

Αρχείο routes.php

```
Route::get('/', function () {
	/* Εμφάνιση όλων των στοιχείων των users */
	$users = DB::select("SELECT * FROM users");
	var_dump($users);

	/* Εμφάνιση όλων των στοιχείων του user με id = 1 */
	$users = DB::select('SELECT * FROM users WHERE id = 1');
	dd($user);

	/* Εμφάνιση όλων των στοιχείων ενος user χωρίς να εμφανίζεται ο πίνακας array*/
	$users = DB::selectOne('SELECT * FROM users WHERE id = 1');
	dd($user);

	/* Εισαγωγή ενος νέου user */
	DB::insert('INSERT INTO users (name, job) VALUES (?, ?)', ['Akis', 'Management']);
	return 'Η εισαγωγή του νέου user έγινε με επιτυχία!';
```

```
    /* Ενημέρωση των στοιχείων ενος υπάρχοντος user*/
    DB::update('UPDATE users SET job = ? WHERE name = ?',
['Developer' ,'Akis']);
    return 'Η ενημέρωση των στοιχείων έγινε με επιτυχία!';

    /* Διαγραφή ενος user */
    DB::delete('DELETE FROM users WHERE id = 4');
    return 'Ο χρήστης διαγράφηκε!';

    /* Γενικό Ερώτημα - Προσθήκη νέας στήλης */
    DB::statement('ALTER TABLE users ADD email VARCHAR(50)');
    return 'Το πεδίο "email" προστέθηκε!';
});
```

Παραδείγματα Ενότητας 4.6.3

Αρχείο routes.php

```
Route::get('/', function () {
    // Εμφάνιση των στοιχείων όλων των χρηστών
    $users = DB::table('users')->get();
    dd($users);

    // Εμφάνιση των στοιχείων των χρηστών με id = 2
    $users = DB::table('users')->where('id', 2)->first();
    dd($users);

    // Εμφάνιση των στοιχείων των χρηστών με id = 2, με χρήση
    Magic Method
    $users = DB::table('users')->whereId(2)->first();
    dd($users);
```

```
    // Εμφάνιση των στοιχείων των χρηστών με id > 2
    $users = DB::table('users')->where('id','>', 2)->first();
    dd($users);

    // Εισαγωγή νέου χρήστη
    DB::table('users')->insert(['name' => 'Mixalis', 'job' =>
'It']);
    return 'Η εισαγωγή του χρήστη έγινε με επιτυχία!';

    // Ενημέρωση στοιχείων χρήστη με id = 4
    DB::table('users')->where('id',   4)->update(['name'   =>
'Nikos']);
    return 'Η ενημέρωση του χρήστη έγινε με επιτυχία!';

    // Διαγραφή χρήστη
    DB::table('users')->where('id', 4)->delete();
    return 'Η διαγραφή του χρήστη έγινε με επιτυχία!';
});
```

Παραδείγματα Ενότητας 4.7

Αρχείο1 routes.php

```
Route::get('/', function () {
    $projects = DB::table('projects')->get();
    return view('welcome')->withProjects($projects);
});
```

```
Route::post('add', function () {
      $name = Input::get('name');
      if ( DB::table('projects')->whereName($name)->first() !==
null )
            return 'Το όνομα υπάρχει ήδη. Δοκιμάστε ξανά!';

      DB::table('projects')->insert(['name' => $name]);
      return Redirect::to('/');
});
```

Αρχείο1 welcome.blade.php

```
<html>
      <body>
      {!! Form::open(['url' => 'add', 'method' => 'POST']) !!}
            {!! Form::token() !!}
            {!! Form::label('name', ' Όνομα Project: ') !!}
            {!! Form::text('name') !!}
            {!! Form::submit(' Προσθήκη!! ') !!}
      {!! Form::close() !!}
      <br>
        @foreach($projects as $project)
        <div>
           {!! $project->Name !!}  ( {!! $project->Money !!} €)
        </div>
        @endforeach
      </body>
</html>
```

Αρχείο2 routes.php

```
Route::get('/', function () {
      $projects = DB::table('projects')->get();
      return view('welcome')->withProjects($projects);
});

Route::post('add', function () {
      $name = Input::get('name');
      if ( DB::table('projects')->whereName($name)->first() !==
null )
      return 'Το όνομα υπάρχει ήδη. Δοκιμάστε ξανά!';
      DB::table('projects')->insert(['name' => $name]);
      return Redirect::to('/');
});

Route::post('donate', function () {
      $donation = Input::get('donation');
      $id = Input::get('id');
      DB::table('projects')->whereId($id)->increment('money',
$donation);
      return Redirect::to('/');
});
```

Αρχείο2 welcome.blade.php

```
<html>
      <head>
      <style type="text/css">
            body {
                margin-top:50px;
                text-align:center;
                background-color: #1c39bb;
                color: white;
             }

             div {
                margin: 30px 0;
             }

             form {
                  display: inline;
             }

      </style>
      </head>
      <body>
      {!! Form::open(['url' => 'add', 'method' => 'POST']) !!}
             {!! Form::token() !!}
             {!! Form::label('name', ' Όνομα Project: ') !!}
             {!! Form::text('name') !!}
             {!! Form::submit(' Προσθήκη!! ') !!}
      {!! Form::close() !!}
      <br>
```

```
        @foreach($projects as $project)
        <div>
              {!! $project->Name !!}  ( {!! $project->Money !!} €)
               -
               {!! Form::open(['url' => 'donate']) !!}
               {!! Form::hidden('id', $project->Id) !!}
               {!! Form::selectRange('donation', 10, 100) !!}
               {!! Form::submit(' Δωρεά! ') !!}
               {!! Form::close() !!}
        </div>
        @endforeach
        </body>
</html>
```

Παραδείγματα ενότητας 4.9.1

Αρχείο routes.php

```
Route::get('/', 'HomeController@showWelcome');


Route::get('about', 'HomeController@showAbout');


Route::get('milkshakes/{flavor?}',
'MilkshakeController@index');
```

Αρχείο HomeController.php

```
<?php
namespace App\Http\Controllers;
class HomeController extends Controller {
      public function showWelcome () {
            return 'Welcome to Controllers';
      }
      public function showAbout () {
            return 'About Us';
      }
}
```

Αρχείο MilkshakeController.php

```
<?php
namespace App\Http\Controllers;
class MilkshakeController extends Controller {
      public function index ($flavor = null) {
            $availableFlavors = ['banana', 'mango', 'vanilla'];
            if ($flavor === null)
                  dd($availableFlavors);
            if (in_array($flavor, $availableFlavors))
            {
                  return ' Έχουμε τη γεύση :  ' . $flavor;
            }
            else
            {
```

```
            return ' Δεν έχουμε τη γεύση ' . $flavor . ' που
ζητήσατε.';
            }
      }
}
```

Παραδείγματα Ενότητας 4.9.2

Αρχείο routes.php

```
Route::get('/', 'HomeController@showWelcome');

Route::controller('portfolio', 'PortfolioController');
```

Αρχείο HomeController.php

```
<?php
namespace App\Http\Controllers;
class HomeController extends Controller {
      public function showWelcome() {
            return view('welcome');
      }
}
```

Αρχείο PortfolioController.php

```
<?php
namespace App\Http\Controllers;
class PortfolioController extends Controller {
      public function getIndex () {
            return 'Αυτή είναι η κύρια σελίδα του portfolio';
```

```
	}
	public function getFilms () {
		return 'Αυτή είναι η λίστα με τις ταινίες μου';
	}
	public function getMusic () {
		return 'Αυτή είναι΄η λίστα με τη μουσική μου';
	}

	public function postProcess () {
		return 'Form was POST-ed';
	}
}
```

Αρχείο welcome.blade.php

```
<html>
	<body>
	{!! Form::open(['url' => 'portfolio/process']) !!}
		{!! Form::submit() !!}
	{!! Form::close() !!}
	</body>
</html>
```

Παραδείγματα Ενότητας 4.9.3

Αρχείο routes.php

```
Route::get('/', 'HomeController@showWelcome');

Route::resource('users', 'UserController');
```

Αρχείο UserController.php

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Http\Requests;
use App\Http\Controllers\Controller;
class UserController extends Controller {
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index() {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
```

```
public function create() {
    return 'Φόρμα για τη δημιουργία νέου χρήστη';
}
/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request) {
    //
}
/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id) {
    //
}
/**
 * Show the form for editing the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function edit($id) {
    return 'Φόρμα για επεξεργασία του χρήστη: ' .$id;
}
```

```
    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id) {
        return 'Η φόρμα προστέθηκε. Ενημερώθηκαν τα στοιχεία του χρήστη ' . $id . ' στη Βάση';
    }
    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id) {
        //
    }
}
```

Αρχείο welcome.blade.php

```
<html>
     <body>
     {!! Form::open(['url' => 'users/user1', 'method' => 'PUT']) !!}
          {!! Form::submit() !!}
     {!! Form::close() !!}
     </body>
```

```
</html>
```

Παραδείγματα Ενότητας 4.10.1

Αρχείο create_books_table.php

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class CreateBooksTable extends Migration {
    public function up() {
        Schema::create('books', function (Blueprint $table) {
            $table->increments('id');
            $table->string('title');
            $table->string('author');
            $table->text('summary')->nullable();
            $table->timestamps();
        });
    }
    public function down() {
        Schema::drop('books');
    }
}
```

Αρχείο add_isbn_field.php

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class AddIsbnField extends Migration {
    public function up() {
        Schema::table('books', function ($table)
        {
            $table->integer('isbn');
            $table->unique('title');
        });
    }
    public function down() {
        Schema::table('books', function ($table)
        {
            $table->dropColumn('isbn');
            $table->dropUnique('books_title_unique');
        });
    }
}
```

Παράδειγμα Ενότητας 4.10.2

Αρχείο DatabaseSeeder.php

```
<?php
use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;
class DatabaseSeeder extends Seeder {
    public function run() {
        Model::unguard();
        $this->call(BooksTableSeeder::class);
        Model::reguard();
    }
}

class BooksTableSeeder extends Seeder {
    public function run () {
        for ($i = 0; $i < 10; $i++)
        {
         DB::table('books')->insert(['title' => 'Βιβλίο '.$i,
'author' => 'Συγγραφέας ' .$i]);
        }
    }
}
```

Παραδείγματα Ενότητας 4.10.3

Αρχείο create_articles_table.php

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class CreateArticlesTable extends Migration {
    public function up() {
     Schema::create('articles', function (Blueprint $table)
     {
          $table->increments('id');
          $table->string('title')->unique();
          $table->text('body');
        });
    }
    public function down() {
        Schema::drop('articles');
    }
}
```

Αρχείο add_timestamps.php

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class AddTimestamps extends Migration {
    public function up() {
        Schema::table('articles', function (Blueprint $table)
        {
```

```
            $table->timestamps();
        });
    }
    public function down() {
        Schema::table('articles', Function (Blueprint $table)
        {
            $table->dropColumn(['created_at', 'updated_at']);
        });
    }
}
```

Αρχείο Article.php

```
<?php
namespace App;
use App\Article;
use Illuminate\Database\Eloquent\Model;
class Article extends Model {
    protected $fillable = ['title', 'body'];
}
```

Αρχείο routes.php

```
Route::get('/', function () {
     // 1ος τρόπος προσθήκης άρθρου.
     $article1 = App\Article::create([
     'title' => 'Laravel 5.1 (1)',
     'body' => 'part 1'
     ]);
     return $article1->id;
```

```
// 2ος τρόπος προσθήκης άρθρου.
$article2 = new App\Article();
$article2->title = 'Laravel 5.1 (2)';
$article2->body = 'part 2';
$article2->save();
return 'Article saved!';

// Προσθήκη νέου άρθρου.
$article3 = App\Article::create([
'title' => 'Laravel 5.1 (3)',
'body' => 'part 3'
]);
return;

// Εμφάνιση όλων των άρθρων της Βάσης.
$articles = App\Article::all();
foreach ($articles as $art)
{
      var_dump($art->title);
}
return;

// Εμφάνιση των άρθρων με id > 1.
$articles = App\Article::where('id', '>', 1);
foreach ($articles as $art)
{
      var_dump($art->title);
}
return;
```

```
    // Εμφάνιση άρθρου με συγκεκριμένο id = 5.
    $article = App\Article::find(5);
    return $article->title;

    // Ενημέρωση στοιχείων του άρθρου με id = 5.
    $article = App\Article::find(5);
    $article->title = 'Learn PHP';
    $article->save();
    return 'Αποθηκεύτηκε! Ο νέος τίτλος είναι: '
    .$article->title;

    // Ενημέρωση στοιχείων των άρθρων με id > 5.
    $articles = App\Article::where('id', '>', 5);
    $articles->update(['body' => '...']);
    return 'Μαζική Ενημέρωση';

    // Διαγραφή άρθρου με id = 5.
    $article = App\Article::destroy(5);

    // Μαζική διαγραφή άρθρων.
    $deletearticles = App\Article::where('id', '<', 5)
    ->delete();
    return 'Μαζική διαγραφή!!';
});
```

Παραδείγματα Ενότητας 4.10.4

Αρχείο create_articles_table.php

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class CreateArticlesTable extends Migration {
    public function up() {
     Schema::create('articles', function (Blueprint $table)
     {
          $table->increments('id');
          $table->string('title')->unique();
          $table->text('body');
        });
    }
    public function down() {
        Schema::drop('articles');
    }
}
```

Αρχείο create_comments_table.php

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class CreateCommentsTable extends Migration {
    public function up() {
     Schema::create('comments', function (Blueprint $table)
     {
```

```
            $table->increments('id');
            $table->text('body');
            $table->integer('article_id')->unsigned();
            $table->timestamps();
            $table->foreign('article_id')->references('id')-
>on('articles');
        });
    }
    public function down() {
        Schema::drop('comments');
    }
}
```

Αρχείο Article.php

```
<?php
namespace App;
use App\Article;
use Illuminate\Database\Eloquent\Model;
class Article extends Model {
     protected $fillable = ['title', 'body'];

     public function comments () {
     return $this->hasMany('App\Comment');
     }
}
```

Αρχείο Comment.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Comment extends Model {
      protected $fillable = ['body', 'article_id'];

      public function article () {
      return $this->belongsTo('App\Article');
      }
}
```

Αρχείο DatabaseSeeder.php

```
<?php
use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;
class DatabaseSeeder extends Seeder {
    public function run() {
        Model::unguard();
        $this->call(CommentsTableSeeder::class);
        Model::reguard();
    }
}
class ArticlesTableSeeder extends Seeder {
    public function run () {
        App\Article::create(['title' => 'Article 1', 'body' =>
'Body 1']);
        App\Article::create(['title' => 'Article 2', 'body' =>
'Body 2']);
```

```
    }
}
class CommentsTableSeeder extends Seeder {
    public function run () {
        for ($i = 0; $i < 4; $i++)
        {
            App\Comment::create(['body'   =>   'Comment'   .$i,
'article_id' => '7']);
        }
        for ($i = 4; $i < 6; $i++)
        {
            App\Comment::create(['body'   =>   'Comment'   .$i,
'article_id' => '8']);
        }
    }
}
```

Αρχείο routes.php

```
Route::get('/', function () {
      $article = App\Comment::find(6)->article;
      var_dump($article->title);

      $comments = App\Article::find(8)->comments;
      foreach($comments as $comment)
      {
            var_dump($comment->body);
      }
});
```

Παραδείγματα Ενότητας 4.11.1

Αρχείο routes.php

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

Route::post('/', function () {
      $rules = [
            'name' => 'required',
            'link' => 'required|url',
            'password' => 'required|min:8',
            'password-repeat' => 'required|same:password'
      ];
      $validator = Validator::make(Input::all(), $rules);
      if ($validator->fails())
      {
      return Redirect::to('/') ->
      withInput()->withErrors($validator->messages());
      }
      return 'Η υποβολή της φόρμας έγινε με επιτυχία!!';
});
```

Αρχείο welcome.blade.php

```
<html>
      <body>
      @foreach($errors->all('<p> :message </p>') as $error)
      {!! $error !!}
      @endforeach
      {!! Form::open(['url' => '/']) !!}
      <p>
            {!! Form::label('name') !!}
            {!! Form::text('name') !!}
      </p>
      <p>
            {!! Form::label('link') !!}
            {!! Form::text('link') !!}
      </p>
      <p>
            {!! Form::label('password') !!}
            {!! Form::password('password') !!}
      </p>
      <p>
            {!! Form::label('password-repeat') !!}
            {!! Form::password('password-repeat') !!}
      </p>
      <p>
            {!! Form::submit() !!}
      </p>
      {!! Form::close() !!}
      </body>
</html>
```

Παραδείγματα Ενότητας 4.11.2

Αρχείο 404.blade.php

```
<html>
      <head>
            <title>404..</title>
      <link
      href="https://fonts.googleapis.com/css?family=Lato:100"
      rel="stylesheet" type="text/css">


      <style>
            html, body {
                  height: 100%;
            }
            body {
                  margin: 0;
                  padding: 0;
                  width: 100%;
                  color: #B0BEC5;
                  display: table;
                  font-weight: 100;
                  font-family: 'Lato';
             }
             .container {
                  text-align: center;
                  display: table-cell;
                  vertical-align: middle;
             }
```

```
            .content {
                text-align: center;
                display: inline-block;
            }
            .title {
                font-size: 72px;
                margin-bottom: 40px;
            }
      </style>
      </head>
      <body>
            <div class="container">
            <div class="content">
                  <div class="title"> 404 error :( </div>
            </div>
            </div>
      </body>
</html>
```

Αρχείο 503.blade.php

```
<html>
      <head>
      <title>Maintenance mode.</title>
      <link
      href="https://fonts.googleapis.com/css?family=Lato:100"
      rel="stylesheet" type="text/css">
      <style>
            html, body {
                  height: 100%;
             }
             body {
                 margin: 0;
                 padding: 0;
                 width: 100%;
                 color: #B0BEC5;
                 display: table;
                 font-weight: 100;
                 font-family: 'Lato';
            }
            .container {
                 text-align: center;
                 display: table-cell;
                 vertical-align: middle;
            }
            .content {
                 text-align: center;
                 display: inline-block;
            }
```

```
            .title {
                font-size: 72px;
                margin-bottom: 40px;
            }
        </style>
        </head>
        <body>
            <div class="container">
            <div class="content">
                <div class="title"> Maintenance Mode.. </div>
            </div>
            </div>
        </body>
</html>
```

Παραδείγματα Ενότητας 4.12

Αρχείο HomeController.php

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Auth;
use App\Http\Requests;
use App\Http\Controllers\Controller;
use App\Http\Controllers\UserController;
use Validator, Input, Redirect, Hash;
class HomeController extends Controller {
	public function getIndex () {
		return Redirect::to('users');
	}
	public function getLogin () {
		return view('login');
	}
	public function postLogin () {
		$creds = [
		'username' => Input::get('username'),
		'password' => Input::get('password')
		];

		if (Auth::attempt($creds))
		{
			return Redirect::to('users');
		}
```

```
            else
            {
                  return Redirect::to('login')->withInput();
            }
      }
      public function getLogout () {
            Auth::logout();
            return Redirect::to('users');
      }
      public function getMembers () {
            return '<h2> Members Area.. </h2>';
      }
}
```

Αρχείο UsersController.php

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Auth;
use App\Http\Requests;
use App\Http\Controllers\Controller;
use Validator, Input, Redirect, Hash;
class UsersController extends Controller {
    public function index() {
        return view('users')->withUsers(\App\User::all());
    }
    public function create() {
        return view('create');
    }
```

```
public function store(Request $request) {
    $rules = [
        'username' => 'required|unique:users',
        'password' => 'required|min:6',
        'password-repeat' => 'required|same:password'
    ];
    $validator = Validator::make(Input::all(), $rules);
    if ($validator->fails())
    {
       return Redirect::to('users/create')->
       withInput()->withErrors($validator->messages());
    }
    \App\User::create([
        'username' => Input::get('username'),
        'password' => Hash::make(Input::get('password')),
        'bio' => Input::get('bio')
        ]);
    return Redirect::to('users');
}
public function show($id) {
    $user = \App\User::find($id);
    if ($user === null)
    {
        return Redirect::to('users');
    }
    return view('profile')->withUser($user);
}
public function edit($id) {
    $user = \App\User::find($id);
    if ($user === null)
```

```
    {
        return Redirect::to('users');
    }
    return view('edit')->withId($id);
}
public function update(Request $request, $id) {
    $rules = [
    'username' => 'unique:users',
    'password' => 'min:6'
    ];
    $validator = Validator::make(Input::all(), $rules);
    if($validator->fails())
    {
       return Redirect::to('users/' .$id. '/edit')
       ->withInput()->withErrors($validator->messages());
    }
    $user = \App\User::find($id);
    if (Input::has('username'))
    {
        $user->username = Input::get('username');
    }
    if (Input::has('bio'))
    {
        $user->bio = Input::get('bio');
    }
    if (Input::has('password'))
    {
     $user->password = Hash::make(Input::get('password'));
    }
    $user->save();
```

```
        return Redirect::to('users/' .$id);
    }
    public function destroy($id) {
        $fallenOne = \App\User::find($id);
        $fallenOne->delete();
    }
}
```

Αρχείο create_users_table.php

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class CreateUsersTable extends Migration {
    public function up() {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('username')->unique();
            $table->string('password');
            $table->text('bio')->nullable();
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::drop('users');
    }
}
```

Αρχείο add_remember_token_to_users.php

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class AddRememberTokenToUsers extends Migration {
    public function up() {
        Schema::table('users', function (Blueprint $table)
        {
            $table->rememberToken();
        });
    }
    public function down() {
        Schema::table('users', function (Blueprint $table)
        {
            $table->dropColumn('remember_token');
        });
    }
}
```

Αρχείο master.blade.php

```
<title>
     @yield('title')
</title>
<body>
     @yield('content')
</body>
```

Αρχείο create.blade.php

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

@section('title')
      Join Us
@endsection

@section('content')
      @foreach($errors->all() as $error)
            <p>{!! $error !!}</p>
      @endforeach
      {!! Form::open(['url' => 'users']) !!}
      <p>
            {!! Form::label('username', 'Username*') !!}
            {!! Form::text('username') !!}
      </p>
      <p>
            {!! Form::label('bio', 'Bio') !!}
            {!! Form::textarea('bio') !!}
      </p>
      <p>
            {!! Form::label('password', 'Password*') !!}
            {!! Form::password('password') !!}
      </p>
      <p>
            {!!    Form::label('password-repeat',    'Password-
            repeat*') !!}
            {!! Form::password('password-repeat') !!}
      </p>
```

```
      <p>
            {!! Form::submit() !!}
      </p>
      {!! Form::close() !!}
@endsection
```

Αρχείο edit.blade.php

```
@extends('layouts.master')
@section('title')
      Edit My Profile
@endsection
@section('content')
      {!! Form::open(['url' => 'users/' .$id, 'method' => 'PUT'])
      !!}
      <p>
            {!! Form::label('username', 'New Username') !!}
            {!! Form::text('username') !!}
      </p>
      <p>
            {!! Form::label('bio', 'New Bio') !!}
            {!! Form::textarea('bio') !!}
      </p>
      <p>
            {!! Form::label('password', 'New Password') !!}
            {!! Form::password('password') !!}
      </p>
      <p>
            {!! Form::submit('Update') !!}
      </p>
```

```
      {!! Form::close() !!}
      <hr>
      {!! Form::open(['url' => 'users/' .$id, 'method' => 'DELETE']) !!}
            {!! Form::submit('Delete') !!}
      {!! Form::close() !!}
@endsection
```

Αρχείο login.blade.php

```
@extends('layouts.master')
@section('title')
      Login
@endsection
@section('content')
      {!! Form::open(['url' => 'login']) !!}
            <p>
                  {!! Form::label('username') !!}
                  {!! Form::text('username') !!}
            </p>
            <p>
                  {!! Form::label('password') !!}
                  {!! Form::password('password') !!}
            </p>
            <p>
                  {!! Form::submit('Login') !!}
            </p>
      {!! Form::close() !!}
@endsection
```

Αρχείο profile.blade.php

```
@extends('layouts.master')
@section('title')
      {!! $user->username !!}
@endsection
@section('content')
      <p>
            <h3> Username: {!! $user->username !!} </h3>
      </p>
      <p>
            {!! $user->bio !!}
      </p>
      {!!  Html::link('users/'  .$user->id.  '/edit',  'Edit  My
Profile') !!}
@endsection
```

Αρχείο users.blade.php

```
@extends('layouts.master')
@section('title')
      Users
@endsection
@section('content')
      @if(Auth::check())
            Current User: {!! Auth::user()->username !!} .({!!
            Html::link('logout', 'Logout') !!})
      @endif
      <h1>Λίστα με όλους τους χρήστες:</h1>
      @foreach($users as $user)
```

```
    <p> {!! $user->username !!} ({!! Html::link('users/'
.$user->id, 'Profile') !!}) </p>
    @endforeach
@endsection
```

Αρχείο welcome.blade.php

```
<html>
    <head>
    <title>Laravel</title>
    <link
    href="https://fonts.googleapis.com/css?family=Lato:100"
    rel="stylesheet" type="text/css">
    <style>
        html, body {
            height: 100%;
        }
        body {
            margin: 0;
            padding: 0;
            width: 100%;
            display: table;
            font-weight: 100;
            font-family: 'Lato';
        }
        .container {
            text-align: center;
            display: table-cell;
            vertical-align: middle;
        }
```

```
            .content {
                text-align: center;
                display: inline-block;
            }
            .title {
                font-size: 96px;
            }
      </style>
      </head>
      <body>
            <div class="container">
            <div class="content">
                  <div class="title">Laravel 5</div>
            </div>
            </div>
      </body>
</html>
```

Αρχείο routes.php

```
Route::resource('users', 'UsersController');

Route::controller('/', 'HomeController');
```

Αρχείο User.php

```
<?php

namespace App;

use Illuminate\Auth\Authenticatable;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Auth\Passwords\CanResetPassword;

use Illuminate\Foundation\Auth\Access\Authorizable;

use Illuminate\Contracts\Auth\Authenticatable as
AuthenticatableContract;

use Illuminate\Contracts\Auth\Access\Authorizable as
AuthorizableContract;

use Illuminate\Contracts\Auth\CanResetPassword as
CanResetPasswordContract;

class User extends Model implements AuthenticatableContract,
                                    AuthorizableContract,
                                    CanResetPasswordContract {

    use Authenticatable, Authorizable, CanResetPassword;

    protected $table = 'users';

    protected $guarded = [];

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

}
```

Βιβλιογραφία

[1] Moodle Documentation – https://docs.moodle.org/30/en/Main_page

[2] PHP - http://www.php.net

[3] Moodle LMS – https://el.wikipedia.org/wiki/Moodle

[4] Σύγχρονη/Ασύγχρονη τηλεκπαίδευση – http://sugxroni-asugxroni-e.pblogs.gr

[5] PHP Frameworks - http://mashable.com/2014/04/04/php-frameworks-build-applications/#xwGCP2M9LGqW

[6] Πλεονεκτήματα του Laravel Framework - http://www.amarinfotech.com/why-laravel-is-best-php-framework-in-2016.html

[7] Χρήση PHP Frameworks - https://webandmobiletech.wordpress.com/2016/01/01/top-5-best-php-frameworks-for-2016-to-become-a-master-developer/

[8] Learning Management Systems - https://en.wikipedia.org/wiki/Learning_management_system

[9] Open eClass - http://www.openeclass.org/

[10] Blackboard – http://uki.blackboard.com

[11] MVC - https://el.wikipedia.org/wiki/Model-view-controller

[12] Tutorials about Laravel Framework - http://www.tutorialspoint.com/laravel/index.htm

[13] Laravel Framework Documentation - https://laravel.com/docs/5.1

[14] Heintzelman, C. (2016). Laravel 5.1 Beauty

[15] Laravel History - https://en.wikipedia.org/wiki/Laravel

[16] Δημοφιλής PHP Framework - http://www.sitepoint.com

[17] Κατσούκας Η., A Very Complete Introduction To Laravel, Αθήνα: DGA, 2016