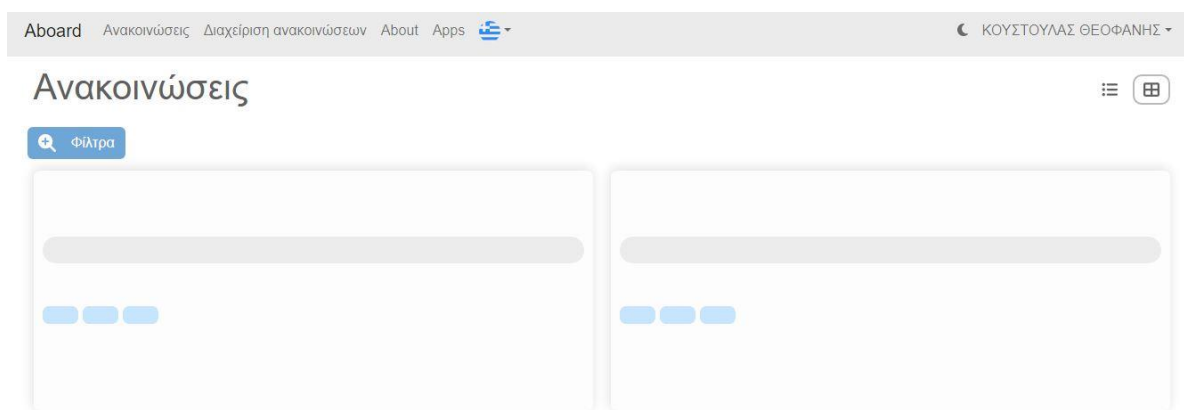


ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επανασχεδιασμός και ανακατασκευή της πλατφόρμας
ανακοινώσεων aboard του τμήματος Μηχανικών
Πληροφορικής και Ηλεκτρονικών Συστημάτων



Του φοιτητή
Θεοφάνη Κουστούλα
Αρ. Μητρώου: 134062

Επιβλέπων
Αντώνης Σιδηρόπουλος

Μάιος 2023

Τίτλος Π.Ε. Επανασχεδιασμός και ανακατασκευή της πλατφόρμας ανακοινώσεων aboard του
τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

Κωδικός Π.Ε. 22299

Θεοφάνης Κουστούλας

Αντώνης Σιδηρόπουλος

Ημερομηνία ανάληψης Π.Ε. 26-10-22

Ημερομηνία περάτωσης Π.Ε. 22-05-23

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Θεοφάνη Κουστούλα που την εκπόνησε/αν. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραίτητως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Στην Χαμαϊδή – Άννα Μπουρονίκου

Πρόλογος

Η παρούσα πτυχιακή εργασία αποτελεί ένα εργαλείο που θα χρησιμοποιηθεί μελλοντικά από το τμήμα και είναι και ένας από τους λόγους που την επέλεξα σκεπτόμενος ότι με αυτόν τον τρόπο θα μπορέσω να παρέχω κάτι στην σχολή που μου παρείχε τόσες γνώσεις για όλα τα χρόνια των σπουδών μου. Δουλεύοντας πάνω στην εργασία άντλησα πληθώρα πληροφοριών σχετικά με την υλοποίηση και συντήρηση διαδικτυακών εφαρμογών οι οποίες θα χρησιμεύσουν στην υπόλοιπη καριέρα μου. Συγκεκριμένα, επανασχεδιάζοντας και υλοποιώντας κομμάτια της εφαρμογής εκβάθυνα στους τρόπους με τους οποίους επιτυγχάνεται η επεκτασιμότητα, κάτι το οποίο είναι από τα σημαντικότερα ζητήματα που προκύπτουν σε τέτοιου είδους εφαρμογές.

Περίληψη

Στη σύγχρονη εποχή της ψηφιακής επικοινωνίας, οι πλατφόρμες ανακοινώσεων αποτελούν ένα απαραίτητο εργαλείο για την επικοινωνία και τη διαχείριση των ανακοινώσεων σε εκπαιδευτικά ιδρύματα.

Η πλατφόρμα aboard του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων είναι ένα τέτοιο εργαλείο και χρησιμοποιείται ευρέως για την ενημέρωση των φοιτητών σχετικά με επιστημονικές εκδηλώσεις, ανακοινώσεις εργασιών και διαγωνισμών καθώς και για την παρουσίαση διαφόρων δραστηριοτήτων που αφορούν το τμήμα.

Στόχος της παρούσας πτυχιακής εργασίας είναι να αναπτυχθεί μια νέα έκδοση της πλατφόρμας aboard η οποία θα περιλαμβάνει βελτιωμένη λειτουργικότητα, χρηστικότητα και επεκτασιμότητα.

Αρχικά, γίνεται ανάλυση των απαιτήσεων του έργου και στην συνέχεια αναλύονται οι τεχνολογίες που χρησιμοποιούνται για την υλοποίηση της πλατφόρμας με μία ιστορική αναδρομή για την κάθε μία όπου δίνεται έμφαση στα προτερήματα και τα μειονεκτήματα που υπάρχουν σε σχέση με αντίστοιχες τεχνολογίες. Στην συνέχεια τονίζονται οι διαδικασίες που υλοποιήθηκαν ώστε να γίνει η μετάβαση στην νέα έκδοση διατηρώντας την λειτουργικότητα της προηγούμενης η οποία είναι συνδεδεμένη με τρίτες εφαρμογές. Έπειτα, γίνεται μία πλήρης παρουσίαση της εφαρμογής και τέλος αποτυπώνεται η σύνοψη της εφαρμογής, προτάσεις βελτίωσης και τα συμπεράσματα που εξάχθηκαν με την ολοκλήρωση της υλοποίησης.

Redesign and reconstruction of the Aboard announcements platform of the Department of Computer Engineering and Electronic Systems.

Theofanis Koustoulas

Abstract

In the modern era of digital communication, announcement platforms are an essential factor for communication and management of announcements in educational institutions. The aboard platform of the Department of Computer Science and Electronic Systems Engineering is widely used for announcing scientific events, work announcements, competitions, and for promoting various activities related to the department.

The goal of this thesis is to develop a new version of the aboard platform that will include improved functionality, usability, and scalability. Starting with an analysis of the technologies used to implement the platform with a historical overview of each one, emphasizing the advantages and disadvantages compared to other technologies. Next, the processes implemented to transition to the new version while retaining the functionality of the previous version, as it is used by third-party applications, are highlighted. Then, the differences between the two versions are clarified, analyzing the reasons why they were implemented in this way. Finally, the individual functions that comprise the system are analyzed.

Ευχαριστίες

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου προς την οικογένεια μου που με στήριξε στον μέγιστο βαθμό για να μπορέσω να φτάσω ως εδώ και να επιτύχω τους στόχους τους οποίους είχα θέσει στην αφετηρία αυτού του δρόμου. Επίσης, θα ήθελα να ευχαριστήσω τον καθηγητή, κ. Αντώνη Σιδηρόπουλο που μέσω της έμπειρης καθοδήγησης του ολοκλήρωσα τον παρόν έργο.

Περιεχόμενα

Πρόλογος.....	1
Περίληψη.....	2
Abstract	3
Ευχαριστίες	4
Περιεχόμενα	5
Κατάλογος Σχημάτων	9
Κατάλογος Πινάκων.....	9
Κατάλογος Εικόνων	9
Κατάλογος Κώδικα	10
Συντομογραφίες.....	12
Κεφάλαιο 1ο: Ανάλυση απαιτήσεων εφαρμογής	1
1.1 Εισαγωγή.....	1
1.2 Κατηγορίες απαιτήσεων	1
1.3 Ανάλυση του τρέχοντος συστήματος	1
1.4 Λειτουργικές απαιτήσεις εφαρμογής.....	2
1.5 Μη λειτουργικές απαιτήσεις εφαρμογής.....	3
1.6 Επίλογος.....	4
Κεφάλαιο 2ο: Εισαγωγή στις τεχνολογίες του έργου.....	5
2.1 Εισαγωγή.....	5
2.2 PHP.....	5
2.2.1 Ιστορική αναδρομή της PHP	5
2.2.2 Οι δυνατότητες της PHP.....	5
2.3 Laravel.....	5
2.3.1 Ιστορική αναδρομή της Laravel	6
2.3.2 Η αρχιτεκτονική MVC	6
2.3.3 Laravel lifecycle	7
2.3.4 Γιατί Laravel.....	8
2.4 Composer	8
2.4.1 Χρήση του composer.....	9
2.4.2 Γιατί composer	9
2.5 NPM	10
2.6 HTML/CSS/JavaScript.....	11

2.6.1	HTML.....	11
2.6.2	CSS.....	11
2.6.3	JavaScript	12
2.7	React.js	12
2.7.1	Components.....	13
2.7.2	State	13
2.7.3	Hooks.....	13
2.7.4	Γιατί ReactJS	14
2.8	Draft.js.....	14
2.8.1	Γιατί Draft.js.....	14
2.9	MySQL.....	14
2.9.1	Πίνακες.....	14
2.9.2	Πρωτεύον κλειδί.....	15
2.9.3	Ξένο κλειδί	15
2.9.4	Περιορισμοί (constraints)	15
2.9.5	Τύποι Συσχετίσεων.....	15
2.9.6	Γιατί MySQL.....	17
2.10	Επίλογος.....	17
Κεφάλαιο 3ο:	Επανασχεδιασμός της εφαρμογής	19
3.1	Εισαγωγή.....	19
3.2	Βάση δεδομένων	19
3.2.1	Τροποποιήσεις.....	19
3.3	Επανασχεδιασμός του back-end.....	20
3.3.1	Διαχωρισμός εκδόσεων	20
3.3.2	Δημιουργία νέας σουίτας routes.....	20
3.3.3	Ανάπτυξη νέου συστήματος αυθεντικοποίησης.....	21
3.3.4	Παρουσίαση νέου API.....	22
3.3.5	Απάντηση της μορφής icalendar	30
3.3.6	Βελτιστοποίηση στα SQL Queries	31
3.4	Επανασχεδιασμός του front-end.....	32
3.4.1	Αρχικό setup εφαρμογής	33
3.4.2	Routing	33
3.4.3	Pages.....	33
3.4.4	Components.....	34
3.4.5	Helpers.....	35

3.4.6	SASS	39
3.4.7	Themes	40
3.4.8	Localization	40
3.5	Επίλογος	41
Κεφάλαιο 4ο:	Παρουσίαση της εφαρμογής	42
4.1	Εισαγωγή	42
4.2	Προβολή ανακοινώσεων	42
4.2.1	Σκελετός ανακοινώσεων	42
4.2.2	Φιλτράρισμα ανακοινώσεων	43
4.2.3	Εναλλαγή τρόπου εμφάνισης λίστας	44
4.3	Προβολή ανακοίνωσης	44
4.4	Προβολή About	45
4.5	Απλός συνδεδεμένος χρήστης	45
4.5.1	Σύνδεση	46
4.5.2	Προβολή Λογαριασμός	46
4.6	Author	47
4.6.1	Actions ανακοινώσεων	48
4.6.2	Διαγραφή ανακοίνωσης	49
4.6.3	Εισαγωγή ανακοίνωσης	50
4.6.4	Επεξεργασία ανακοίνωσης	51
4.7	Admin	53
4.8	Πολυθεματικότητα και πολυγλωσσικότητα	54
4.9	Responsive	55
4.9.1	Μενού πλοήγησης	55
4.9.2	Ανακοινώσεις	56
4.9.3	Φόρμα ανακοίνωσης	56
4.10	Επίλογος	57
Κεφάλαιο 5ο:	Συμπεράσματα και προτάσεις βελτίωσης	58
5.1	Εισαγωγή	58
5.2	Σύνοψη	58
5.3	Προτάσεις βελτίωσης	58
5.3.1	Αναβάθμιση έκδοσης Laravel	58
5.3.2	Υλοποίηση Unit και Integration tests	58
5.3.3	React tests	59
5.3.4	Redux	59

5.3.5	Επιβολή ενός μόνο προτύπου συγγραφής κώδικα.....	60
5.4	Συμπεράσματα.....	60
5.5	Επίλογος.....	61
ΒΙΒΛΙΟΓΡΑΦΙΑ.....		62

Κατάλογος Σχημάτων

Σχήμα 1.1 Η αρχιτεκτονική MVC.....	6
Σχήμα 1.2 Laravel lifecycle.....	7
Σχήμα 1.3 Συσχέτιση ένα προς ένα.....	15
Σχήμα 1.4 Παράδειγμα σχέσης ένα προς ένα.....	16
Σχήμα 1.5 Σχέση ένα προς πολλά.....	16
Σχήμα 1.6 Παράδειγμα σχέσης ένα προς πολλά.....	16
Σχήμα 1.7 Σχέση πολλά προς πολλά.....	17
Σχήμα 1.8 Παράδειγμα σχέσης πολλά προς πολλά.....	17

Κατάλογος Πινάκων

Πίνακας 2.1 Εντολές composer.....	10
Πίνακας 2.2 Εντολές npm.....	11
Πίνακας 3.1 API end-points.....	23
Πίνακας 3.2 React Routes.....	33
Πίνακας 3.3 Λίστα pages.....	34
Πίνακας 3.4 Κλάση request.....	36
Πίνακας 3.5 Κλάση User.....	37
Πίνακας 3.6 Κλάση UriHelper.....	39
Πίνακας 3.7 Υπόλοιποι helpers.....	39
Πίνακας 3.8 Κλάση I18n.....	40
Πίνακας 4.1 Σκελετός ανακοινώσεων.....	43

Κατάλογος Εικόνων

Εικόνα 3.1 Παράδειγμα σχήματος της βάσης δεδομένων.....	19
Εικόνα 3.2 Διάγραμμα ακολουθίας διαδικασίας σύνδεσης χρήστη.....	21
Εικόνα 3.3 Παράδειγμα χρήσης component.....	34
Εικόνα 3.4 CustomEditor.....	35
Εικόνα 3.5 Themes εφαρμογής.....	40
Εικόνα 4.1 Προβολή ανακοινώσεων.....	42
Εικόνα 4.2 Φίλτρα αναζήτησης.....	44
Εικόνα 4.3 Προβολή ανακοινώσεων με μορφή λίστας.....	44
Εικόνα 4.4 Πλήρης προβολή ανακοίνωσης.....	45
Εικόνα 4.5 Προβολή about.....	45
Εικόνα 4.6 Login του συστήματος login.tee.ihu.gr.....	46
Εικόνα 4.7 Προβολή λογαριασμός - Ακολουθήστε ετικέτες.....	46
Εικόνα 4.8 Ομαδοποίηση επιλεγμένων στοιχείων.....	47
Εικόνα 4.9 Προβολή αναφοράς προβλήματος.....	47
Εικόνα 4.10 Προβολή Οι ανακοινώσεις μου.....	48
Εικόνα 4.11 Actions ανακοινώσεων.....	48

Εικόνα 4.12 Action button στην προβολή μίας ανακοίνωσης.....	49
Εικόνα 4.13 Popur επιβεβαίωσης διαγραφής	50
Εικόνα 4.14 Ολοκλήρωση διαγραφής	50
Εικόνα 4.15 Προβολή εισαγωγής κειμένων ανακοίνωσης.....	51
Εικόνα 4.16 Προβολή επιπρόσθετων στοιχείων ανακοίνωσης	51
Εικόνα 4.17 Προβολή επεξεργασίας ανακοίνωσης 1.....	52
Εικόνα 4.18 Προβολή επεξεργασίας ανακοίνωσης 2.....	53
Εικόνα 4.19 Προβολή δικαιωμάτων διαχειριστή σε ανακοινώσεις τρίτων.....	53
Εικόνα 4.20 Προβολή πεδίου μεταφοράς ανακοίνωσης σε άλλο author	54
Εικόνα 4.21 Προβολή με σκούρο θέμα και αγγλική γλώσσα	55
Εικόνα 4.22 Προβολή φίλτρων σε σκούρο θέμα.....	55
Εικόνα 4.23 Responsive header	56
Εικόνα 4.24 Προβολή ανακοινώσεων σε smartphone	56
Εικόνα 4.25 Φόρμα ανακοίνωσης responsive	57
Εικόνα 4.26 Φόρμα ανακοίνωσης λοιπά πεδία responsive	57
Εικόνα 5.1 Σύγκριση διαχείρισης πληροφορίας χωρίς redux και με redux.....	60

Κατάλογος Κώδικα

Κώδικας 2.1 Παράδειγμα αρχείου composer.json	9
Κώδικας 2.2 Παράδειγμα αρχείου package.json.....	10
Κώδικας 2.3 Παράδειγμα κώδικα σε react.....	13
Κώδικας 3.1 View tags_leafs	20
Κώδικας 3.2 API Routes	21
Κώδικας 3.3 Παράδειγμα response του GET /api/v2/announcements.....	25
Κώδικας 3.4 Παράδειγμα response του GET /api/v2/announcements/{id}	25
Κώδικας 3.5 Παράδειγμα response του GET /api/v2/tags	26
Κώδικας 3.6 Παράδειγμα response του GET /api/v2/filtertags	27
Κώδικας 3.7 Παράδειγμα response του GET /api/v2/most_used_tags	27
Κώδικας 3.8 Παράδειγμα response του GET /api/v2/authors.....	28
Κώδικας 3.9 Παράδειγμα response του GET /api/v2/auth/whoami.....	28
Κώδικας 3.10 Παράδειγμα response του GET /api/v2/auth/subscriptions.....	28
Κώδικας 3.11 Παράδειγμα request προς το POST /api/v2/auth/subscriptions.....	29
Κώδικας 3.12 Παράδειγμα request προς το POST /api/v2/auth/subscribe	29
Κώδικας 3.13 Παράδειγμα response από το POST /api/v2/auth/subscribe.....	30
Κώδικας 3.14 Παράδειγμα response της μορφής icalendar	30
Κώδικας 3.15 Μέθοδος μετατροπής collection σε ical string	31
Κώδικας 3.16 Μέθοδος οριοθέτησης string σε γραμμές των 75 χαρακτήρων.....	31
Κώδικας 3.17 Παράδειγμα χρήσης query builder	31
Κώδικας 3.18 Παραγόμενο query του query builder	32
Κώδικας 3.19 Χρήση join στο query ανακοινώσεων.....	32
Κώδικας 3.20 Παραγόμενο query από το διορθωμένο query builder	32
Κώδικας 3.21 CheckTree component.....	35
Κώδικας 3.22 Το αντικείμενο storage.....	37

Κώδικας 3.23 Μέθοδος transformTags	38
Κώδικας 3.24 Δεδομένα τύπου tag.....	38
Κώδικας 3.25 Η μέθοδος findOption	38
Κώδικας 3.26 Παράδειγμα χρήσης της μεθόδου μετάφρασης t.....	40

Συντομογραφίες

ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
MVC	Model View Controller
MPA	Multi Page Application
SPA	Single Page Application
DOM	Document Object Model
RDBMS	Relational Database Management Systems

Κεφάλαιο 1ο: Ανάλυση απαιτήσεων εφαρμογής

1.1 Εισαγωγή

Οι απαιτήσεις ενός έργου είναι ένα από τα σημαντικότερα βήματα για την υλοποίηση της εφαρμογής. Ως απαιτήσεις ορίζονται οι δηλώσεις που περιγράφουν είτε κάποιες πτυχές της λειτουργικότητας ενός συστήματος είτε περιορισμούς που θα πρέπει να εμπεριέχονται σε ένα σύστημα[1]. Συγκεκριμένα, οι απαιτήσεις περιγράφονται προτασιακά με τρόπο που να βοηθούν τόσο στην εμπέδωση των διαφόρων λειτουργιών από όλες τις πλευρές που συμμετέχουν στη διεκπεραίωση ενός έργου, όσο και στην υπόλοιπη διαδικασία υλοποίησης του έργου καθώς αποτελούν την βάση για όλα τα επόμενα βήματα.

1.2 Κατηγορίες απαιτήσεων

Οι απαιτήσεις μίας εφαρμογής χωρίζονται σε δύο κατηγορίες. Τις λειτουργικές και τις μη λειτουργικές απαιτήσεις.

Οι λειτουργικές απαιτήσεις περιγράφουν συγκεκριμένες λειτουργίες που θα πρέπει να έχει το σύστημα όπως για παράδειγμα η διαδικασία εγγραφής και αυθεντικοποίησης ενός χρήστη.

Οι μη λειτουργικές απαιτήσεις δεν περιγράφουν συγκεκριμένες λειτουργίες του συστήματος αλλά γενικότερες έννοιες που θα πρέπει να ικανοποιηθούν ώστε να εξασφαλιστεί η εύρυθμη λειτουργία του συστήματος. Για παράδειγμα μία μη λειτουργική απαίτηση θα μπορούσε να είναι η ασφάλεια του συστήματος, κάτι το οποίο δεν δύναται να αναλυθεί σε μία συγκεκριμένη λειτουργία παρά μόνο σαν γενικότερη έννοια καθώς ο όρος ασφάλεια εμπίπτει σχεδόν σε όλους τους τομείς ενός συστήματος.

1.3 Ανάλυση του τρέχοντος συστήματος

Στόχος της παρούσας εργασίας είναι ο επανασχεδιασμός της εφαρμογής aboard οπότε ένα από τα πρώτα βήματα για την κατανόηση του έργου είναι να γίνει μία ανάλυση της τωρινής εφαρμογής ώστε να περάσουμε στις αλλαγές που πρέπει να πραγματοποιηθούν ώστε να υπάρξει μία επιτυχημένη μετάβαση. Η τωρινή εφαρμογή υλοποιεί επιτυχώς ένα πλήρες σύστημα ανακοινώσεων το οποίο περιέχει διαδικασία αυθεντικοποίησης χρηστών με το πρωτόκολλο OAuth2.0 για σύνδεση μέσω του συστήματος login.iee.ihu.gr που παρέχεται από το τμήμα. Ο κάθε χρήστης έχει διαφορετικά δικαιώματα και υπάρχουν τέσσερις διαφορετικές κατηγορίες χρηστών. Οι μη συνδεδεμένοι χρήστες οι οποίοι έχουν πρόσβαση μόνο στις δημόσιες ανακοινώσεις (με εξαίρεση την χρήση της εφαρμογής μέσα από τις εγκαταστάσεις του τμήματος, όπου αποκτούν πρόσβαση και σε μη δημόσιες ανακοινώσεις). Οι απλοί συνδεδεμένοι χρήστες (φοιτητές) οι οποίοι έχουν πρόσβαση σε όλες τις ανακοινώσεις και στις προτιμήσεις λογαριασμού. Οι συγγραφείς οι οποίοι μπορούν να δημιουργήσουν ανακοινώσεις και ο admin ο οποίος λαμβάνει όλα τα δικαιώματα στην εφαρμογή. Η τωρινή υλοποίηση παρέχει επιπρόσθετα το καρφίτσωμα σημαντικών ανακοινώσεων στην αρχή για συγκεκριμένο χρονικό διάστημα. Έπειτα από την ανάλυση του παρόντος συστήματος διαπιστώνεται πως είναι ένα πλήρες σύστημα χωρίς ελλείψεις το οποίο παρέχει όλες τις σημαντικές λειτουργίες που θα πρέπει να βρίσκονται σε τέτοιου είδους εφαρμογές. Ο λόγος του επανασχεδιασμού του παρόντος έργου δεν είναι για λειτουργικά ελαττώματα ή ελλείψεις, αλλά περισσότερο για την υποστήριξη της επεκτασιμότητας καθώς και του εκσυγχρονισμού του έργου.

1.4 Λειτουργικές απαιτήσεις εφαρμογής

Η νέα εφαρμογή aboard υλοποιείται με έναν συνδυασμό των τεχνολογιών PHP (Laravel), JavaScript (ReactJS) και MySQL τις οποίες θα εξετάσουμε αναλυτικά στο επόμενο κεφάλαιο. Σκοπός του παρόντος έργου είναι ο εκσυγχρονισμός της εφαρμογής οπότε δεν θα αναλυθούν όλες οι απαιτήσεις από τον αρχικό σχεδιασμό αλλά μόνο αυτές που αφορούν το τωρινό έργο. Σε αυτό το κεφάλαιο θα εξετάσουμε αυτές τις απαιτήσεις ξεχωρίζοντας το back-end από το front-end για λόγους συνέπειας.

Back-end

- **Μετατροπή όλου του συστήματος από υβριδικό σε REST API**
Στην υπάρχουσα υλοποίηση κάποια end-points επιστρέφουν html μορφή και κάποια άλλα μόνο την χρήσιμη πληροφορία σε μορφή JSON. Η παραπάνω λύση είναι επαρκής αλλά δεν είναι επεκτάσιμη καθώς πρέπει να πετύχουμε τον διαχωρισμό του front-end από το back-end.
- **Δημιουργία νέας διαδικασίας αυθεντικοποίησης**
Για την νέα εφαρμογή τα στοιχεία της αυθεντικοποίησης δεν αποθηκεύονται σε session στον server, αλλά επιστρέφεται στον client ένα token της μορφής του πρωτοκόλλου JSON Web Token (RFC 7519). Για τον σκοπό αυτό θα πρέπει να δημιουργηθεί νέα εφαρμογή στο login.tee.ihu.gr.
- **Διαχωρισμός της παρούσας εφαρμογής από την νέα εφαρμογή**
Για την νέα εφαρμογή θα δημιουργηθούν νέα αντικείμενα κρατώντας την προηγούμενη υλοποίηση άθικτη.
- **Αποστολή δεδομένων με την μορφή icalendar**
Για να συγχρονιστούν τα events που περιέχονται στην εφαρμογή με τρίτες εφαρμογές θα πρέπει να υλοποιηθεί μία διαδικασία μετατροπής των δεδομένων σε μορφή icalendar (RFC 5545).

Front-end

- **Βελτιστοποίηση διαδικασίας συγγραφής ανακοίνωσης**
Στην νέα εφαρμογή η ανακοίνωση η συγγραφή ανακοίνωσης θα περιέχει τις παρακάτω λειτουργίες.
- Την μετατροπή κειμένου σε bold, italic, underline
- Την δημιουργία επικεφαλίδων (H1, H2 και H3)
- Την δημιουργία λιστών
- Την προσθήκη φωτογραφιών μέσω επικόλλησης και επικόλλησης συνδέσμου
- Την αλλαγή μεγέθους των φωτογραφιών
- **Αλλαγή στον τρόπο αναζήτησης**
Το μενού είναι κρυμμένο αρχικά και εμφανίζεται μόνο όταν ο χρήστης το επιλέξει. Επίσης οι παράμετροι αναζήτησης εισάγονται σαν URI και όχι σαν GET μεταβλητές.
- **Λειτουργία δύο θεμάτων**
Η εφαρμογή περιέχει φωτεινή και σκοτεινή εμφάνιση.
- **Λειτουργία δύο γλωσσών**
Για λόγους προσβασιμότητας η εφαρμογή θα πρέπει να υποστηρίζει την εναλλαγή γλώσσας με τρόπο κατανοητό προς τον χρήστη.

1.5 Μη λειτουργικές απαιτήσεις εφαρμογής

Για το συγκεκριμένο έργο η μη λειτουργικές απαιτήσεις αφορούν κατά κόρον απαιτήσεις που προκύπτουν σε συνδυασμό με το προηγούμενο σύστημα και θα αναλυθούν παρακάτω. Πάλι για την καλύτερη κατανόηση των απαιτήσεων χρειάζεται να γίνει ο διαχωρισμός σε back-end και front-end.

Back-end

- **Ασφάλεια**

Καθώς η νέα υλοποίηση απαιτεί την εκ νέου ανάπτυξη ενός συστήματος αυθεντικοποίησης θα πρέπει να βεβαιωθούμε για την ασφάλεια των δεδομένων. Συγκεκριμένα θα πρέπει να διασφαλίσουμε την ασφάλεια, την ιδιωτικότητα και την ακεραιότητα των δεδομένων. Ένας χρήστης για να έχει πρόσβαση σε συγκεκριμένες πτυχές της εφαρμογής θα πρέπει να αυθεντικοποιηθεί μέσω του συστήματος αυθεντικοποίησης του τμήματος.

- **Legacy support**

Η παρούσα εφαρμογή θα πρέπει να είναι προσβάσιμη και λειτουργική ακόμα και μετά το ανέβασμα σε παραγωγικό περιβάλλον της νέας εφαρμογής. Αυτό σημαίνει πως η ανάπτυξη της νέας εφαρμογής θα πρέπει να γίνει με τρόπο μη παρεμβατικό και διαχωρισμένο από το τωρινό σύστημα. Αυτό επιτυγχάνεται με τον διαχωρισμό των υλοποιήσεων εσωτερικά σε V1 και V2 αντίστοιχα. Έτσι, τα εξαρτώμενα συστήματα από την τωρινή εφαρμογή θα παραμείνουν λειτουργικά και η νέα εφαρμογή με την σουίτα των end-points εξάγεται από διαφοροποιημένο URI.

- **Επεκτασιμότητα**

Η εφαρμογή θα πρέπει να ακολουθεί πρότυπα και αρχιτεκτονική η οποία να διευκολύνει την μελλοντική επέκταση χωρίς την ανάγκη ριζικών αλλαγών

- **Αξιοπιστία**

Η εφαρμογή θα πρέπει να περιέχει όσο το δυνατόν λιγότερα σφάλματα και στην περίπτωση σφάλματος να υπάρχει ένας μηχανισμός διαχείρισης ώστε να διευκολύνει την διαδικασία αποσφαλμάτωσης

- **Απόδοση**

Η απόδοση μίας εφαρμογής κρίνεται από το κατά πόσο ένας χρήστης μπορεί να ολοκληρώσει μία ενέργεια καθώς και από τις καθυστερήσεις που ενδεχομένως αντιμετωπίσει. Η εφαρμογή θα πρέπει να λαμβάνει υπόψιν τους λόγους καθυστέρησης και να γίνουν οι απαραίτητες αλλαγές ώστε να επιτυγχάνεται ο καλύτερος χρόνος απόκρισης.

Front-end

- **Μετατροπή εφαρμογής από MPA σε SPA**

Ένα βασικό μειονέκτημα μίας MPA εφαρμογής είναι το γεγονός ότι μετά από κάθε ανακατεύθυνση του χρήστη θα πρέπει να επαναφορτωθεί όλη η προβολή, ακόμη και τα κομμάτια που δεν θα επηρεαζόταν από αυτή την αλλαγή όπως το header και το footer. Στις SPA εφαρμογές η φόρτωση της εφαρμογής γίνεται μόνο μία φορά και όλες οι μελλοντικές πληροφορίες λαμβάνονται με την χρήση AJAX αιτημάτων. Αυτό ανεβάζει αρκετά την απόδοση της εφαρμογής καθώς αποφεύγεται η αίτηση και επανεκτύπωση δεδομένων που δεν επηρεάζονται από κάποια ανακατεύθυνση ή αλλαγή.

1.6 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκε η έννοια των απαιτήσεων ξεχωρίζοντας τις δύο κατηγορίες απαιτήσεων σε λειτουργικές και μη λειτουργικές. Στην συνέχεια έγινε αναφορά στην υπάρχουσα εφαρμογή ώστε να γίνουν κατανοητές οι αλλαγές που έχουν υλοποιηθεί. Τέλος, αναλύθηκαν οι λειτουργικές και μη λειτουργικές απαιτήσεις που θα πρέπει να ικανοποιηθούν ώστε να ολοκληρωθεί το έργο του επανασχεδιασμού της εφαρμογής.

Κεφάλαιο 2ο: Εισαγωγή στις τεχνολογίες του έργου

2.1 Εισαγωγή

Για την ανάπτυξη της εφαρμογής μελετήθηκαν τεχνολογίες οι οποίες θα κάλυπταν κάθε πτυχή της εφαρμογής καθώς και η υπάρχουσα υλοποίηση και υποδομή ώστε να μην υπάρξει μεγάλη απόκλιση και να επιτευχθεί μία ομαλή μετάβαση στην νέα έκδοση. Παρακάτω θα αναλύσουμε τις τεχνολογίες που επιλέχθηκαν με αναφορά σε παρόμοιες τεχνολογίες ως μέτρο σύγκρισης.

2.2 PHP

Η PHP είναι μία γλώσσα προγραμματισμού η οποία χρησιμοποιείται κυρίως για διαδικτυακές εφαρμογές και είναι μια από τις πιο διαδεδομένες γλώσσες προγραμματισμού για αυτές τις εργασίες. Συγκεκριμένα μέσω τις PHP παρέχεται η δυνατότητα υλοποίησης ιστοσελίδων δυναμικού περιεχομένου που μπορούν να αλληλοεπιδρούν με μία βάση δεδομένων ή άλλα συστήματα.

2.2.1 Ιστορική αναδρομή της PHP

Η PHP δημιουργήθηκε το 1994 από τον Rasmus Lerdorf ο οποίος υλοποίησε και τις δύο πρώτες εκδόσεις της. Τα αρχικά του ακρωνύμιου της PHP κατά την πρώτη υλοποίηση σήμαιναν «Personal Home Page». Η PHP ξεκίνησε ως ένα έργο ανοιχτού κώδικα όπου και παραμένει έως και σήμερα, βασισμένο στην γλώσσα C. Το 1995 δημοσιοποιήθηκε και η πρώτη έκδοση της PHP με όνομα PHP Tools. Εκείνο τον καιρό οι τεχνολογίες ιστού ήταν ακόμη σε βρεφικό στάδιο και έτσι η γλώσσα βρήκε πρόσφορο έδαφος και κοινό για να αναπτυχθεί. Το 1996 η PHP δέχθηκε μία ραγδαία αλλαγή στον πηγαίο κώδικα με την νέα έκδοση με όνομα PHP/FI. Ωστόσο ο τρόπος που λειτουργούσε ήταν πολύ διαφορετικός από αυτό που γνωρίζουμε σήμερα. Η πρώτη έκδοση που παρείχε μία δόση από τα σημερινά standards ήταν η PHP 3 η οποία δημιουργήθηκε από τον Andi Gutmans και Zeev Suraski σε συνεργασία με τον δημιουργό Rasmus Lerdorf.[2]

2.2.2 Οι δυνατότητες της PHP

Η PHP σήμερα απέχει πολύ από την εικόνα που είχαμε πριν 19 χρόνια με δυνατότητες αντικειμενοστραφούς συγγραφής και ασφάλειας στο ίδιο επίπεδο με τις μεγάλες γλώσσες στον χώρο, όπως την Java. Επιπρόσθετα, η κοινότητα της PHP έχει δημιουργήσει πληθώρα frameworks τα οποία βελτιώνουν τους χρόνους υλοποίησης σε μικρά, μεσαία και μεγάλα έργα όπως το WordPress και Joomla για μικρά και μεσαία έργα καθώς και την Laravel για μεσαία και μεγάλα έργα. Η PHP μας δίνει την δυνατότητα να υλοποιήσουμε λειτουργίες με απλό τρόπο και σύνταξη, κάτι που δεν υφίσταται σε γλώσσες και frameworks όπως η Java και το .NET.

2.3 Laravel

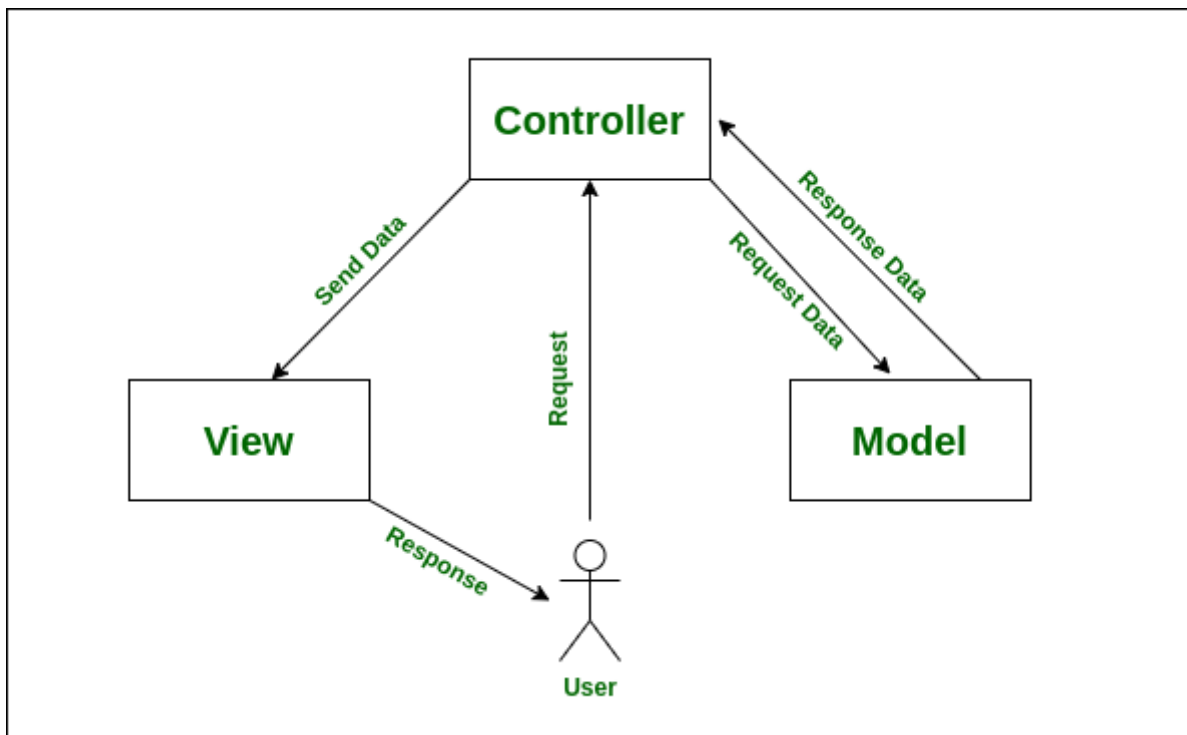
Η Laravel αποτελεί ένα framework βασισμένο σε PHP που υλοποιεί την αρχιτεκτονική MVC την οποία θα καλύψουμε σε επόμενο κεφάλαιο. Η Laravel χρησιμοποιείται κυρίως για υλοποίηση διαδικτυακών εφαρμογών ενσωματώνοντας μεγάλη γκάμα πακέτων που παρέχουν λύσεις για απαιτήσεις κοινές σε διαδικτυακές εφαρμογές όπως η σύνδεση και η εγγραφή χρηστών με ασφαλή τρόπο. Μέσω της Laravel ο προγραμματιστής μπορεί να ξεκινήσει την υλοποίηση της εφαρμογής του έχοντας ήδη έτοιμες τις βασικές λειτουργίες όπως την δρομολόγηση, την αυθεντικοποίηση χρηστών και την ασφάλεια.[3]

2.3.1 Ιστορική αναδρομή της Laravel

Η Laravel ξεκίνησε το 2011 ως αντικαταστάτης του τότε πολυχρησιμοποιημένου framework CodeIgniter για να «διορθώσει» το πρόβλημα των ελλείψεων όπως την διαδικασία αυθεντικοποίησης των χρηστών. Από τότε μέχρι σήμερα έχουν αλλάξει πολλά στο framework με τις μεγαλύτερες αλλαγές να φαίνονται το 2013 με την έκδοση Laravel 4 όπου αλλάζει αρκετά το συντακτικό, προστίθεται build-in τρόπος για testing και η υποστήριξη για πολλαπλά συστήματα αρχείων[4]. Ωστόσο δεν ήταν λίγες οι προσθήκες και οι βελτιώσεις στην έκδοση 6 με κάποιες από αυτές να παρέχουν λύσεις authorization, καλύτερη διαχείριση στα middlewares και lazy collections για γρήγορη προσπέλαση σε μεγάλο όγκο δεδομένων. Σήμερα η Laravel βρίσκεται στην έκδοση 10 και εκμεταλλεύεται τις βελτιώσεις της PHP 8 υλοποιώντας ένα strict σύστημα και παρέχοντας ακόμα περισσότερες λύσεις σε γνωστά προβλήματα στην ανάπτυξη εφαρμογών.

2.3.2 Η αρχιτεκτονική MVC

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, η Laravel υλοποιεί την αρχιτεκτονική MVC. Η αρχιτεκτονική MVC χωρίζει την εφαρμογή σε 3 μεγάλα λογικά κομμάτια. Το model που είναι υπεύθυνο για την μοντελοποίηση της βάσης και των σχέσεων της, το view που είναι η διεπαφή χρήστη και ο controller που συνδέει τα models με τα views και κατ' επέκταση τα αιτήματα του χρήστη με το εσωτερικό της εφαρμογής.



Σχήμα 2.1 Η αρχιτεκτονική MVC

2.3.2.1 Model

Όπως αναφέρθηκε παραπάνω το μοντέλο μετατρέπει τις οντότητες και τις σχέσεις της βάσης δεδομένων σε αντικείμενα. Συγκεκριμένα τα μοντέλα θα πρέπει να υλοποιούν τις παρακάτω λειτουργίες[5]:

- Μεταφορά της πληροφορίας
- Μετατροπή της πληροφορίας σε δομές δεδομένων χρήσιμες για την εφαρμογή

- Αποθήκευση της πληροφορίας
- Επεξεργασία της πληροφορίας
- Διαγραφή της πληροφορίας

Με τον όρο πληροφορία εννοούμε το σύνολο των δεδομένων που απαρτίζουν μία οντότητα. Αν πάρουμε για παράδειγμα την οντότητα «Μαθητής», αυτή η οντότητα περιέχει τα στοιχεία του μαθητή (Όνομα, Επώνυμο κ.α.), τα μαθήματα που παρακολουθεί ο μαθητής, τα μαθήματα που παρακολούθησε επιτυχώς ο μαθητής καθώς και τις βαθμολογίες, το ιστορικό σε κάθε μάθημα και λοιπά. Για όλα τα παραπάνω δεδομένα υπάρχουν ξεχωριστοί πίνακες οι οποίοι έχουν κάποια σχέση μεταξύ τους. Αυτούς τους πίνακες και τις σχέσεις τους τα μοντέλα μετατρέπουν σε μία ενοποιημένη οντότητα προσπελάσιμη και επεξεργάσιμη από την εφαρμογή.

2.3.2.2 View

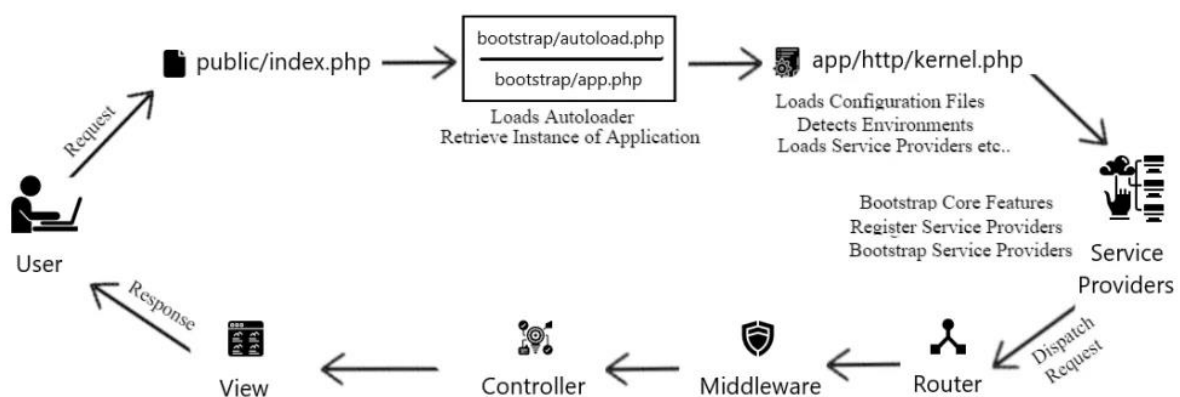
Οι προβολές (views) είναι υπεύθυνες για το πως τα δεδομένα θα προβληθούν στον τελικό χρήστη καθώς και για τις διεπαφές του χρήστη με την εφαρμογή. Τα δεδομένα της κάθε προβολής προέρχονται από κάποιον controller και συνήθως μεταποιούνται σε HTML. Η Laravel έχει υλοποιήσει ένα σύστημα προτύπων (templates) όπου μία προβολή θέτει την διάταξη της σελίδας και άλλες μικρότερες προβολές γεμίζουν τις ενότητες (sections) της διάταξης με τα δεδομένα. Το παραπάνω σύστημα ονομάζεται blade και με τον τρόπο που υλοποιείται βοηθάει στην επαναχρησιμοποίηση προτύπων και ενότητων.

2.3.2.3 Controller

Οι ελεγκτές (controllers) είναι αυτοί που ενώνουν τα views με τα models καθώς τα αιτήματα των χρηστών καταλήγουν εκεί για να επεξεργαστούν και μετά από την επικοινωνία με τα αναγκαία models να διαμορφωθεί η εκάστοτε απάντηση προς τον χρήστη, δίνοντας δεδομένα σε κάποιο view. Στην Laravel οι controllers είναι εξοπλισμένοι με διεπαφές Request και Response για να κάνουν την διαδικασία ευκολότερη.

2.3.3 Laravel lifecycle

Παρακάτω θα εξετάσουμε πως λειτουργεί η Laravel από την εκκίνηση ενός αιτήματος μέχρι της επιστροφή μιας απάντησης.[6]



Σχήμα 2.2 Laravel lifecycle

Σε κάθε αίτημα εκτελούνται τα παρακάτω:

- Εκτελείται το αρχείο `public/index.php` το οποίο περιέχει εντολές ενσωμάτωσης του υπόλοιπου framework της Laravel μέσω του αρχείου `bootstrap/autoload.php`.
- Έπειτα φορτώνεται ο πυρήνας (kernel) του συστήματος. Η Laravel περιέχει δύο τύπους πυρήνα. Τον HTTP Kernel που αφορά web requests και τον Console kernel που αφορά αιτήσεις μέσω κονσόλας. Για παράδειγμα ένα console command που θα χρησιμοποιήσει τον console kernel είναι το `php artisan migrate`.
- Στην συνέχεια ο πυρήνας φορτώνει τις απαραίτητες υπηρεσίες που θα χρειαστούν για τις υπόλοιπες λειτουργίες στην πορεία.
- Οι αιτήσεις μεταφέρονται στον εκάστοτε δρομολογητή. Η Laravel περιέχει πολλούς τύπους δρομολογητών όπως τον web και τον api. Μπορούμε επίσης να δημιουργήσουμε και άλλους προσαρμοσμένους δρομολογητές, όπως θα δούμε στην πορεία, για τις ανάγκες της εφαρμογής.
- Ο δρομολογητής αφού εντοπίσει μέσω του URI τον controller που είναι υπεύθυνος για την αίτηση, την μεταφέρει εκεί. Ωστόσο κατά την μεταφορά εκτελείται κατά περιπτώσεις το middleware που περιέχει διαδικασίες ασφαλείας όπως για παράδειγμα την αυθεντικοποίηση του χρήστη.
- Ο controller με την σειρά του εκτελεί λογικές διαδικασίες για την διεκπεραίωση της αίτησης όπως το να λάβει/αποστείλει δεδομένα σε κάποιο model και να μεταφέρει δεδομένα προς κάποιο view.
- Τα δεδομένα μεταφέρονται στο επιλεγμένο από τον controller view τα οποία μετατρέπονται σε προβολή και συμπεριλαμβάνονται στην απάντηση προς τον χρήστη.

2.3.4 Γιατί Laravel

Η Laravel αυτή την στιγμή θεωρείται μία από τις καλύτερες επιλογές για ανάπτυξη διαδικτυακών εφαρμογών καθώς έχει υιοθετήσει αρκετές καλές πλευρές από τους προπάτορές της όπως το CodeIgniter την symphony κ.α.. Η Laravel έχει μία από τις μεγαλύτερες κοινότητες σε σχέση με άλλα MVC Frameworks και φαίνεται ότι η ομάδα πίσω από το έργο είναι ενεργή κάνοντας αρκετές αναβαθμίσεις σε τακτά χρονικά διαστήματα[7]. Μέσω της Laravel επιτυγχάνεται η ανάπτυξη PHP εφαρμογών ραγδαία, με ασφάλεια και με μεγάλο βαθμό επεκτασιμότητας λόγω της αρχιτεκτονικής και των προτύπων σχεδίασης που ακολουθούνται.

2.4 Composer

Το composer είναι ένα εργαλείο διαχείρισης των εξαρτήσεων από βιβλιοθήκες ή πακέτα για ένα PHP έργο[8]. Αυτό σημαίνει ότι με την χρήση του composer μπορούμε να έχουμε μία αναφορά των πακέτων που χρειάζονται για να λειτουργήσει η εφαρμογή μας και με ευκολία να τροποποιήσουμε αυτές τις εξαρτήσεις από τα πακέτα. Το composer λειτουργεί παρόμοια με την εντολή `apt` που θα βρούμε σε συστήματα linux μόνο που τα εγκατεστημένα πακέτα ορίζονται ανά project και όχι καθολικά στο σύστημα. Αξίζει να σημειωθεί ότι το composer λειτουργεί σε οποιοδήποτε λειτουργικό σύστημα υποστηρίζει την χρήση PHP γιατί και το ίδιο είναι βασισμένο σε PHP. Επίσης με το composer μπορούμε να θέσουμε εκτός από εξαρτήσεις, και φραγμούς για ένα έργο όπως την έκδοση PHP που θα πρέπει να χρησιμοποιηθεί και την έκδοση των υπολοίπων πακέτων που χρησιμοποιούνται στο έργο. Πολλά νέα PHP frameworks εκμεταλλεύονται το composer γιατί προσφέρει έναν γρήγορο τρόπο εγκατάστασης και εκκίνησης, διασφαλίζοντας ότι θα υπάρχουν τα προαπαιτούμενα στοιχεία στο σύστημα όπως η έκδοση PHP.

2.4.1 Χρήση του composer

Η χρήση του composer βασίζεται κυρίως σε δύο αρχεία, το `composer.json` και το `composer.lock`. Έστω ότι έχουμε δημιουργήσει το παρακάτω `composer.json` αρχείο.

```
{
  "require": {
    "monolog/monolog": "2.0.*"
  }
}
```

Κώδικας 2.1 Παράδειγμα αρχείου `composer.json`

Όπως φαίνεται στο παραπάνω αρχείο με την λέξη κλειδί `require` δηλώνονται τα πακέτα που θα πρέπει να προστεθούν στο έργο. Στην περίπτωσή μας το πακέτο είναι το `monolog` και ο περιορισμός στην έκδοση είναι η τελευταία έκδοση του 2.0. Ο αστερίσκος χρησιμοποιείται ως μπαλαντέρ και δηλώνει πως θα πρέπει να ληφθεί η τελευταία έκδοση που να είναι μεγαλύτερη ή ίση με την 2.0.0 και μικρότερη της 2.1. Για να εγκαταστήσουμε τοπικά τα δηλωμένα πακέτα θα πρέπει να τρέξουμε την εντολή:

```
php composer.phar update
```

Αυτή η εντολή αρχικά θα αναλύσει τα δηλωμένα πακέτα ώστε να βρει την θέση τους και την τελευταία έκδοση σύμφωνα με τους περιορισμούς. Έπειτα θα δημιουργήσει το αρχείο `composer.lock` που θα περιέχει τις πραγματικές εκδόσεις των πακέτων. Στο τέλος θα εκτελέσει την διαδικασία `install` όπου θα κατεβάσει τα αρχεία των πακέτων σε έναν φάκελο (συνήθως τον φάκελο `vendor`) και θα δημιουργήσει το αρχείο `vendor/autoload.php`. Το αρχείο `vendor/autoload.php` αυτοματοποιεί την διαδικασία της φόρτωσης των αρχείων για κάθε βιβλιοθήκη, οπότε θα πρέπει να προστίθεται στην αρχή του `php script` μας για να χρησιμοποιήσουμε τις εγκατεστημένες βιβλιοθήκες χωρίς περεταίρω προσθήκες αρχείων. Για να προσθέσουμε νέα βιβλιοθήκη στην λίστα με τα `requires` μπορούμε απλά να τρέξουμε την εντολή:

```
php composer.phar require new/package ~2.1.*
```

και το `composer` αυτόματα θα προσθέσει το πακέτο στην λίστα του `composer.json` και θα το εγκαταστήσει τοπικά.

2.4.2 Γιατί composer

Αυτή την στιγμή το `composer` δεν έχει κάποια άμεσα ανταγωνιστική πλατφόρμα στον συγκεκριμένο σκοπό που εκτελεί. Με το `composer` μπορεί κανείς να διαχειριστεί τα πακέτα ενός project εύκολα και γρήγορα ασχέτως το πλήθος των ατόμων που εργάζονται πάνω στο έργο, αφού ο καθένας θα ενημερώνει και θα ενημερώνεται μέσω των αλλαγών στα αρχεία `composer.json` και `composer.lock` χωρίς να προβεί σε χειροκίνητες διαδικασίες. Παρακάτω θα παρουσιάσουμε κάποιες από τις πιο συνηθισμένες εντολές του `composer`.

Εντολή	Περιγραφή
<code>composer update</code>	Ορίζει την τελευταία δυνατή έκδοση του κάθε πακέτου, δημιουργεί το <code>composer.lock</code> και κατεβάζει τις νέες εκδόσεις (αν υπάρχουν) για τα πακέτα

<code>composer update vendor-name/package-name</code>	Αναβαθμίζει μόνο το δηλωθέν πακέτο
<code>composer require vendor-name/package-name</code>	Ορίζει το δηλωθέν πακέτο στο αρχείο <code>composer.json</code> και το κατεβάζει τοπικά
<code>composer outdated</code>	Τυπώνει όλα τα πακέτα που έχουν ξεπερασμένη έκδοση
<code>composer remove vendor-name/package-name</code>	Αφαιρεί το πακέτο από τις εξαρτήσεις και το διαγράφει
<code>composer install</code>	Κατεβάζει όλα τα εξαρτημένα πακέτα

Πίνακας 2.1 Εντολές composer

2.5 NPM

Όπως το composer έτσι και το NPM είναι ένα σύστημα διαχείρισης και διανομής βιβλιοθηκών JavaScript. Μέσω του NPM μπορούμε να διαχειριστούμε τις εξαρτήσεις από βιβλιοθήκες ενός project. Το NPM είναι γραμμένο σε JavaScript και έχει ως προαπαιτούμενο το NodeJS που επίσης είναι γραμμένο σε JavaScript. Εφόσον έχουμε εγκαταστήσει το NPM για να το εκκινήσουμε σε κάποιο project θα χρησιμοποιήσουμε την εντολή

```
npm init <όνομα project>
```

Αυτή η εντολή θα δημιουργήσει ένα αρχείο `package.json` όπου σε αυτό θα αναγράφονται οι βιβλιοθήκες που χρησιμοποιούνται στο project καθώς και γενικότερες πληροφορίες για το project όπως το όνομα, η έκδοση και ο author. Παρακάτω θα δούμε ένα παράδειγμα ενός τέτοιου αρχείου.

```
{
  "name": "abroad-ui",
  "version": "0.1.0",
  "dependencies": {
    "axios": "^1.1.3",
  }
}
```

Κώδικας 2.2 Παράδειγμα αρχείου package.json

Στο παραπάνω παράδειγμα βλέπουμε τις πληροφορίες όνομα(name), έκδοση(version), εξαρτήσεις από πακέτα(dependencies) και πιο συγκεκριμένα την εξάρτηση από την βιβλιοθήκη axios. Για να λάβουμε αυτά τα πακέτα θα πρέπει να χρησιμοποιήσουμε την εντολή

```
npm install
```

Αυτή η εντολή θα κατεβάσει όλα τα απαραίτητα πακέτα στις εκδόσεις που δηλώνονται στο αρχείο και θα δημιουργήσει το αρχείο `package.lock`. Αυτό το αρχείο χρησιμοποιείται κυρίως για ομαδικά project όπου θα πρέπει όλοι στην ομάδα να κατέχουν ακριβώς τις ίδιες εκδόσεις για τα εξαρτώμενα πακέτα, ακριβώς όπως στην περίπτωση του composer που εξετάσαμε στο προηγούμενο κεφάλαιο. Αν θέλουμε να εισάγουμε νέα βιβλιοθήκη αρκεί να τρέξουμε την εντολή

```
npm install <όνομα βιβλιοθήκης>
```

και αυτομάτως θα εισαχθεί η βιβλιοθήκη στο `package.json` και θα κατέβει τοπικά. Όλες οι βιβλιοθήκες που λαμβάνονται μέσω του npm υπάρχουν στον φάκελο `node_modules` όπου βρίσκεται στο ίδιο path με το `package.json`. Τέλος, πολλές βιβλιοθήκες μπορεί να περιέχουν δικά τους `package.json` επειδή

έχουν δικές τους εξαρτήσεις από άλλες βιβλιοθήκες. Αυτές τις εξαρτήσεις τις διαχειρίζεται το npm αυτόματα. Παρακάτω θα δούμε κάποιες από τις βασικές εντολές του npm προσθέτοντας μία εξήγηση για την κάθε μία.

Εντολή	Περιγραφή
npm init	Αρχικοποιεί το npm δημιουργώντας το αρχείο package.json
npm install ή npm i	Κατεβάζει όλα τα εξαρτώμενα πακέτα του project
npm install <package> ή npm I <package>	Θέτει το πακέτο στις εξαρτήσεις και το κατεβάζει
npm uninstall <package> ή npm un <package>	Αφαιρεί το πακέτο από τις εξαρτήσεις και το διαγράφει
npm run	Τυπώνει όλα τα διαθέσιμα script που μπορούν να τρέξουν
npm run <script>	Τρέχει το script
npm search <package>	Κάνει αναζήτηση για το συγκεκριμένο πακέτο
npm view <package>	Τυπώνει πληροφορίες για το συγκεκριμένο πακέτο
npm ls	Τυπώνει όλες τις εξαρτήσεις το project

Πίνακας 2.2 Εντολές npm

2.6 HTML/CSS/JavaScript

Με την χρήση HTML, CSS και JavaScript μπορούμε να αναπτύξουμε ιστοσελίδες οι οποίες να είναι διαδραστικές και προσβάσιμες από μία μεγάλη γκάμα συσκευών. Παρακάτω θα αναλυθεί η κάθε τεχνολογία ξεχωριστά.

2.6.1 HTML

Η HTML (Hyper Text Markup Language) είναι μία γλώσσα σήμανσης η οποία χρησιμοποιείται για την ανάπτυξη ιστοσελίδων. Συγκεκριμένα με την χρήση HTML περιγράφεται η δομή ενός εγγράφου από τον τίτλο, τις παραγράφους και τις ενότητες μέχρι τις λίστες και τους υπερσυνδέσμους. Αυτό επιτυγχάνεται με την χρήση ετικετών που περικλείουν ένα κείμενο για να του αποδώσουν μία ιδιότητα. Η HTML εμφανίστηκε για πρώτη φορά στον χώρο το 1991 με όνομα HTML Tags και από τότε έχουν δημοσιευτεί αρκετές εκδόσεις με την τελευταία να είναι η HTML 5 το 2006 η οποία παραμένει έως σήμερα[9].

2.6.2 CSS

Η CSS με την σειρά της είναι και αυτή μία γλώσσα σήμανσης όπως η HTML και χρησιμοποιείται για να παρέχει κανόνες για τον τρόπο εμφάνισης των HTML στοιχείων. Επίσης οι ίδιοι κανόνες μπορούν να γραφτούν απευθείας πάνω στο στοιχείο με την χρήση της ετικέτας στοιχείου style, όπως και μέσα στο αρχείο HTML μέσα στην ετικέτα style.

2.6.3 JavaScript

Η JavaScript είναι μία γλώσσα προγραμματισμού σεναρίου η οποία έχει εξελιχθεί ώστε να υποστηρίζει αντικειμενοστραφή αρχιτεκτονική. Η πρώτη έκδοση της JavaScript δημοσιεύτηκε τον Σεπτέμβριο του 1995 από τις εταιρίες Netscape και Sun Microsystems με αρχικό όνομα LiveScript και στην συνέχεια JavaScript 1.0. Ωστόσο παρότι η λέξη java περιέχεται στον τίτλο της γλώσσας, οι ομοιότητες των δύο γλωσσών είναι πάρα πολύ λίγες[10]. Σήμερα, η JavaScript χρησιμοποιείται από το μεγαλύτερο ποσοστό των ιστοσελίδων και είναι μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού στον κόσμο. Για τις ιστοσελίδες η JavaScript χρησιμοποιείται για να ενσωματώσει διαδραστικές λειτουργίες όπως συμβάντα σε HTML στοιχεία με την χρήση του DOM. Το DOM είναι ένα αντικείμενο το οποίο καταγράφει το έγγραφο HTML με την δομή δένδρου. Μέσω του DOM μπορούμε να προσπελάσουμε και να μεταβάλλουμε τα HTML στοιχεία με τον τρόπο που επιθυμούμε κατά περίπτωση. Χωρίς την JavaScript οι ιστοσελίδες θα ήταν απλά κείμενα με μόνη δυνατότητα τον υπερσύνδεσμο και τίποτε άλλο.

2.7 React.js

Η react ή αλλιώς ReactJS είναι ένα front-end framework βασισμένο σε JavaScript που χρησιμοποιείται για ανάπτυξη διεπαφών χρήστη. Η ReactJS αρχικά κατασκευάστηκε από την εταιρία Meta (πρώην Facebook) και συγκεκριμένα από τον Jordan Walke. Η πρώτη έκδοση της ReactJS ήταν το FaxJS[11] που ήταν ένα πρωτότυπο framework και χρησιμοποιήθηκε από το facebook το 2012 και το Instagram το 2013. Με την ReactJS αναπτύσσονται SPA εφαρμογές οι οποίες χρησιμοποιούν αντί για το DOM, ένα Virtual DOM για την ενημέρωση των διεπαφών. Το πλεονέκτημα του Virtual DOM είναι ότι μέσω αυτού μπορούν να ανανεώνονται μόνο συγκεκριμένα κομμάτια της σελίδας, πράγμα που μειώνει δραματικά τους χρόνους απόκρισης και τον συνολικό φόρτο εκτέλεσης[12]. Παρακάτω θα αναλύσουμε τον τρόπο που λειτουργεί η ReactJS και γιατί βρίσκεται στην υψηλότερη θέση στις προτιμήσεις των προγραμματιστών. Για την περεταίρω ανάλυση θα χρησιμοποιήσουμε το παρακάτω παράδειγμα κώδικα της React.

```
import React, { Fragment, useState } from "react";
const Parent = () => {
  const [clickCount, setClickCount] = useState(0)
  const incrementClickCount = () => {
    setClickCount(clickCount + 1)
  }
  return (
    <Fragment>
      <Child title="I am a child component"
onButtonClick={incrementClickCount} />
      <span className="click-counter">{clickCount}</span>
    </Fragment>
  )
}

const Child = (props) => {
  return (<Fragment>
    <h3 className={'title'}>{props.title}</h3>
    <button onClick={props.onButtonClick}>Click me!</button>
  </Fragment>)
}
export default Parent;
```

Κώδικας 2.3 Παράδειγμα κώδικα σε react

2.7.1 Components

Το κύριο συστατικό της ReactJS είναι τα Components. Ένα component είναι στην ουσία ένα κομμάτι διεπαφής χρήστη και εμπεριέχει την κατάσταση του (state), τις μεθόδους του, τις ιδιότητες του (props) και μία μέθοδο render. Στο render δηλώνονται με JSX τα κομμάτια του component μαζί με τιμές των ιδιοτήτων και της κατάστασης. Κάθε component μπορεί να απαρτίζεται από άλλα, μικρότερα components που με την σειρά τους θα έχουν δικό τους state. Οι ιδιότητες κάθε component προέρχονται από το σημείο που ορίζεται το component [13]. Αναλύοντας το παράδειγμα κώδικα που ορίζει δύο components, το Parent και το Child, η ιδιότητα title που χρησιμοποιείται στο Component Child προέρχεται από το component Parent όπου και ορίζεται το Child. Με αυτόν τον τρόπο επιτυγχάνεται η μετάδοση της πληροφορίας από πάνω προς τα κάτω. Αξίζει να σημειωθεί ότι μία ιδιότητα μπορεί να έχει ως τιμή και μία μέθοδο ή ένα αντικείμενο. Έτσι μπορούμε να μεταφέρουμε κάποια πληροφορία από κάτω προς τα πάνω μέσω κάποιων events. Στο παραπάνω παράδειγμα θέσαμε ως ιδιότητα το onClick που εμπεριέχει την μέθοδο incrementClickCount. Στο Child θέσαμε σαν callback του button onclick το props.onClick. Οπότε κάθε φορά που κλικάρεται το κουμπί του Child ενημερώνεται το state του Parent.

2.7.2 State

Στην ReactJS το state ορίζει την κατάσταση στην οποία βρίσκεται το component κάθε φορά. Κάθε αλλαγή στο state ενός component ενεργοποιεί και ένα νέο render. Έτσι μπορούμε να ενημερώνουμε μόνο τα σημεία των προβολών που έχουν αλλάξει, χωρίς να χρειάζεται να ενημερώσουμε κάποιο άλλο σημείο του DOM. Επίσης θα πρέπει να τονιστεί ότι οι αλλαγές στο state εκτελούνται ασύγχρονα. Πιο συγκεκριμένα ο κύκλος εκτέλεσης ενός component γίνεται με την εξής σειρά:

- Αρχικοποιούνται οι τιμές και οι μέθοδοι αλλαγής τιμής του state μέσω του hook useState.
- Κάθε αλλαγή σε μία τιμή του state μπαίνει σε μια ουρά εκτέλεσης σύμφωνα με την σειρά εκτέλεσης που πραγματοποιείται στον κώδικα.
- Εκτελείται ασύγχρονα κάθε αλλαγή του state.
- Εκτελείται re-render του component.

Συνοψίζοντας, το state λειτουργεί σαν στιγμιότυπο διεπαφής χρήστη ενός component όπου στην κάθε αλλαγή επιστρέφει μία ανανεωμένη έκδοση αυτής της διαδραστικής διεπαφής [13].

2.7.3 Hooks

Όπως αναφέραμε στο παραπάνω κεφάλαιο, για τον ορισμό μιας τιμής state χρησιμοποιήσαμε το hook useState. Τα hooks στην React είναι μέθοδοι που επιτρέπουν να ενώσουμε ιδιότητες της React (όπως το state) στο component. Η React παρέχει αρκετά ενσωματωμένα hooks για να διευκολύνει την διαδικασία ανάπτυξης μιας εφαρμογής. Το hook useState είναι ένα από αυτά και χρησιμοποιείται για να εισάγουμε την διαδικασία του state στο component επιστρέφοντας την μεταβλητή state και την μέθοδο επεξεργασίας της τιμής. Στο παραπάνω παράδειγμα υλοποιείται η τιμή state clickCount και μέσω της αποδόμησης πίνακα που παρέχεται από την JS λαμβάνεται από το useState η μεταβλητή clickCount και η μέθοδος επεξεργασίας setClickCount. Με αυτό τον τρόπο «ενώνεται» το state με το υπόλοιπο component και σε κάθε αλλαγή του state επιστρέφεται ενημερωμένο το JSX μέσω του re-render που εκτελείται.

2.7.4 Γιατί ReactJS

Η React έχει αλλάξει ριζικά τον τρόπο ανάπτυξης front-end εφαρμογών από την απλή σύνταξη HTML /CSS σε μία ιεραρχική διαδικασία με στόχο την απόδοση, την επαναχρησιμοποίηση και την επεκτασιμότητα της εφαρμογής [12]. Μέσω της React μας δίνεται η δυνατότητα να υλοποιήσουμε μεγάλης κλίμακας εφαρμογές σε μικρό χρονικό διάστημα παρέχοντας τα εργαλεία και τις διαδικασίες που βελτιώνουν τον τρόπο με τον οποίο αναπτύσσονται οι εφαρμογές. Εκτός από την React υπάρχουν και άλλες τεχνολογίες οι οποίες προσπαθούν να λύσουν τα ίδια προβλήματα όπως η Angular και η vue. Ωστόσο η χρήση virtual DOM και ο τρόπος που επιτυγχάνεται το state management στην React την καθιστούν μία από τις πιο ελκυστικές επιλογές για το εγγύς μέλλον.

2.8 Draft.js

Η Draft.js είναι μία βιβλιοθήκη ανοιχτού κώδικα βασισμένη στην ReactJS η οποία αποτελεί framework για rich text editors. Ένα rich text editor είναι παρόμοιο με ένα πεδίο textbox στο οποίο έχουν προστεθεί λειτουργίες μορφοποίησης κειμένου όπως μετατροπή κειμένου σε bold, δημιουργία λίστας και άλλα. Η Draft.js παρέχει μία γκάμα εργαλείων που μετατρέπουν ένα απλό textbox σε rich text editor. Επίσης παρέχει και API για προγραμματιστές ώστε να δημιουργήσουν rich text editor σύμφωνα με τις απαιτήσεις τους. Έτσι το draft.js είναι ιδανικό για να ικανοποιηθούν οι απαιτήσεις στην δική μας εφαρμογή. Συγκεκριμένα μέσω του draft.js δίνεται η δυνατότητα στον προγραμματιστή να δημιουργήσει και να παρέμβει σε συμβάντα όπως την επικόλληση κειμένου και αρχείου και να εφαρμόσει την λογική που αρμόζει για την επίτευξη ενός στόχου.

2.8.1 Γιατί Draft.js

Είναι αλήθεια ότι υπάρχουν αρκετές λύσεις για υλοποίηση rich text editor όπως το tinymce, το jodit και άλλα, αλλά σε περιπτώσεις στις οποίες οι απαιτήσεις είναι συγκεκριμένες και παρεκκλίνουν από τις κοινότητες λειτουργίες τότε οι υπόλοιπες βιβλιοθήκες φαίνονται ανεπαρκείς. Εν αντιθέσει το draft.js καθώς αποτελεί βιβλιοθήκη ανοιχτού κώδικα και επειδή λειτουργεί ως γεννήτρια για rich text editors αποτελεί την ιδανική λύση για την περίπτωση αυτής της εφαρμογής.

2.9 MySQL

Η αποθήκευση των δεδομένων είναι μία από τις πιο σημαντικές λειτουργίες μίας εφαρμογής. Η εφαρμογή θα πρέπει να αποθηκεύει, να ενημερώνει και να προσφέρει τα δεδομένα με τρόπο γρήγορο, ασφαλή και ακέραιο. Μια λύση στο παραπάνω πρόβλημα προσφέρουν τα Σχεσιακά Συστήματα Βάσεων Δεδομένων (Relational Database Management Systems αλλιώς RDBMS). Τα βασικά στοιχεία ενός RDBMS είναι οι πίνακες δεδομένων και οι σχέσεις μεταξύ των πινάκων. Η MySQL είναι ένα RDBMS και παρακάτω θα αναλυθούν τα στοιχεία που την απαρτίζουν.

2.9.1 Πίνακες

Οι πίνακες είναι το βασικότερο στοιχείο της MySQL και απαρτίζονται από πεδία διαφόρων τύπων καθώς και από ένα πρωτεύον κλειδί και προαιρετικά από ένα ή παραπάνω ξένα κλειδιά. Τα δεδομένα ενός πίνακα αποθηκεύονται ως γραμμές στον πίνακα και ορίζονται μοναδικά βάσει της τιμής του πρωτεύοντος κλειδιού.

2.9.2 Πρωτεύον κλειδί

Το πρωτεύον κλειδί αποτελείται από ένα ή παραπάνω πεδία ενός πίνακα. Η τιμή του πρωτεύοντος κλειδιού (ή ο συνδυασμός των τιμών αν έχουμε παραπάνω πεδία) θα πρέπει να είναι μοναδική καθώς μέσω του πρωτεύοντος κλειδιού μπορούμε να ξεχωρίσουμε την κάθε γραμμή του πίνακα από τις υπόλοιπες.

2.9.3 Ξένο κλειδί

Ένα ξένο κλειδί είναι ένα πεδίο στον πίνακα του οποίου η τιμή ταυτίζεται με την τιμή ενός πεδίου άλλου πίνακα. Με το ξένο κλειδί υλοποιούνται και οι σχέσεις μεταξύ των πινάκων. Ο τύπος των σχετιζόμενων πεδίων πρέπει να είναι ίδιος και η τιμή στον δεύτερο πίνακα μοναδική.

2.9.4 Περιορισμοί (constraints)

Οι περιορισμοί χρησιμοποιούνται για να θέσουν κανόνες όσον αφορά τα δεδομένα ενός πίνακα καθώς και κανόνες που αφορούν ολόκληρο τον πίνακα[14]. Παρακάτω θα εξετάσουμε μερικούς περιορισμούς που χρησιμοποιούνται στην MySQL.

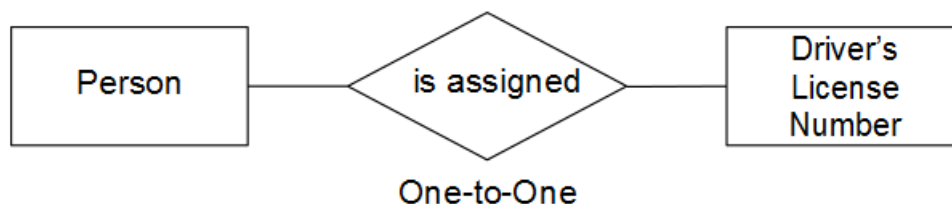
- NOT NULL – Εξασφαλίζει ότι ένα πεδίο θα έχει υποχρεωτικά τιμή
- UNIQUE – Εξασφαλίζει ότι η τιμή ενός πεδίου θα είναι μοναδική σε σχέση με την τιμή του σε άλλες γραμμές
- PRIMARY KEY – Είναι ο συνδυασμός του NOT NULL και UNIQUE
- FOREIGN KEY – Αποτρέπει την εισχώρηση / επεξεργασία τιμών που θα κατέστρεφαν την σχέση μεταξύ δύο πινάκων
- DEFAULT – Ορίζει μία τιμή για ένα πεδίο σε περίπτωση που δεν οριστεί μια τιμή κατά την εισαγωγή δεδομένων

Μέσω των περιορισμών εξασφαλίζεται η σωστή λειτουργία μίας σχεσιακής βάσης δεδομένων κρατώντας τις σχέσεις των πινάκων και αποτρέποντας την εισχώρηση λανθασμένων ή ελλιπών δεδομένων.

2.9.5 Τύποι Συσχετίσεων

Οι συσχετίσεις των οντοτήτων του μοντέλου οντοτήτων-συσχετίσεων, υλοποιούνται στο σχεσιακό μοντέλο με την χρήση ξένων κλειδιών. Οι Συσχετίσεις κατηγοριοποιούνται σε 3 βασικές κατηγορίες και παρακάτω παρουσιάζουμε πως υλοποιούνται στο σχεσιακό μοντέλο.

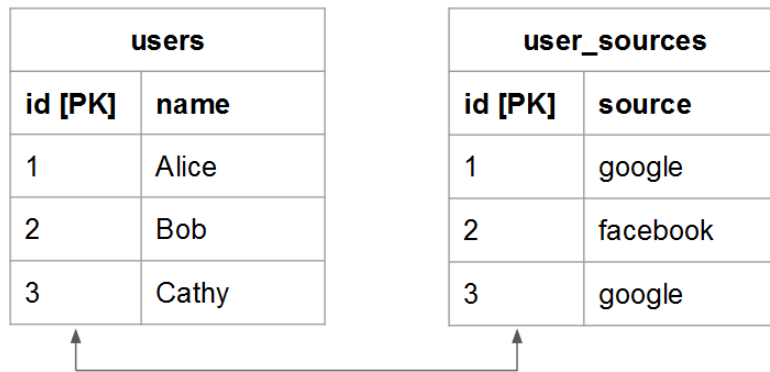
2.9.5.1 Συσχέτιση Ένα προς ένα



Σχήμα 2.3 Συσχέτιση ένα προς ένα

Στην συσχέτιση ένα προς ένα κάθε αντικείμενο της μιας κατηγορίας αντιστοιχίζεται με ακριβώς ένα αντικείμενο της άλλης. Αυτό στο σχεσιακό μοντέλο υλοποιείται με τη δημιουργία δυο πινάκων οι οποίοι

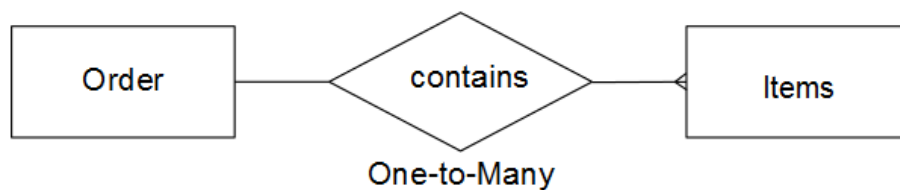
έχουν το ίδιο κύριο κλειδί. Επιπλέον το ίδιο πεδίο θα πρέπει να είναι ξένο κλειδί προς τον βασικό πίνακα.



Σχήμα 2.4 Παράδειγμα σχέσης ένα προς ένα

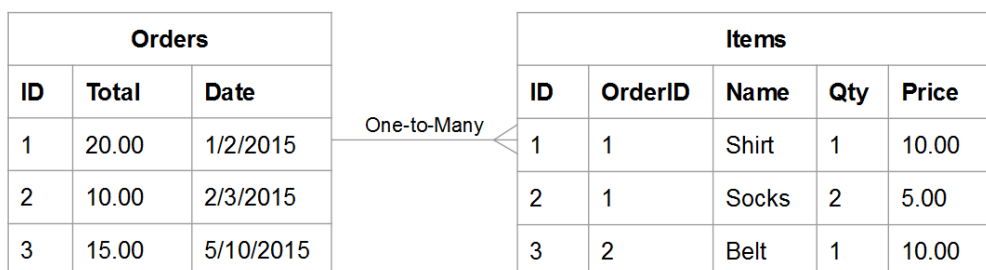
Στο παραπάνω παράδειγμα το κύριο κλειδί του πίνακα user_sources είναι και ξένο κλειδί για τον βασικό πίνακα users.

2.9.5.2 Ένα προς πολλά



Σχήμα 2.5 Σχέση ένα προς πολλά

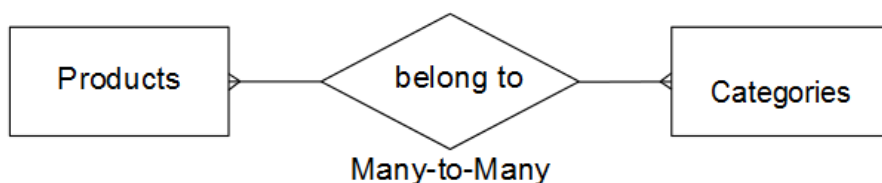
Στην σχέση ένα προς πολλά ένα αντικείμενο μίας κατηγορίας συσχετίζεται με ένα ή περισσότερα αντικείμενα μίας άλλης κατηγορίας. Αυτή η σχέση στο σχεσιακό μοντέλο υλοποιείται με την δημιουργία δύο πινάκων όπου ο δεύτερος πίνακας περιέχει μία στήλη ως ξένο κλειδί προς τον βασικό πίνακα[15] .



Σχήμα 2.6 Παράδειγμα σχέσης ένα προς πολλά

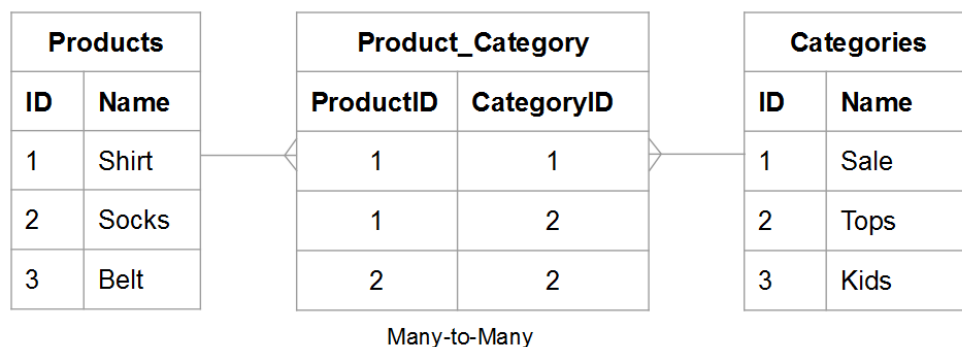
Στο παραπάνω παράδειγμα ο πίνακας Items περιέχει την στήλη OrderID η οποία είναι ξένο κλειδί προς τον βασικό πίνακα Orders. Έτσι ένα ID του πίνακα Orders μπορεί να υπάρχει σε πολλές γραμμές στο πεδίο OrderID στον πίνακα Items και επιτυγχάνεται η συσχέτιση ένα προς πολλά.

2.9.5.3 Πολλά προς πολλά



Σχήμα 2.7 Σχέση πολλά προς πολλά

Στην σχέση πολλά προς πολλά ένα αντικείμενο μίας κατηγορίας A μπορεί να συσχετίζεται με πολλά αντικείμενα της κατηγορίας B αλλά και το αντίστροφο. Δηλαδή ένα αντικείμενο της κατηγορίας B να αντιστοιχίζεται με πολλά αντικείμενα της κατηγορίας A. Στο σχεσιακό μοντέλο αυτή η συσχέτιση επιτυγχάνεται με την δημιουργία τριών πινάκων, δύο κύριων και ενός που θα εκφράζει την σχέση. Στον πίνακα που εκφράζει την σχέση περιέχεται ξένο κλειδί προς τον πίνακα A και ξένο κλειδί προς τον πίνακα B. Έτσι μέσω αυτού του πίνακα εφαρμόζεται και η συσχέτιση[15] .



Σχήμα 2.8 Παράδειγμα σχέσης πολλά προς πολλά

Στο παραπάνω παράδειγμα βλέπουμε 3 πίνακες, τον Products, τον Categories και τον Product_Category. Ο πίνακας Product_Category έχει την στήλη ProductID η οποία είναι ξένο κλειδί προς τον πίνακα Products και την στήλη CategoryID η οποία είναι ξένο κλειδί προς τον πίνακα Categories. Με αυτό τον τρόπο μία γραμμή Products αντιστοιχίζεται με πολλές γραμμές Categories και μια γραμμή Categories αντιστοιχίζεται με πολλές γραμμές Products.

2.9.6 Γιατί MySQL

Αν συγκρίναμε την MySQL με άλλες τεχνολογίες RDBMS στον χώρο δεν θα βρίσκαμε πολλές διαφορές. Ωστόσο είναι σημαντικό να τονιστεί πως η MySQL διανέμεται με την άδεια GPL (General Public License) και έχει την δυνατότητα λειτουργίας σε πληθώρα λειτουργικών συστημάτων σε αντίθεση με άλλες τεχνολογίες[16] . Αυτό σημαίνει πως η MySQL διανέμεται δωρεάν και παρέχει την ευελιξία που χρειαζόμαστε για το εκάστοτε έργο. Επίσης η PHP διαθέτει ενσωματωμένες λειτουργίες για διασύνδεση με MySQL βάσεις δεδομένων όπως τα αντικείμενα mysqli και PDO σε αντίθεση με την διασύνδεση άλλου τύπου βάσεων δεδομένων που απαιτείται η χρήση εξωτερικής βιβλιοθήκης.

2.10 Επίλογος

Σε αυτό το κεφάλαιο έγινε μία ανάλυση των τεχνολογιών που χρησιμοποιήθηκαν για την διεκπεραίωση του έργου. Συγκεκριμένα, για κάθε τεχνολογία έγινε μία ιστορική αναδρομή, αναλύθηκαν οι τρόποι

Κεφάλαιο 2

λειτουργίας και η αρχιτεκτονική καθώς και τα πλεονεκτήματα που εντοπίζονται σε σχέση με παρόμοιες τεχνολογίες.

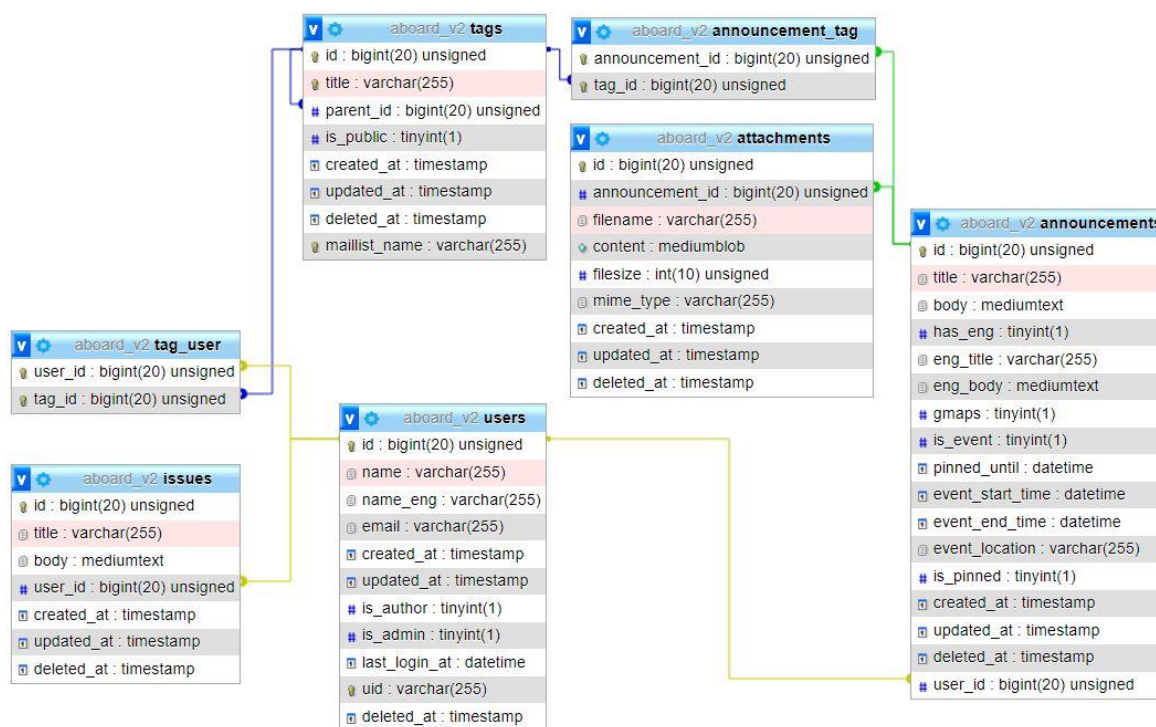
Κεφάλαιο 3ο: Επανασχεδιασμός της εφαρμογής

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλυθούν τα βήματα που ακολουθήθηκαν ώστε να υλοποιηθεί η νέα εφαρμογή έχοντας ως βάση τις απαιτήσεις. Κάποιες από τις παρακάτω αναφορές αφορούν δομικές αλλαγές και δεν έχει δοθεί μεγαλύτερη έμφαση καθώς η λειτουργία έχει παραμείνει η ίδια. Αντίθετα, άλλα κομμάτια της εφαρμογής δημιουργήθηκαν εκ νέου και αναλύονται με περισσότερη λεπτομέρεια.

3.2 Βάση δεδομένων

Η βάση δεδομένων είναι ένα από τα σημαντικότερα κομμάτια μίας εφαρμογής και για αυτό το λόγο ο σχεδιασμός του σχήματος θα πρέπει να μελετάται με προσοχή και ακολουθώντας τους κανόνες κανονικοποίησης. Παρακάτω παρουσιάζεται το απλουστευμένο το σχήμα της βάσης δεδομένων της εφαρμογής εμφανίζοντας τους κύριους πίνακες.



Εικόνα 3.1 Παράδειγμα σχήματος της βάσης δεδομένων

3.2.1 Τροποποιήσεις

Το παραπάνω σχήμα δέχθηκε τροποποιήσεις σε λίγα σημεία λόγω της καλής σχεδίασης που είχε ήδη. Συγκεκριμένα προστέθηκε στον πίνακα users το πεδίο deleted_at τύπου datetime για την υλοποίηση του soft delete των users. Επίσης δημιουργήθηκε το view tags_leafs που χρησιμεύει στην εύρεση των τελικών στοιχείων του πίνακα tags που υλοποιεί δενδροειδή δομή δεδομένων. Παρακάτω θα παρουσιαστεί το query του view προς εξήγηση.

```
CREATE VIEW `tags_leafs` AS
(
  select `t`.`id` AS `id`,`t`.`title` AS `title`,
```

```

        `t`.`parent_id` AS `parent_id`, `t`.`is_public` AS `is_public`,
        `t`.`created_at` AS `created_at`, `t`.`updated_at` AS `updated_at`,
        `t`.`deleted_at` AS `deleted_at`, `t`.`maillist_name` AS `maillist_name`,
        1 - count(distinct `c`.`parent_id`) AS `is_leaf`
    from
        (`tags` `t`
         left join `tags` `c` on(`t`.`id` = `c`.`parent_id`)
        )
    group by `t`.`id`
)

```

Κώδικας 3.1 View tags_leafs

Το παραπάνω query αντλεί τα δεδομένα από τον πίνακα tags κάνοντας left join τον εαυτό του ενώνοντας τις γραμμές του πρώτου πίνακα που το id είναι ίσο με το parent_id του δεύτερου πίνακα. Στο τέλος μέσω του group_by t.id δημιουργούνται σύνολα με βάση το id του πρώτου πίνακα και έτσι μπορούμε να μετρήσουμε αν το id υπάρχει ως parent_id σε άλλη γραμμή για την κάθε γραμμή του πίνακα και τελικά αν είναι leaf μέσω της πράξης

```
1 - count(distinct `c`.`parent_id`)
```

3.3 Επανασχεδιασμός του back-end

Για τον επανασχεδιασμό του back-end αρχικά μελετήθηκε εκτενώς η υπάρχουσα υποδομή. Στην συνέχεια έγινε διαχωρισμός των λειτουργιών στις δύο εκδόσεις ώστε να παραμείνει ενεργή η τωρινή έκδοση του back-end για τρίτες εφαρμογές που συνεχίζουν να την χρησιμοποιούν.

3.3.1 Διαχωρισμός εκδόσεων

Για τον διαχωρισμό των εκδόσεων έπρεπε να αλλάξει ριζικά η δομή της εφαρμογής. Συγκεκριμένα η ανάγκη διαχωρισμού προέκυψε για τους Controllers, τα Models, τα Middleware, τα Requests και τα Notifications. Πλέον όλα τα παραπάνω ανήκουν σε διαφορετικό namespace σύμφωνα με την έκδοση τους V1 και V2 αντίστοιχα.

3.3.2 Δημιουργία νέας σουίτας routes

Τα routes της νέας εφαρμογής δημιουργήθηκαν στο αρχείο api/v2 ώστε να μην παρέμβουμε στην υπάρχουσα δρομολόγηση. Η Laravel μέσω της κλάσης Route μας εξοπλίζει με αρκετά εργαλεία ώστε να ορίζουμε routes με τρόπο άμεσο. Στο παρακάτω παράδειγμα θα εξετάσουμε μερικές από τις επιλογές που μας δίνονται.

```

Route::prefix('tags')->group(function () {
    Route::middleware(['auth.admin'])->group(function () {
        Route::post('/', 'Tag\TagController@store');
        Route::put('/{id}', 'Tag\TagController@update');
        Route::delete('/{id}', 'Tag\TagController@destroy');
    });
    Route::get('/', 'Tag\TagController@index');
    Route::get('/{id}', 'Tag\TagController@show');
});

```

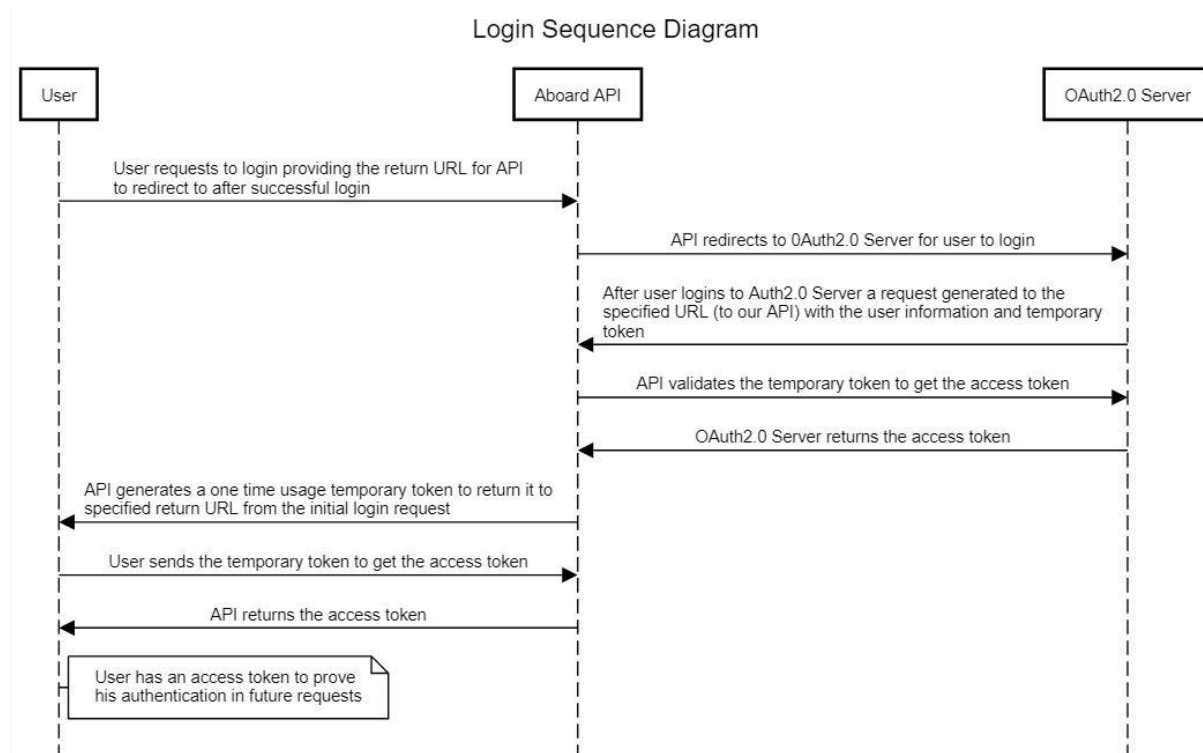
```
Route::get('/{id}/users', 'Tag\TagController@returnUsers');
});
```

Κώδικας 3.2 API Routes

Αρχικά με την static μέθοδο prefix μπορούμε να ορίσουμε το κομμάτι του URI που θα ξεκινούν τα εμφωλευμένα routes. Έπειτα, μέσω της μεθόδου group ορίζουμε τα εσωτερικά routes. Τέλος, με την μέθοδο middleware ορίζουμε τα φίλτρα πρόσβασης για μία ομάδα routes. Χωρίς τα παραπάνω εργαλεία η σύνταξη των routes θα περιείχε επαναλαμβανόμενο κώδικα, κάτι το οποίο πολλές φορές οδηγεί και σε λάθη ή αβλεψίες.

3.3.3 Ανάπτυξη νέου συστήματος αυθεντικοποίησης

Η υπάρχουσα υλοποίηση παρέχει μία υβριδική υλοποίηση της διατήρησης ενός χρήστη σε σύνδεση μέσω session και μέσω bearer token που μετατρέπεται κάθε φορά σε session για την διεκπεραίωση ενός αιτήματος. Ο παραπάνω τρόπος παρότι είναι λειτουργικός, δεν είναι αναγκαίος στην νέα υλοποίηση καθώς η νέα εφαρμογή είναι stateless. Έτσι αποφασίστηκε να χρησιμοποιηθεί JWT για την αυθεντικοποίηση και την διατήρηση σύνδεσης του χρήστη. Πλέον κάθε αίτημα πλην του login περιέχει στο Header Authorization το token του χρήστη εφόσον έχει συνδεθεί. Στο back-end γίνεται επαλήθευση του token με την χρήση του Auth guard api/v2. Επίσης έχει δημιουργηθεί ξεχωριστός Provider για την σύνδεση χρήστη μέσω του login.ieu.gr ώστε να λειτουργεί ανεξάρτητα από την παλαιά υλοποίηση. Παρακάτω θα δούμε ένα διάγραμμα ακολουθίας που εξηγεί την διαδικασία σύνδεσης ενός χρήστη.



Εικόνα 3.2 Διάγραμμα ακολουθίας διαδικασίας σύνδεσης χρήστη

Αρχικά ο χρήστης στέλνει αίτημα σύνδεσης προς το API παρέχοντας το URL επιστροφής έπειτα από την αυθεντικοποίηση. Στην συνέχεια το API ανακατευθύνει προς τον OAuth2.0 Server με τα στοιχεία Client ID, Response type και Redirect URL που είναι απαραίτητα για την διαδικασία αυθεντικοποίησης. Αφού συνδεθεί ο χρήστης με τα στοιχεία του, ο OAuth2.0 Server επιστρέφει τις πληροφορίες προς το

API μέσω του redirect URL. Με την σειρά του το API επιστρέφει στο URL που δήλωσε ο χρήστης στο αρχικό αίτημα ένα προσωρινό token μιας χρήσης. Στην συνέχεια ο χρήστης στέλνει στο API το προσωρινό token για να λάβει το κανονικό access token. Στα επόμενα αιτήματα του χρήστη προστίθεται και το access token ως validation της σύνδεσης του.

3.3.4 Παρουσίαση νέου API

Στην νέα εφαρμογή όλα τα αιτήματα εκτελούνται μόνο από το API ώστε να αποδεσμευτεί το back-end από το front-end. Ο παρακάτω πίνακας παρουσιάζει όλα τα end-points του API με την μέθοδο αιτήματος και μία σύντομη περιγραφή για το καθένα.

End-point	Method	Περιγραφή
/api/v2/announcements	GET	Επιστρέφει τις ανακοινώσεις φιλτραρισμένες και paginated.
/api/v2/announcements/{id}	GET	Επιστρέφει μία ανακοίνωση
/api/v2/announcements/ {announcement_id}/{attachment_id}	GET	Επιστρέφει το επισυναπτόμενο αρχείο μίας ανακοίνωσης
/api/v2/my_announcements/	GET	Επιστρέφει τις ανακοινώσεις του συνδεδεμένου χρήστη. Απαιτεί σύνδεση και δικαιώματα author.
/api/v2/announcements/edit_view/{id}	GET	Επιστρέφει μία ανακοίνωση συνδεδεμένου χρήστη βελτιστοποιημένη για διαδικασίες επεξεργασίας. Απαιτεί σύνδεση, δικαιώματα author και δικαίωμα επεξεργασίας της ανακοίνωσης.
/api/v2/announcements/{id}	POST	Εισάγει νέα ανακοίνωση. Απαιτεί σύνδεση και δικαιώματα author.
/api/v2/announcements/{id}	PUT	Αλλάζει τις πληροφορίες μίας ανακοίνωσης. Απαιτεί σύνδεση, δικαιώματα author και δικαίωμα επεξεργασίας της ανακοίνωσης.
/api/v2/announcements/{id}	DELETE	Διαγράφει μία ανακοίνωση. Απαιτεί σύνδεση, δικαιώματα author και δικαίωμα επεξεργασίας της ανακοίνωσης.
/api/v2/tags/	GET	Επιστρέφει τις ετικέτες
/api/v2/tags/{id}	GET	Επιστρέφει την ετικέτα
/api/v2/tags/{id}/users	GET	Επιστρέφει τους χρήστες που ακολουθούν την ετικέτα. Επιτρέπεται η πρόσβαση μόνο από συγκεκριμένη IP

/api/v2/filtertags	GET	Επιστρέφει τις ετικέτες που χρησιμοποιούνται στις εμφανιζόμενες ανακοινώσεις
/api/v2/subscribetags	GET	Επιστρέφει τις ετικέτες βελτιστοποιημένες για χρήση στην επιλογή subscribe
/api/v2/all_tags	GET	Επιστρέφει όλες τις ετικέτες
/api/v2/most_used_tags	GET	Επιστρέφει τις πιο χρησιμοποιημένες ετικέτες για γρήγορη χρήση στις φόρμες εισαγωγής / επεξεργασίας ανακοίνωσης. Απαιτεί σύνδεση και δικαιώματα author.
/api/v2/authors	GET	Επιστρέφει τους συγγραφείς των εμφανιζόμενων ανακοινώσεων.
/api/v2/all_authors	GET	Επιστρέφει όλους τους συγγραφείς. Επιτρέπεται μόνο στον admin
/api/v2/auth/login_web	GET	Αρχικό αίτημα σύνδεσης. Προαιρετική παράμετρος είναι η GET παράμετρος redirect όπου δηλώνει το URL επιστροφής έπειτα από επιτυχή σύνδεση.
/api/v2/auth/token	POST	Επιστρέφει access token έπειτα από τον έλεγχο του προσωρινού token. Το αίτημα πρέπει να περιέχει την POST μεταβλητή token.
/api/v2/auth/logout	GET	Ακυρώνει το access token
/api/v2/auth/whoami	GET	Επιστρέφει τα στοιχεία του χρήστη έπειτα από έλεγχο του access token
/api/v2/auth/subscriptions	GET	Επιστρέφει τις ετικέτες που ακολουθεί ο χρήστης
/api/v2/auth/subscribe	POST	Θέτει τις ετικέτες που ακολουθεί ο χρήστης

Πίνακας 3.1 API end-points

Για την καλύτερη κατανόηση των end-points θα δούμε παρακάτω ένα παράδειγμα για κάθε end-point.

3.3.4.1 Μέθοδοι GET

/api/v2/announcements

Το συγκεκριμένο end-point δέχεται ως παραμέτρους τα παρακάτω:

- **sortId:** default 0

- **perPage**: default 10
- **title**: default ''
- **page**: default 1
- **body**: default ''
- **tags[]**: Array από τα ID των tags
- **users[]**: Array από τα ID των users
- **updatedAfter**: Date με format Y-m-d
- **updatedBefore**: Date με format Y-m-d

Η απάντηση στο request εμφανίζεται παρακάτω:

```
{
  "data": [
    {
      "id": 61480,
      "title": "Ανακοίνωση",
      "eng_title": "Announcement",
      "body": "<p>Ανακοίνωση</p>",
      "preview": "Ανακοίνωση",
      "eng_preview": "Announcement",
      "has_eng": null,
      "created_at": "2023-05-07 20:06",
      "updated_at": "2023-05-07 20:08",
      "is_pinned": null,
      "pinned_until": null,
      "is_event": null,
      "event_start_time": null,
      "event_end_time": null,
      "event_location": null,
      "gmaps": null,
      "tags": [
        {
          "id": 3,
          "title": "2ο Εξάμηνο",
          "parent_id": 163,
          "is_public": false,
          "maillist_name": "sem2"
        }
      ],
      "attachments": [
        {
          "id": 1,
          "announcement_id": 1,
          "filename": "TEST.pdf",
          "filesize": 147719,
          "mime_type": "application/pdf",
          "attachment_url":
"http://tboard.iee.ihu.gr/announcements/61480/attachments/1?action=download",
          "attachment_url_view":
"http://tboard.iee.ihu.gr/announcements/61480/attachments/1"
        }
      ],
      "author": {
        "name": "TEST TEST",
        "id": 1
      },
      "announcement_url": "http://tboard.iee.ihu.gr/announcements/1"
    }
  ],
  "meta": {
    "current_page": "1",
    "from": 1,

```



```

    "last_page": 1,
    "path": "http://tboard.iee.ihu.gr/api/v2/announcements",
    "per_page": "10",
    "to": 1,
    "total": 1
  }
}

```

Κώδικας 3.3 Παράδειγμα response του GET /api/v2/announcements

/api/v2/announcements/{id}

Επιστρέφει την ανακοίνωση με το συγκεκριμένο ID.

Παράδειγμα απάντησης:

```

{
  "id": 61480,
  "title": "Ανακοίνωση",
  "eng_title": "Announcement",
  "body": "<p>Ανακοίνωση</p>",
  "preview": "Ανακοίνωση",
  "eng_preview": "Announcement",
  "has_eng": null,
  "created_at": "2023-05-07 20:06",
  "updated_at": "2023-05-07 20:08",
  "is_pinned": null,
  "pinned_until": null,
  "is_event": null,
  "event_start_time": null,
  "event_end_time": null,
  "event_location": null,
  "gmaps": null,
  "tags": [
    {
      "id": 3,
      "title": "2ο Εξάμηνο",
      "parent_id": 163,
      "is_public": false,
      "maillist_name": "sem2"
    }
  ],
  "attachments": [
    {
      "id": 1,
      "announcement_id": 1,
      "filename": "TEST.pdf",
      "filesize": 147719,
      "mime_type": "application/pdf",
      "attachment_url":
"http://tboard.iee.ihu.gr/announcements/61480/attachments/1?action=download",
      "attachment_url_view":
"http://tboard.iee.ihu.gr/announcements/61480/attachments/1"
    }
  ],
  "author": {
    "name": "TEST TEST",
    "id": 1
  },
  "announcement_url": "http://tboard.iee.ihu.gr/announcements/1"
}

```

Κώδικας 3.4 Παράδειγμα response του GET /api/v2/announcements/{id}

/api/v2/tags/

Επιστρέφει όλες τις εγγραφές tag.

Παράδειγμα απάντησης

```
{
  "data": [
    {
      "id": 1,
      "title": "Όλες οι ανακοινώσεις",
      "parent_id": null,
      "is_public": false,
      "maillist_name": "all"
    },
    {
      "id": 26,
      "title": "test",
      "parent_id": 1,
      "is_public": false,
      "maillist_name": "test1"
    }
  ]
}
```

Κώδικας 3.5 Παράδειγμα response του GET /api/v2/tags

/api/v2/filtertags

Επιστρέφει τα tags που χρησιμοποιούνται στις εμφανιζόμενες ανακοινώσεις. Αν για παράδειγμα ένα tag δεν χρησιμοποιείται από καμία ανακοίνωση δεν θα εμφανιστεί στο response.

Παράδειγμα απάντησης:

```
[
  {
    "id": 202,
    "title": "Ερευνητικά Εργαστήρια",
    "parent_id": 1,
    "is_public": false,
    "created_at": null,
    "updated_at": null,
    "deleted_at": null,
    "maillist_name": null,
    "announcements_count": 6,
    "children_recursive": [
      {
        "id": 18748,
        "title": "ImseLab",
        "parent_id": 202,
        "is_public": true,
        "created_at": "2021-07-19 09:11:04",
        "updated_at": "2021-07-19 09:11:04",
        "deleted_at": null,
        "maillist_name": "imselab",
        "announcements_count": 6,
        "children_recursive": []
      }
    ]
  },
  {
    "id": 2,
    "title": "Εξάμηνο Α",
    "parent_id": 1,
    "is_public": false,
    "created_at": null,

```

```

    "updated_at": null,
    "deleted_at": null,
    "maillist_name": null,
    "announcements_count": 6,
    "children_recursive": []
  }
]

```

Κώδικας 3.6 Παράδειγμα response του GET /api/v2/filtertags

Αξίζει να τονίσουμε ότι η απάντηση περιέχει εμφωλευμένα την σχέση parent – child για διευκόλυνση στην χρήση σαν φίλτρο.

/api/v2/subscribetags

Επιστρέφει τα tags βελτιστοποιημένα για την χρήση στο μενού subscribe. Η απάντηση του συγκεκριμένου end-point είναι ίδια με το /api/v2/filtertags με μόνες διαφορές την εμφάνιση του root tag «Όλες οι ανακοινώσεις» και η ονομασία του πεδίου children_recursive όπου σε αυτό το end-point είναι childrensub_recursive.

/api/v2/all_tags

Επιστρέφει όλα τα tags όπως στο /api/v2/subscribetags χωρίς το root tag. Ο διαχωρισμός έχει γίνει κυρίως για μελλοντικές επεκτάσεις που μπορεί να διαφοροποιούν το συγκεκριμένο end-point σε σχέση με τα άλλα.

/api/v2/most_used_tags

Επιστρέφει τα πιο χρησιμοποιημένα tags του χρήστη.

Στο request header θα πρέπει να υπάρχει Authorization: Bearer ACCESS_TOKEN.

Παράδειγμα απάντησης:

```

[
  {
    "id": 25,
    "title": "1101 Μαθηματικά I",
    "parent_id": 2,
    "is_public": 0,
    "created_at": "2020-05-11 21:15:15",
    "updated_at": "2020-05-11 21:15:15",
    "deleted_at": null,
    "maillist_name": "course_1101",
    "is_leaf": 1,
    "weight": "183333.3333"
  }
]

```

Κώδικας 3.7 Παράδειγμα response του GET /api/v2/most_used_tags

/api/v2/authors

Επιστρέφει τους ενεργούς συγγραφείς.

Παράδειγμα απάντησης:

```

[
  {
    "id": 1,
    "name": "ΕΠΩΝΥΜΟ1 ΟΝΟΜΑ1",
    "announcements_count": 5
  }
]

```

```
    },
    {
      "id": 2,
      "name": " ΕΠΩΝΥΜΟ2 ΟΝΟΜΑ2",
      "announcements_count": 1
    }
  ]
```

Κώδικας 3.8 Παράδειγμα response του GET /api/v2/authors

/api/v2/auth/whoami

Επιστρέφει τα στοιχεία χρήστη.

Στο request header θα πρέπει να υπάρχει Authorization: Bearer ACCESS_TOKEN.

Παράδειγμα απάντησης:

```
{
  "id": 1,
  "name": " ΕΠΩΝΥΜΟ1 ΟΝΟΜΑ1",
  "name_eng": " ΕΠΩΝΥΜΟ1 ΟΝΟΜΑ1",
  "email": "myemail@test.test",
  "created_at": "2023-04-22 10:39:42",
  "updated_at": "2023-05-10 14:30:21",
  "is_author": 1,
  "is_admin": 1,
  "last_login_at": null,
  "uid": "UID123",
  "deleted_at": null
}
```

Κώδικας 3.9 Παράδειγμα response του GET /api/v2/auth/whoami

/api/v2/auth/subscriptions

Επιστρέφει τα tags που ακολουθεί ο χρήστης.

Στο request header θα πρέπει να υπάρχει Authorization: Bearer ACCESS_TOKEN.

Παράδειγμα απάντησης:

```
[
  {
    "id": 1,
    "title": "Όλες οι ανακοινώσεις",
    "parent_id": null,
    "is_public": false,
    "created_at": "2020-05-11 21:15:15",
    "updated_at": "2020-05-11 21:15:15",
    "deleted_at": null,
    "maillist_name": "all",
    "pivot": {
      "user_id": 1,
      "tag_id": 1
    }
  }
]
```

Κώδικας 3.10 Παράδειγμα response του GET /api/v2/auth/subscriptions

3.3.4.2 Μέθοδοι POST

/api/v2/announcements

Εισάγει νέα ανακοίνωση.

Στο request header θα πρέπει να υπάρχει Authorization: Bearer ACCESS_TOKEN.

Παράδειγμα request body:

```
body: <p>τεστ</p>
eng_body: ""
title: τεστ
eng_title: test
tags[0]: 2
tags[1]: 3
```

Κώδικας 3.11 Παράδειγμα request προς το POST /api/v2/auth/subscriptions

Η απάντηση που θα λάβουμε είναι ίδια με το /api/v2/announcements/{id}.

/api/v2/auth/subscribe

Επεξεργάζεται τις ετικέτες που ακολουθεί ο χρήστης. Όταν γίνεται ένα request το σύστημα διαγράφει τις παλαιές επιλογές και εισάγει τις νέες.

Στο request header θα πρέπει να υπάρχει Authorization: Bearer ACCESS_TOKEN.

Παράδειγμα request body:

```
tags: [11, 15]
```

Κώδικας 3.12 Παράδειγμα request προς το POST /api/v2/auth/subscribe

Παράδειγμα απάντησης:

```
{
  "id": 1,
  "subscriptions": [
    {
      "id": 11,
      "title": "Πρακτική άσκηση",
      "parent_id": 1,
      "is_public": false,
      "created_at": "2020-05-11 21:15:15",
      "updated_at": "2020-05-11 21:15:15",
      "deleted_at": null,
      "maillist_name": "praktiki",
      "pivot": {
        "user_id": 1,
        "tag_id": 11
      }
    },
    {
      "id": 15,
      "title": "Νέα τμήματος",
      "parent_id": 1,
      "is_public": true,
      "created_at": "2020-05-11 21:15:15",
      "updated_at": "2020-05-11 21:15:15",
      "deleted_at": null,
      "maillist_name": "dept_news",
      "pivot": {
        "user_id": 1,
        "tag_id": 15
      }
    }
  ]
}
```

```
]
}
```

Κώδικας 3.13 Παράδειγμα response από το POST /api/v2/auth/subscribe

3.3.4.3 Μέθοδοι PUT

/api/v2/announcements/{id}

Επεξεργάζεται την ανακοίνωση με το συγκεκριμένο ID.

Στο request header θα πρέπει να υπάρχει Authorization: Bearer ACCESS_TOKEN. Επίσης ο χρήστης θα πρέπει να έχει δικαίωμα επεξεργασίας της ανακοίνωσης. Το request και το response είναι ίδια με την αντίστοιχη μέθοδο POST.

3.3.5 Απάντηση της μορφής icalendar

Για τις ανάγκες συγχρονισμού των events με τρίτες εφαρμογές υλοποιήθηκε διαδικασία response της μορφής icalendar σύμφωνα με το RFC 5545. Το συγκεκριμένο response το λαμβάνουμε από το endpoint /api/v2/announcements εφόσον εισάγουμε στο header το Content-Type: text/calendar.

Παράδειγμα response τύπου icalendar:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//it/iee//ihu v1.0
BEGIN:VEVENT
UID:61294-iee-ihu
DTSTAMP:20220915T232830Z
DTSTART:20220917T110000Z
DTEND:20220917T150000Z
NAME:test
DESCRIPTION:test
X-ALT-DESC;FMTTYPE=text/html:<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
    <HTML><BODY>
    <p>test<p>
    </BODY></HTML>
LOCATION:ΑΜΦΙΘΕΑΤΡΟ ΗΛΕΚΤΡΟΝΙΚΗΣ
END:VEVENT
END:VCALENDAR
```

Κώδικας 3.14 Παράδειγμα response της μορφής icalendar

Για την αναπαραγωγή του παραπάνω response αναπτύχθηκε αλγόριθμος που μετατρέπει το collection από object σε string με τους περιορισμούς που μας δίνει το RFC 5545.

```
protected function collection_to_ical($collection)
{
    $ret = "BEGIN:VCALENDAR\n";
    $ret .= "VERSION:2.0\n";
    $ret .= "PRODID:-//it/iee//ihu v1.0\n";

    foreach ($collection as $item) {
        $DTSTAMP = date('Ymd\THis\Z', strtotime($item->created_at));
        $DTSTART = date('Ymd\THis\Z', strtotime($item->event_start_time));
        $DTEND = date('Ymd\THis\Z', strtotime($item->event_end_time));
        $description = $this->get_ical_complied_string($item->body);
        $preview = strip_tags($item->body);
        $preview_complied = $this->get_ical_complied_string($preview);
        $event_location = $this->get_ical_complied_string($item->event_location);
```

```

        $ret .= "BEGIN:VEVENT\n";
        $ret .= "UID:{$item->id}-iee-ihu\n";
        $ret .= "DTSTAMP:{$DTSTAMP}\n";
        $ret .= "DTSTART:{$DTSTART}\n";
        $ret .= "DTEND:{$DTEND}\n";
        $ret .= "NAME:{$item->title}\n";
        $ret .= "DESCRIPTION:{$preview_complied}\n";
        $ret .= "X-ALT-DESC;FMTTYPE=text/html:<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 3.2//EN\">
        <HTML><BODY>\n{$description}\n</BODY></HTML>\n";
        $ret .= "LOCATION:{$event_location}\n";
        $ret .= "END:VEVENT\n";
    }
    $ret .= "END:VCALENDAR\n";
    return $ret;
}

```

Κώδικας 3.15 Μέθοδος μετατροπής collection σε ical string

```

private function get_ical_complied_string($ogString)
{
    $compliedArray = [];
    $str_len = strlen($ogString);
    // 75 characters is the limit of a line. After that a new line is recognized as
    // a concatenation between the two strings
    for($i = 0; $i < $str_len; $i += 75) {
        $compliedArray[] = mb_substr($ogString, $i, 75, 'utf8');
    }

    return implode("\n", $compliedArray);
}

```

Κώδικας 3.16 Μέθοδος οριοθέτησης string σε γραμμές των 75 χαρακτήρων

3.3.6 Βελτιστοποίηση στα SQL Queries

Για την βέλτιστη αναζήτηση ανακοινώσεων πρέπει στο SQL query να συνδυαστούν αρκετοί πίνακες όπως ο πίνακας users, ο tags και ο announcement_tag. Το παραπάνω query έχει αρκετή πολυπλοκότητα και ο τρόπος που θα γραφτεί θα παίζει σημαντικό ρόλο στον συνολικό φόρτο. Ωστόσο καθώς μέσω των μοντέλων της Laravel δεν είναι εμφανές το παραγόμενο query, μπορεί να χρησιμοποιηθούν συγκεκριμένες μέθοδοι που δεν θα ήταν ιδανικές για τέτοιες περιπτώσεις όπως η μέθοδος exists η οποία θα «τρέξει» το subquery τόσες φορές όσες οι γραμμές του εξωτερικού query. Για αυτό τον λόγο πραγματοποιήθηκαν αλλαγές που αφαιρούν τα παραπάνω προβλήματα, κάνοντας χρήση του JOIN για να επιτευχθεί το ίδιο αποτέλεσμα. Παρακάτω θα εξετάσουμε ένα παράδειγμα χρήσης του query builder με τα αντίστοιχα αποτελέσματα.

```

when(is_array($tags) && count($tags) > 0, function ($query) use ($tags){
    $query->whereHas('tags', function ($query) use ($tags) {
        $query->whereIn('id', $tags);
    });
});

```

Κώδικας 3.17 Παράδειγμα χρήσης query builder

Το παραπάνω παράδειγμα αποτελεί κομμάτι ενός μεγαλύτερου query builder chain για το μοντέλο Announcements και συμμετέχουν αρκετοί πίνακες. Ο σκοπός του query είναι να φιλτράρει τα αποτελέσματα όταν έχει επιλεγθεί μία ή παραπάνω ετικέτες. Το query λειτουργεί ορθά και όντως φιλτράρει τα αποτελέσματα φέρνοντας μόνο τις ανακοινώσεις που περιέχουν τις επιλεγμένες ετικέτες αλλά το πρόβλημα έγκειται στον τρόπο που χτίζει το query. Αν είχαμε για παράδειγμα για φίλτρα τις ετικέτες με ID 1 και 2 και τυπώναμε το παραγόμενο query του builder θα βλέπαμε αυτό.

```
select `announcements`.* from `announcements`
where exists (
  select * from `tags` inner join `announcement_tag` on
    `tags`.`id` = `announcement_tag`.`tag_id`
    where `announcements`.`id` = `announcement_tag`.`announcement_id`
    and `id` in (1) and `tags`.`deleted_at` is null
) and `announcements`.`deleted_at` is null
```

Κώδικας 3.18 Παραγόμενο query του query builder

Όποτε αν και λειτουργεί σωστά σε μικρή κλίμακα, σε μεγάλο όγκο δεδομένων το παραπάνω query builder θα προκαλούσε καθυστερήσεις. Για να αποφευχθεί η χρήση της εντολής whereHas έπρεπε να γίνουν δύο ενώσεις πινάκων και συγκεκριμένα οι πίνακες tags και announcement_tag όπως παρακάτω.

```
$query->join('announcement_tag as ann_tag', function ($join) {
  $join->on('announcements.id', '=', 'ann_tag.announcement_id');
})
->join('tags', function ($join) {
  $join->on('ann_tag.tag_id', '=', 'tags.id');
})
->when(is_array($tags) && count($tags) > 0, function ($query) use ($tags){
  $query->whereIn('tags.id', $tags);
})
->groupBy('announcements.id');
```

Κώδικας 3.19 Χρήση join στο query ανακοινώσεων

Έτσι μπορούμε να πάρουμε την πληροφορία των tags που περιέχονται στις ανακοινώσεις και να φιλτράρουμε αντίστοιχα. Λόγω του join όμως μία γραμμή ανακοίνωσης θα μπορεί να εμφανιστεί πολλές φορές καθώς έχουμε σχέση πολλά προς πολλά με τις ετικέτες. Άρα θα πρέπει να γίνει και χρήση της εντολής DISTINCT ή GROUP BY announcements.id. Με την χρήση του παραπάνω query builder και αν είχαμε για παράδειγμα ως φίλτρο τις ετικέτες με ID 1 και 2 τότε το παραγόμενο query θα ήταν το παρακάτω.

```
select `announcements`.* from `announcements`
inner join `announcement_tag` as `ann_tag`
  on `announcements`.`id` = `ann_tag`.`announcement_id`
inner join `tags`
  on `ann_tag`.`tag_id` = `tags`.`id`
where `tags`.`id` in (1, 2)
  and `tags`.`is_public` = 1
  and `announcements`.`deleted_at` is null
group by `announcements`.`id`
```

Κώδικας 3.20 Παραγόμενο query από το διορθωμένο query builder

Αξίζει να σημειωθεί πως αντί για group by θα μπορούσαμε να χρησιμοποιήσουμε το distinct στο selection των επιστρεφόμενων γραμμών.

3.4 Επανασχεδιασμός του front-end

Για τον επανασχεδιασμό του front-end, όπως και στο back-end, μελετήθηκε αρχικά εκτενώς η υπάρχουσα υλοποίηση ώστε να γίνουν κατανοητές οι λειτουργίες που υπάρχουν ήδη. Ωστόσο, μιας και στην τωρινή εφαρμογή το back-end είναι ενσωματωμένο με το front-end αποφασίστηκε να γίνει ολική εκ νέου υλοποίηση σε ReactJS. Στα επόμενα κεφάλαια θα αναλυθούν τα επιμέρους βήματα για την ανάπτυξη της εφαρμογής, τα προβλήματα που υπήρξαν και οι τρόποι που αντιμετωπίστηκαν.

3.4.1 Αρχικό setup εφαρμογής

Για το αρχικό setup χρησιμοποιήθηκε το npm και συγκεκριμένα η εντολή `npx create-react-app {όνομα εφαρμογής}` με την οποία λαμβάνεται ένα αρχικό, άδαιο react project με όλα τα απαραίτητα αρχεία ώστε να ξεκινήσει η ανάπτυξη της εφαρμογής.

3.4.2 Routing

Η διαδικασία routing της εφαρμογής υλοποιείται στο κύριο αρχείο `App.js` με την χρήση της βιβλιοθήκης `react-router-dom`. Τα routes χωρίζονται σε δύο βασικές κατηγορίες. Τα public routes και τα private routes. Τα public routes εφαρμόζονται χωρίς κάποια προ απαιτούμενη συνθήκη σε αντίθεση με τα private routes που ελέγχουν συνθήκες σύνδεσης και δικαιωμάτων χρήστη.

Route	Τύπος	Page
/	Public	Announcements
/login_success	Public	LoginSuccess
/about	Public	About
/announcements	Public	Announcements
/announcement/:announcementId	Public	FullAnnouncement
/my_announcements	Private	MyAnnouncements
/add_announcement	Private	AnnouncementForm
/edit_announcement/:announcementId	Private	AnnouncementForm
/account	Private	User

Πίνακας 3.2 React Routes

3.4.3 Pages

Η κάθε προβολή χωρίζεται σε page. Ένα page είναι ένα component που υλοποιεί τον σκελετό μίας προβολής και τα λειτουργικά κομμάτια του και συνήθως απαρτίζεται από άλλα, μικρότερα components. Στην υλοποίηση της εφαρμογής το κάθε page είναι υπεύθυνο για την αρχικοποίηση και μεταβολή των δεδομένων που παρουσιάζονται στην προβολή. Τα child components που μπορεί να υπάρχουν στο page λαμβάνουν τα δεδομένα από το page μέσω props. Στην εφαρμογή έχουν υλοποιηθεί τα παρακάτω pages:

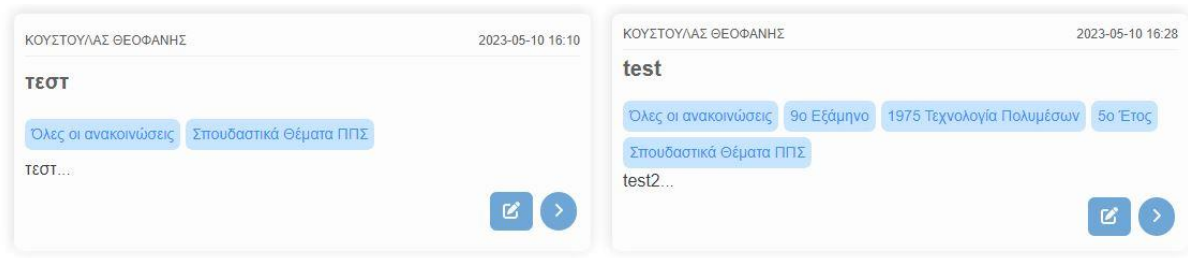
Page	Περιγραφή
Announcements	Περιέχει το listing των αναζητήσεων
LoginSuccess	Περιέχει την διαδικασία validation του temporary token μετά την επιτυχή αυθεντικοποίηση
About	Περιέχει πληροφορίες σχετικά με τους δημιουργούς της εφαρμογής
MyAnnouncements	Περιέχει τις ανακοινώσεις του συνδεδεμένου χρήστη
FullAnnouncement	Περιέχει την πλήρη προβολή μίας ανακοίνωσης

AnnouncementForm	Περιέχει την φόρμα εισαγωγής / επεξεργασίας ανακοίνωσης
User	Περιέχει τις επιλογές χρήστη

Πίνακας 3.3 Λίστα pages

3.4.4 Components

Κάθε page περιέχει ένα ή περισσότερα components. Τα components περιέχουν κομμάτια μίας προβολής και συνήθως χρησιμοποιούνται από παραπάνω από ένα page. Για τις ανάγκες της εφαρμογής τα components χωρίστηκαν σε layout components και single components. Τα layout components χρησιμοποιούνται στο σκελετό και είναι τα Header, Section και Footer τα οποία εμπεριέχονται σε κάθε κομμάτι της εφαρμογής. Τα single components περιέχουν μικρές προβολές που παρουσιάζουν μία οντότητα όπως την μικρογραφία ανακοίνωσης, τα φίλτρα και το pagination. Όπως αναφέραμε σε προηγούμενο κεφάλαιο, ένα component περιέχει το state και τα props. Με την χρήση props μπορούμε να χρησιμοποιήσουμε το ίδιο component σαν δομικό πρότυπο και να θέσουμε τις πληροφορίες από το Parent component.

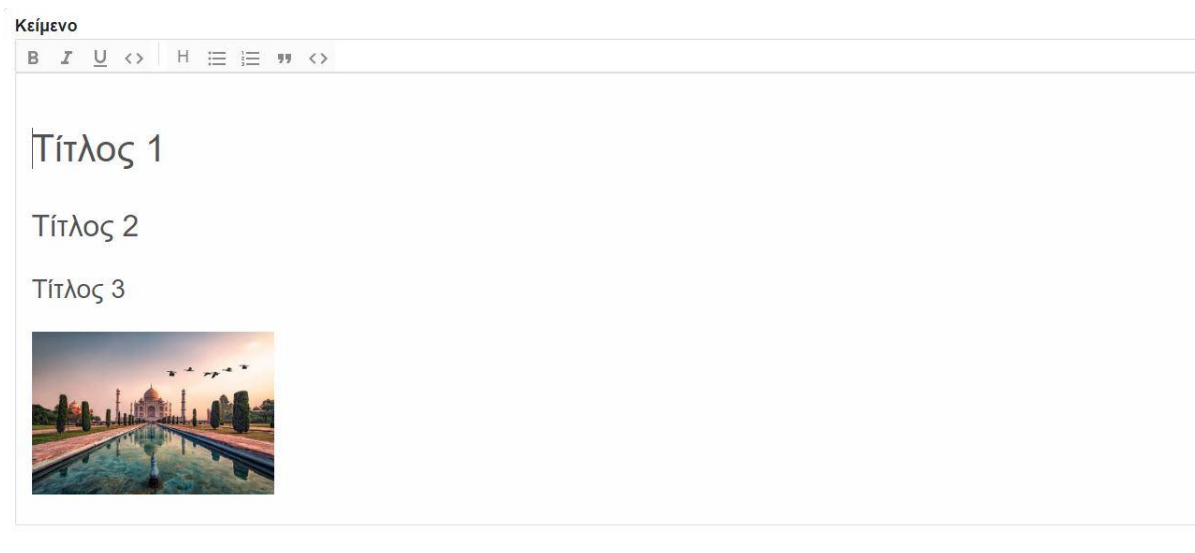


Εικόνα 3.3 Παράδειγμα χρήσης component

Στο παραπάνω παράδειγμα χρησιμοποιούμε το component Announcement το οποίο όμως περιέχει διαφορετικά δεδομένα για τίτλο, ετικέτες και κείμενο. Σε πολλά σημεία της εφαρμογής χρησιμοποιούνται components για την βέλτιστη επαναχρησιμοποίηση κώδικα. Τα περισσότερα components αντιπροσωπεύουν μικρές ενότητες HTML αλλά κάποια χρησιμοποιούνται με πολλές διαφοροποιήσεις και λειτουργίες ανάλογα με τις απαιτήσεις της εκάστοτε προβολής. Τέτοια components είναι τα CustomEditor και CheckTree.

3.4.4.1 CustomEditor

Το component CustomEditor υλοποιεί τον βασικό editor του κειμένου των ανακοινώσεων. Συγκεκριμένα μέσω της χρήσης του Draft.js παρέχει δυνατότητες μορφοποίησης κειμένου όπως και εισαγωγή και επεξεργασία φωτογραφιών. Το μενού επιλογών υλοποιήθηκε με την χρήση των components της βιβλιοθήκης draft-js-plugins και η εισαγωγή φωτογραφίας με επικόλληση ή επικόλληση link φωτογραφίας υλοποιήθηκε χρησιμοποιώντας τις ιδιότητες του editor handlePastedText και handlePastedFile αντίστοιχα. Μέσω των ιδιοτήτων αυτών δίνεται η δυνατότητα χειρισμού της επικόλλησης και έτσι μετατρέπεται το εκάστοτε κείμενο σε εικόνα όταν πρέπει.



Εικόνα 3.4 CustomEditor

3.4.4.2 CheckTree

Το component CheckTree είναι βιβλιοθήκη που ανέπτυξα προσωπικά διότι δεν υπήρχε κάποια βιβλιοθήκη που να καλύπτει τις απαιτήσεις του έργου. Συγκεκριμένα για την εφαρμογή θα έπρεπε να υπάρχει ένα component διαχείρισης δενδροειδούς δομής δεδομένων πολλαπλής επιλογής στοιχείων όπου κατά περιπτώσεις να επιλέγονται όλα τα παιδιά ενός γονικού στοιχείου όταν επιλέγεται το ίδιο και κατά περιπτώσεις να ομαδοποιούνται τα στοιχεία παιδιά όταν επιλέγονται όλα για να επιλεγθεί μόνο το γονικό στοιχείο. Το component CheckTree χρησιμοποιείται στα φίλτρα αναζήτησης ανακοινώσεων, στην δημιουργία / επεξεργασία ανακοινώσεων και στην επιλογή ετικετών που ακολουθεί ένας χρήστης.

```
<CheckTree
  id String
  options Object
  placeholder String
  value Object
  parent_value Object
  onChangeParent Function
  onChange Function
  checkAllByParent Boolean
  showOnlyParent Boolean
/>
```

Κώδικας 3.21 CheckTree component

Στο παραπάνω κομμάτι κώδικα εμφανίζεται η σύνταξη ενός CheckTree component με τις ιδιότητες του και τον τύπο τους. Η ιδιότητα options περιέχει τα δεδομένα του δέντρου και μέσω των ιδιοτήτων value, parent_value, onChange και onChangeParent μεταβάλλονται οι επιλεγμένες τιμές. Τέλος μέσω του checkAllByParent θέτουμε το component να επιλέγει όλα τα παιδιά ενός γονικού στοιχείου όταν επιλέγεται το ίδιο και με την ιδιότητα showOnlyParent θέτουμε την λειτουργία ομαδοποίησης των παιδιών ώστε όταν όλα είναι επιλεγμένα να επιλέγεται μόνο το γονικό στοιχείο.

3.4.5 Helpers

Για τις ανάγκες της εφαρμογής δημιουργήθηκαν helpers όπου παρέχουν λειτουργίες που χρησιμοποιούνται σε πολλά σημεία της εφαρμογής ώστε να υποστηριχτεί η επεκτασιμότητα και η γενίκευση.

3.4.5.1 Axios και κλάση Request

Για τα request της εφαρμογής προς τον server χρησιμοποιήθηκε η βιβλιοθήκη axios μέσω της οποίας παράγονται XMLHttpRequests και διαχειρίζονται ασύγχρονα με την χρήση promise. Για την διαχείριση των requests δημιουργήθηκε η κλάση Request η οποία περιέχει όλους τους τύπους αιτημάτων (POST, GET κ.λπ.). Μέσω της κλάσης Request δίνεται η δυνατότητα διαχείρισης του header και των εκτελούμενων requests με αφαιρετικό τρόπο.

Όνομα	Τύπος	Περιγραφή
axiosInstance	AxiosInstance	Αντικείμενο της βιβλιοθήκης axios
request_instances	Array	Instance set από τα request που εκτελούνται (τα χρειαζόμαστε για να κάνουμε cancel αν χρειαστεί)
bearer	String:nullable	Περιέχει το access token σε περίπτωση που υπάρχει
createHttpRequest(method, url, data = null, single = true, multipart = false)	Function	Δημιουργεί το request εισάγοντας επιπρόσθετες πληροφορίες (όπως το bearer) πριν την εκτέλεση
get(url, single = true)	Function	Δημιουργεί request τύπου GET. Η παράμετρος single ορίζει αν ακυρωθεί το request σε περίπτωση νέου request στο ίδιο end-point
post(url, data, single = false, multipart = false)	Function	Δημιουργεί request τύπου POST. Η παράμετρος multipart ορίζει αν στο header πρέπει να προστεθεί Content-Type: multipart/form-data
patch(url, data)	Function	Δημιουργεί request τύπου PATCH
put(url, data, multipart = false)	Function	Δημιουργεί request τύπου PUT
delete(url, data)	Function	Δημιουργεί request τύπου DELETE
handleError(error, response)	Function	Σε περίπτωση error σε request ελέγχεται αν πρόκειται για 401 και συγκεκριμένα αν έχει λήξει το token ώστε να αφαιρεθεί

Πίνακας 3.4 Κλάση request

3.4.5.2 Η κλάση User

Για να παρέχεται μία γενικευμένη λύση των ιδιοτήτων του χρήστη δημιουργήθηκε η κλάση User. Μέσω του αντικειμένου User παρέχουμε σε όποιο κομμάτι της εφαρμογής χρειάζεται την κατάσταση χρήστη όπως το αν έχει συνδεθεί και τα δικαιώματα που έχει.

Όνομα	Τύπος	Περιγραφή
-------	-------	-----------

user	Object	Αντικείμενο με τα στοιχεία του χρήστη. Τα στοιχεία μεταβάλλονται όταν μεταβάλλεται το storage
getPrivileges	Async Function	Επιστρέφει ασύγχρονα τα δικαιώματα του χρήστη, έπειτα από αίτημα προς τον server

Πίνακας 3.5 Κλάση User

3.4.5.3 Το αντικείμενο storage

Για να παρέχουμε μία δυναμική λύση σε αλλαγές που επηρεάζουν τα περισσότερα σημεία της εφαρμογής δημιουργήθηκε το αντικείμενο storage. Κάθε πληροφορία που θέλουμε η μεταβολή της να διαδίδεται καθολικά αποθηκεύεται σε αυτό το αντικείμενο. Έπειτα παρέχεται μία function onChange η οποία θέτει callback functions σε περίπτωση που ανιχνευτεί αλλαγή.

```
const storage = {
  onChangeFns: [],
  onChange(fn) {
    this.onChangeFns.push(fn);
  },
  onceChange(fn) {
    fn.once = true;
    this.onChangeFns.push(fn);
  },
  get(key, defaultValue = null) {
    if (typeof this[key] !== 'undefined') {
      return this[key];
    }
    return defaultValue;
  },
  set(key, value) {
    this[key] = value;
    this.onChangeFns.forEach((fn) => {
      fn(key, value);
      if (fn.once) {
        this.onChangeFns.splice(this.indexOf(fn), 1);
      }
    });
  },
};
```

Κώδικας 3.22 Το αντικείμενο storage

Έχοντας το παραπάνω ως αντικείμενο μπορούμε σε άλλο σημείο της εφαρμογής να θέσουμε μία callback function η οποία θα εκτελείται σε κάθε αλλαγή.

3.4.5.4 Η κλάση TagsHelper

Οι ετικέτες είναι μία πληροφορία που χρησιμοποιείται αρκετά στην εφαρμογή όπως στην αναζήτηση ανακοινώσεων, την εισαγωγή / επεξεργασία ανακοίνωσης και στο μενού που ο χρήστης επιλέγει ποιες ετικέτες θα ακολουθεί. Ωστόσο πριν μπορέσουμε να χρησιμοποιήσουμε τις πληροφορίες ετικετών όπως παρέχονται από τον server πρέπει να τις μεταβάλλουμε ώστε να χρησιμοποιηθούν από τις εκάστοτε βιβλιοθήκες. Σε αυτό το πρόβλημα δίνει την λύση η κλάση TagsHelper παρέχοντας αναδρομικές μεθόδους εύρεσης και μεταβολής ετικετών. Παρακάτω θα αναλύσουμε τις δύο μεθόδους της κλάσης TagsHelper.

```

transformTags = (tags, children_index = 'children_recursive') => {
  return tags.map(t => {
    const label = t.announcements_count ? `[${t.announcements_count}]` : t.title
    if (t[children_index] && t[children_index].length > 0) {
      return {
        label: label,
        value: t.id,
        parent: t.parent_id ? t.parent_id : null,
        children: this.transformTags(t[children_index], children_index)
      }
    } else {
      return {
        label: label,
        value: t.id,
        parent: t.parent_id ? t.parent_id : null
      }
    }
  })
}

```

Κώδικας 3.23 Μέθοδος transformTags

Η μέθοδος transformTags είναι μία αναδρομική μέθοδος που χρησιμοποιείται σε πολλά σημεία της εφαρμογής. Συγκεκριμένα η μέθοδος μετατρέπει όλα τα στοιχεία του δέντρου σε μορφή του τύπου

```

{
  label: 'Ετικέτα 1',
  value: 2, // το id της ετικέτας
  parent: 1,
  children: []
}

```

Κώδικας 3.24 Δεδομένα τύπου tag

Η δεύτερη μέθοδος της κλάσης είναι η findOption η οποία αναζητεί ένα tag σε όλα τα επίπεδα και είναι η παρακάτω.

```

findOption(value, options) {
  const found = options.filter(o => o.value === value)
  if (found.length > 0) {
    return found[0]
  }
  let recursive_search_result = false;
  options.forEach(o => {
    if (o.children && o.children.length > 0) {
      let search = this.findOption(value, o.children)
      if (search) {
        recursive_search_result = search
        return
      }
    }
  })
  return recursive_search_result
}

```

Κώδικας 3.25 Η μέθοδος findOption

Όπως βλέπουμε η μέθοδος ελέγχει για ένα επίπεδο προελαύνοντας τα στοιχεία επαναληπτικά και για τα στοιχεία που έχουν παιδιά καλεί τον εαυτό της με αρχικά options τα δεδομένα των παιδιών.

3.4.5.5 Η κλάση UriHelper

Για να γενικευτεί η διαδικασία μεταβολής του URI στην αναζήτηση ανακοινώσεων δημιουργήθηκε η κλάση UriHelper. Η κλάση αυτή είναι υπεύθυνη για την αρχικοποίηση και τις μεταβολές του URI καθώς και για την μετατροπή από πληροφορίες του URI σε query string και το αντίστροφο. Με αυτό επιτυγχάνεται η διάδοση της πληροφορίας των φίλτρων σε όλα τα εξαρτώμενα components όπως και η διαχείριση του push και pop state του history.

Όνομα	Τύπος	Περιγραφή
raw_uri	String	Το αρχικό URL που λαμβάνουμε από το window.location.href
pieces	Array	Το URL χωρισμένο στα σημεία του path
mapper	Object	Αντικείμενο με τα στοιχεία του URI του τύπου variable_name: variable_value
get(key, default_value = null)	Function	Επιστρέφει την τιμή του mapper[key]. Αν δεν υπάρχει τιμή επιστρέφει το default_value
set(key, value, silent = false)	Function	Θέτει τιμή στο mapper[key]. Η τιμή του silent ορίζει αν η μεταβολή θα αλλάξει και το URI
uri_to_string	Function	Επιστρέφει τις τιμές του mapper σε string της μορφής /όνομα/τιμή/όνομα2/τιμή2
uri_to_query_string	Function	Επιστρέφει τις τιμές του mapper σαν query string

Πίνακας 3.6 Κλάση UriHelper

3.4.5.6 Υπόλοιπα helpers

Εκτός από τα παραπάνω helpers που παρουσιάστηκαν δημιουργήθηκαν και χρησιμοποιούνται και άλλα helpers μικρής σημασίας και θα παρουσιαστούν παρακάτω ονομαστικά με μία μικρή περιγραφή.

Όνομα helper	Περιγραφή
JsonToFormDataHelper	Μετατρέπει ένα object σε FormData object. Χρησιμοποιείται στις διαδικασίες εισαγωγής / επεξεργασίας ανακοινώσεων
history	Χρησιμοποιείται στον Router της εφαρμογής
cookieHelper	Χρησιμοποιείται για set και get cookies

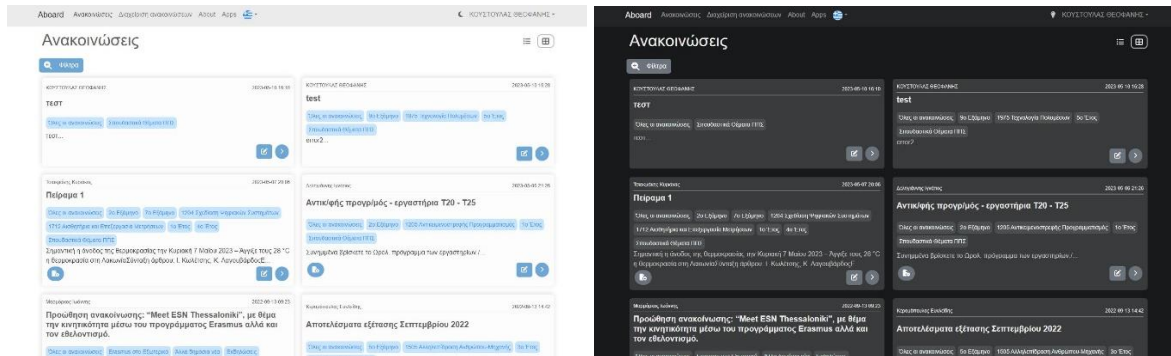
Πίνακας 3.7 Υπόλοιποι helpers

3.4.6 SASS

Για το οπτική υλοποίηση της εφαρμογής χρησιμοποιήθηκε η SASS η οποία επεκτείνει την κλασική CSS παρέχοντας δυνατότητες ομαδοποίησης, χρήσης μεταβλητών και επεκτασιμότητας. Ο σκελετός των κανόνων και των ιδιοτήτων CSS της εφαρμογής βρίσκεται σε ένα SCSS αρχείο το οποίο παράγει ένα απλό αρχείο css αρχείο που μπορούμε να χρησιμοποιήσουμε.

3.4.7 Themes

Είναι αλήθεια ότι μεγάλο ποσοστό των χρηστών προτιμούν μια ιστοσελίδα να βασίζεται σε σκούρα χρώματα χωρίς έντονη φωτεινότητα. Ωστόσο δεν μπορούμε να παραβλέψουμε το ποσοστό των χρηστών που θα θέλανε μία φωτεινή προβολή. Έτσι η εφαρμογή αναπτύχθηκε με τρόπο που να ικανοποιεί και τις δύο αυτές ομάδες παρέχοντας δύο θέματα, ένα σκοτεινό και ένα φωτεινό.



Εικόνα 3.5 Themes εφαρμογής

3.4.8 Localization

Ένα σημαντικό κομμάτι μίας εφαρμογής είναι η προσβασιμότητα. Με τον όρο προσβασιμότητα εννοείται ο τρόπος περιήγησης στην εφαρμογή αλλά και η γλώσσα εμφάνισης της πληροφορίας. Η εφαρμογή θα πρέπει να προβάλλει τα λεκτικά, εκτός από την μητρική γλώσσα της χώρας του τμήματος και σε μία ακόμη διεθνή γλώσσα όπως τα αγγλικά. Αυτό επιτυγχάνεται στην εφαρμογή με την υλοποίηση ενός internationalization αλγορίθμου και συγκεκριμένα του αντικείμενου `i18n`.

Όνομα	Τύπος	Περιγραφή
<code>language</code>	String	Η επιλεγμένη γλώσσα. Default 'el'
<code>translations</code>	Object	Οι μεταφράσεις για ελληνικά και αγγλικά. Ως κλειδί του μεταφρασμένου κειμένου χρησιμοποιείται το αμετάφραστο ελληνικό κείμενο για ευκολία στην χρήση.
<code>setLanguage(lang)</code>	Function	Μέθοδος set της μεταβλητής language
<code>onLanguageChange(fn)</code>	Function	Παίρνει σαν όρισμα μία μέθοδο που εκτελείται στην εναλλαγή γλώσσας
<code>t(key)</code>	Function	Επιστρέφει το μεταγλωττισμένο κείμενο με βάση το key και την επιλεγμένη γλώσσα
<code>get_locale_data(data, key)</code>	Function	Επιστρέφει το μεταγλωττισμένο αν υπάρχει μέσα στο αντικείμενο data με βάση το κλειδί

Πίνακας 3.8 Κλάση `I18n`

Μέσω του αντικείμενου `i18n` μπορούμε να χρησιμοποιήσουμε την μέθοδο `t` στα σημεία που θα είχαμε στατικό κείμενο ώστε να επιστρέφεται το κείμενο στην σωστή γλώσσα όπως στον παρακάτω κώδικα.

```
<h1>{i18n.t('Ανακοινώσεις')}</h1>
```

Κώδικας 3.26 Παράδειγμα χρήσης της μεθόδου μετάφρασης `t`

3.5 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκε ο τρόπος που πραγματοποιήθηκε ο επανασχεδιασμός της εφαρμογής με τα επιμέρους κομμάτια του. Η ανάλυση χωρίστηκε σε 3 ενότητες, την βάση δεδομένων, το back-end και το front-end. Για το κομμάτι της βάσης δεδομένων παρουσιάστηκε το σχήμα που χρησιμοποιεί η εφαρμογή. Για το κομμάτι του back-end αναλύθηκε ο τρόπος που διαχωρίστηκε η δομή, το νέο API και τα νέα features. Τέλος, παρουσιάστηκε η υλοποίηση του front-end με τις οντότητες που περιέχει.

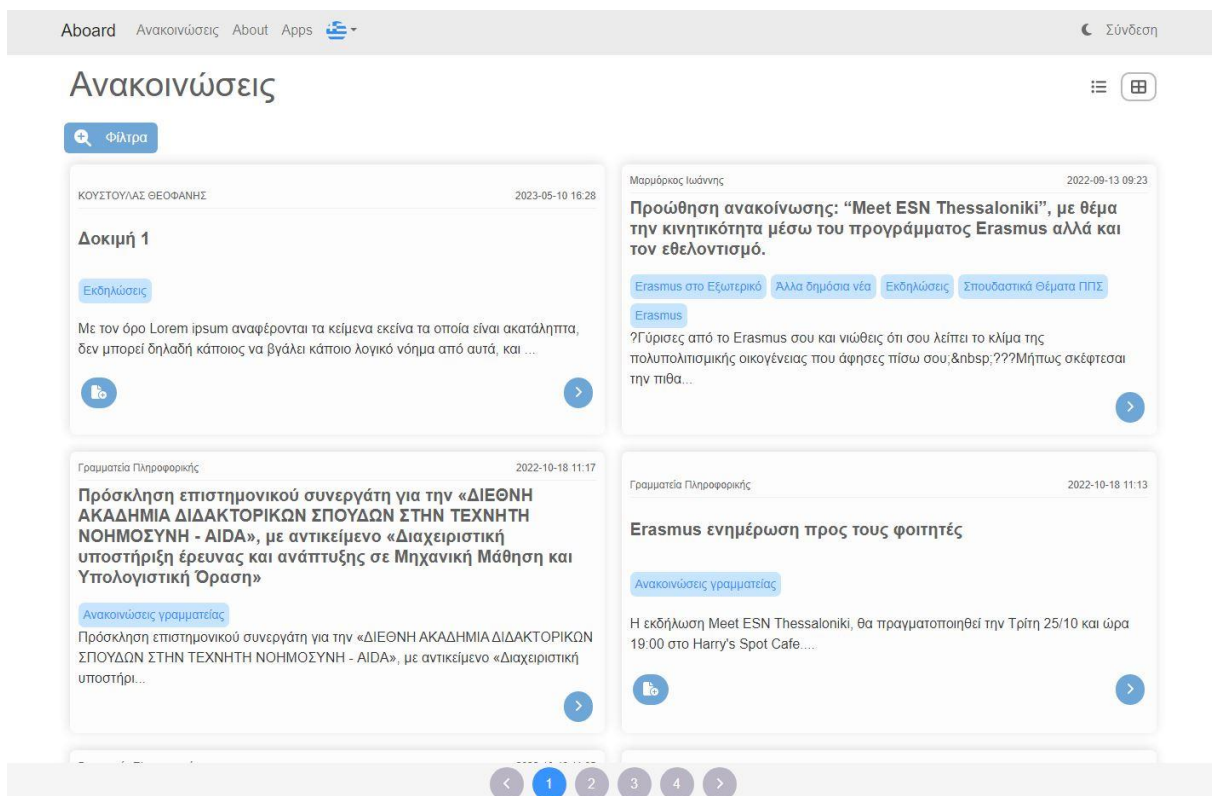
Κεφάλαιο 4ο: Παρουσίαση της εφαρμογής

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστεί η εφαρμογή από την σκοπιά του ασύνδετου χρήστη, του συνδεδεμένου χρήστη, του author και του admin.

4.2 Προβολή ανακοινώσεων

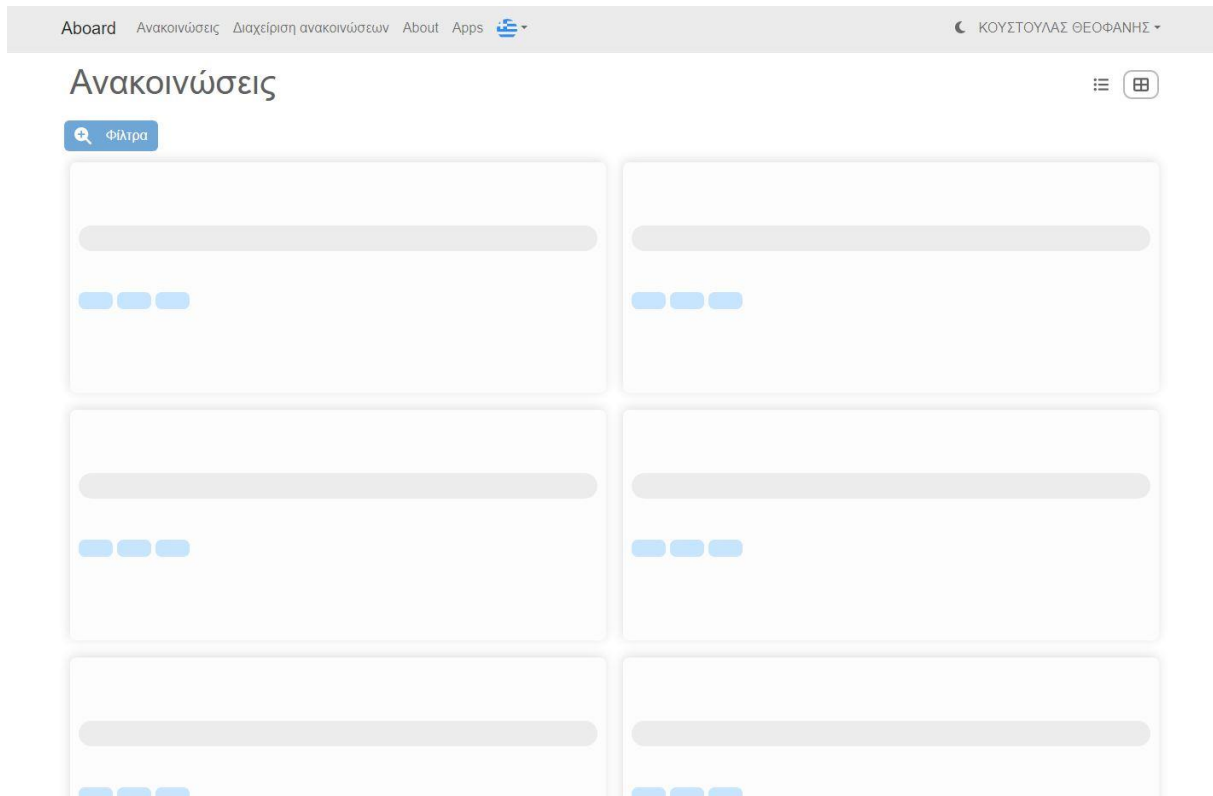
Στην προβολή ανακοινώσεων εμφανίζεται η λίστα των ανακοινώσεων με την προεπιλεγμένη ταξινόμηση κατά ημερομηνία σε φθίνουσα σειρά και με προτεραιότητα στις ανακοινώσεις που είναι σημαντικές. Επίσης στο κάτω μέρος της προβολής υπάρχει η σελιδοποίηση. Το προεπιλεγμένο πλήθος ανακοινώσεων ανά σελίδα είναι δέκα ανακοινώσεις.



Εικόνα 4.1 Προβολή ανακοινώσεων

4.2.1 Σκελετός ανακοινώσεων

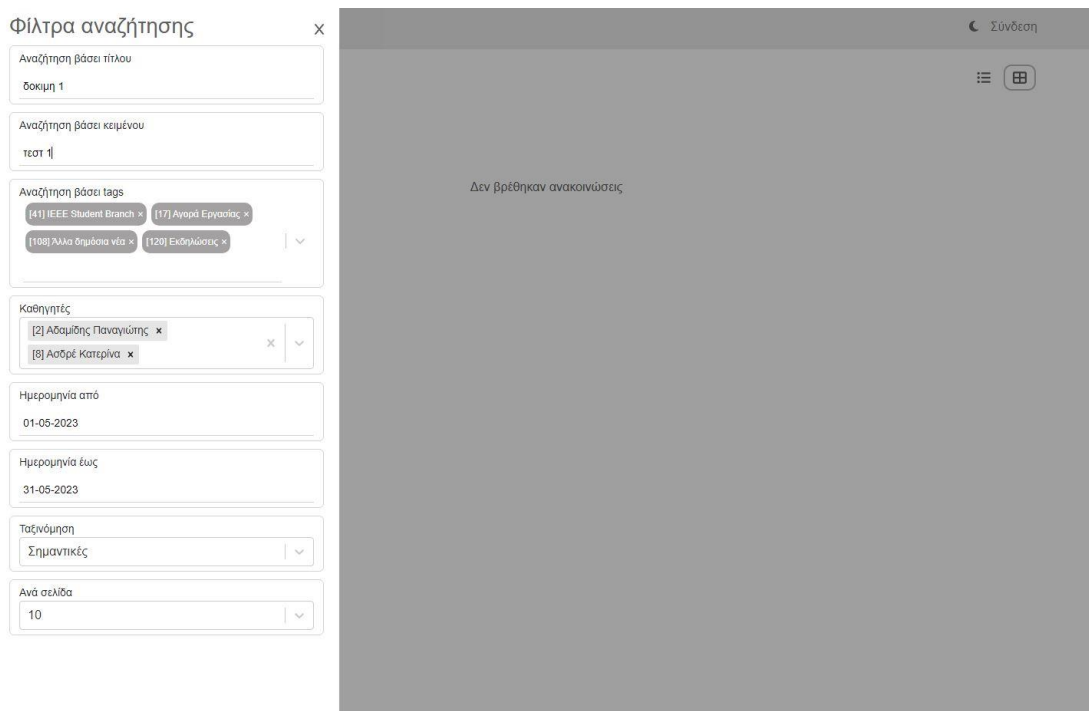
Για την ομαλή περιήγηση των χρηστών δημιουργήθηκε η λειτουργία των σκελετών ανακοινώσεων που εμφανίζονται πριν από την φόρτωση των κανονικών ανακοινώσεων. Αυτή η τεχνική έχει υιοθετηθεί και από πολλές άλλες ιστοσελίδες όπως το youtube και το facebook.



Πίνακας 4.1 Σκελετός ανακοινώσεων

4.2.2 Φιλτράρισμα ανακοινώσεων

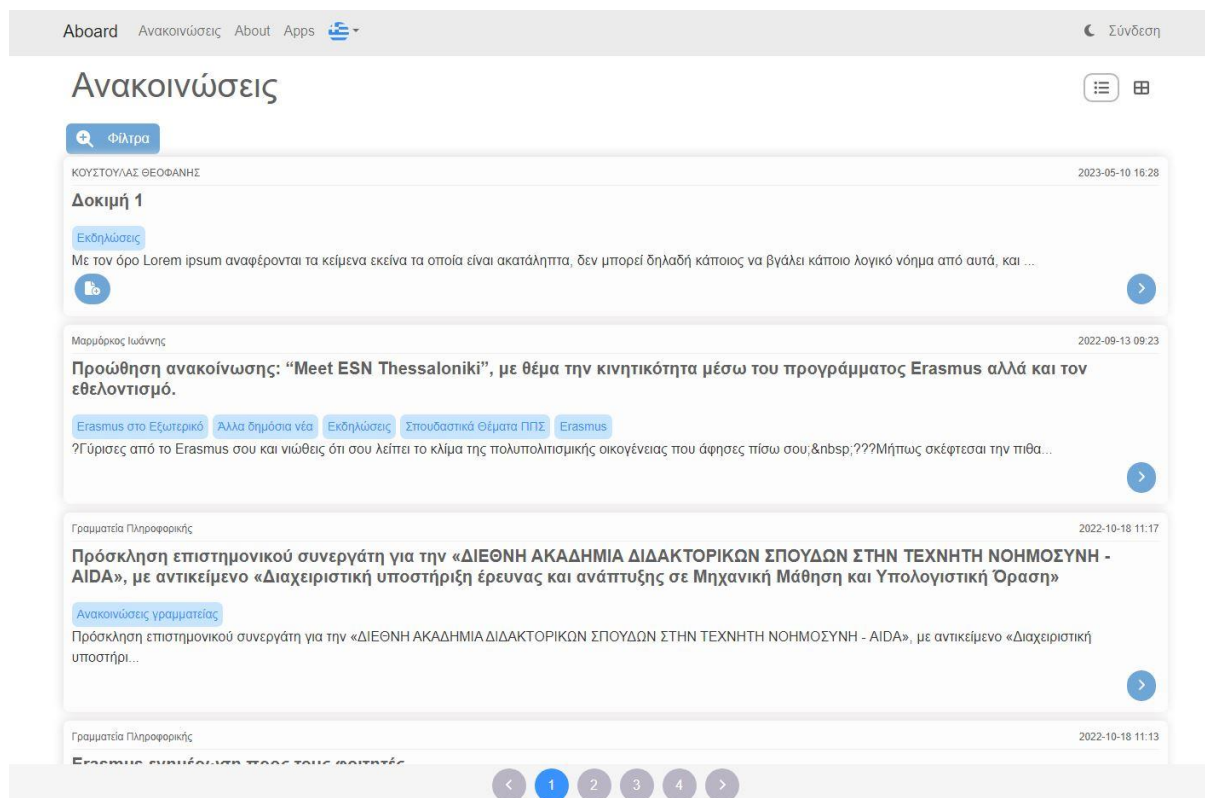
Πατώντας το κουμπί Φίλτρα στα αριστερά εμφανίζεται ένα μενού που περιέχει επιλογές φιλτραρίσματος αποτελεσμάτων με χρήση τίτλου, κειμένου, ετικετών, καθηγητών και εύρος ημερομηνιών. Τα φίλτρα ετικετών και καθηγητών είναι πολλαπλής επιλογής. Εκτός από τα φίλτρα υπάρχουν οι επιλογές ταξινόμησης και πλήθος ανακοινώσεων ανά σελίδα.



Εικόνα 4.2 Φίλτρα αναζήτησης

4.2.3 Εναλλαγή τρόπου εμφάνισης λίστας

Στο πάνω-δεξιά μέρος της προβολής υπάρχουν δύο κουμπιά που εναλλάσσουν τον τρόπο εμφάνισης της λίστας ανακοινώσεων σε λίστα και σε πλέγμα αντίστοιχα.



Εικόνα 4.3 Προβολή ανακοινώσεων με μορφή λίστας

4.3 Προβολή ανακοίνωσης

Σε αυτή την προβολή εμφανίζονται πλήρως οι πληροφορίες της ανακοίνωσης.

Δοκιμή 1

Το **Lorem Ipsum** είναι απλά ένα κείμενο χωρίς νόημα για τους επαγγελματίες της τυπογραφίας και στοιχειοθεσίας. Το Lorem Ipsum είναι το επαγγελματικό πρότυπο όσον αφορά το κείμενο χωρίς νόημα, από τον 15ο αιώνα, όταν ένας ανώνυμος τυπογράφος πήρε ένα δοκίμιο και ανακάτεψε τις λέξεις για να δημιουργήσει ένα δείγμα βιβλίου. Όχι μόνο επιβίωσε πέντε αιώνες, αλλά κυριάρχησε στην ηλεκτρονική στοιχειοθεσία, παραμένοντας με κάθε τρόπο αναλλοίωτο. Έγινε δημοφιλές τη δεκαετία του '60 με την έκδοση των δειγμάτων της Letraset όπου περιελάμβαναν αποσπάσματα του Lorem Ipsum, και πιο πρόσφατα με το λογισμικό ηλεκτρονικής σελιδοποίησης όπως το Aldus PageMaker που περιείχαν εκδοχές του Lorem Ipsum.

Γιατί το χρησιμοποιούμε;

Είναι πλέον κοινά παραδεκτό ότι ένας ανωνύμος αποσπάται από το περιεχόμενο που διαβάζει, όταν εξετάζει τη διαμόρφωση μίας σελίδας. Η ουσία της χρήσης του Lorem Ipsum είναι ότι έχει λίγο-πολύ μία ομαλή κατανομή γραμμάτων, αντίθετα με το να βάλει κανείς κείμενο όπως 'Εδώ θα μπει κείμενο, εδώ θα μπει κείμενο', κάνοντάς το να φαίνεται σαν κανονικό κείμενο. Πολλά λογισμικά πακέτα ηλεκτρονικής σελιδοποίησης και επεξεργαστές ιστότοπων πλέον χρησιμοποιούν το Lorem Ipsum σαν προκαθορισμένο δείγμα κειμένου, και η αναζήτηση για τις λέξεις 'Lorem ipsum' στο διαδίκτυο θα αποκαλύψει πολλά web site που βρίσκονται στο στάδιο της δημιουργίας. Διάφορες εκδόσεις έχουν προκύψει με το πέρασμα των χρόνων, άλλες φορές κατά λάθος, άλλες φορές σκόπιμα (με σκοπό το χιούμορ και άλλα συναφή).

Από που προέρχεται;

Αντίθετα με αυτό που θεωρεί η πλειοψηφία, το Lorem Ipsum δεν είναι απλά ένα τυχαίο κείμενο. Οι ρίζες του βρίσκονται σε ένα κείμενο Λατινικής λογοτεχνίας του 45 π.Χ., φτάνοντας την ηλικία του πάνω από 2000 έτη. Ο Richard McClintock, καθηγητής Λατινικών στο κολλέγιο Hampden-Dydney στην Βιρτζίνια

Εκδηλώσεις

Συνημμένα: 49420bea1c664fe09e47acbc01162588_9366.jpg

Εικόνα 4.4 Πλήρης προβολή ανακοίνωσης

4.4 Προβολή About

Στην προβολή About εμφανίζονται πληροφορίες για το έργο με την λίστα όσων εργάστηκαν και συνεργάστηκαν για να φτάσει στην σημερινή μορφή.

Aboard Ανακοινώσεις Διαχείριση ανακοινώσεων About Apps

ΚΟΥΣΤΟΥΛΑΣ ΘΕΟΦΑΝΗΣ

Σχετικά με το project

Το παρόν σύστημα είναι αποτέλεσμα συνεργασίας πολλών φοιτητών στο πλαίσιο εκπόνησης πτυχιακής ή και διπλωματικής εργασίας. Κατά χρονική σειρά (συν-)εργάστηκαν οι:

- Σιδηρόπουλος Αντώνης: Αρχική ιδέα, σχεδιασμός και επίβλεψη.
- Νικολαΐδης Νικόλας-Χρήστος: Πτυχιακή εργασία, 2018-2020. Σχεδιασμός και ανάπτυξη της αρχικής έκδοσης της web εφαρμογής (API και web client).
- Ραφαήλ Μονογιός: Πτυχιακή εργασία, 2019-2020. Υλοποίηση GUI Client για Android.
- Μπαρμπας Αντώνιος: Πτυχιακή εργασία, 2020-2021. Σύστημα διάδοσης ειδοποιήσεων.
- Στίνης Γεώργιος: Πτυχιακή εργασία, 2021-2022. Βελτίωση/Επανασχεδιασμός web client.
- Παπαδόπουλος Παντελεήμων-Νεκτάριος: Πτυχιακή εργασία, 2021-2022. Βελτίωση/Επανασχεδιασμός WEB API.
- Θεοφάνης Κουστόυλας: Πτυχιακή εργασία, 2022-2023. Βελτίωση/Επανασχεδιασμός web API και web client.

Το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων ευχαριστεί όλους τους παραπάνω και όλους όσους συνέβαλαν άμεσα ή έμμεσα στην ανάπτυξη αυτού του συστήματος.

Για περισσότερες πληροφορίες σχετικά με το project επισκεφθείτε την σελίδα του project στο [Github](#).

Εικόνα 4.5 Προβολή about

4.5 Απλός συνδεδεμένος χρήστης

Ο απλός συνδεδεμένος χρήστης χωρίς δικαιώματα author και admin έχει πρόσβαση σε όλες τις παραπάνω προβολές και η μόνη διαφοροποίηση υπάρχει στην λίστα ανακοινώσεων όπου εμφανίζονται και ανακοινώσεις που ανήκουν σε ιδιωτικές ετικέτες.

4.5.1 Σύνδεση

Για να συνδεθεί ο χρήστης θα πρέπει να πατήσει το κουμπί Σύνδεση. Έπειτα θα ανακατευθυνθεί στην πλατφόρμα login.iee.ihu.gr για να εισάγει τα στοιχεία σύνδεσης.

Εικόνα 4.6 Login του συστήματος login.iee.ihu.gr

4.5.2 Προβολή Λογαριασμός

Στο μενού λογαριασμός ο χρήστης έχει το δικαίωμα να μεταβάλλει τις ετικέτες που ακολουθεί.

Εικόνα 4.7 Προβολή λογαριασμός - Ακολουθήστε ετικέτες

Αξίζει να σημειωθεί πως τα επιλεγμένα στοιχεία ομαδοποιούνται και θέτουν ως επιλεγμένο στοιχείο μόνο το γενικό όταν είναι όλα επιλεγμένα.

Ακολουθήστε ετικέτες

Ακολουθώντας ετικέτες μπορείτε να ενημερώνεστε για νέες ανακοινώσεις στις ετικέτες που σας ενδιαφέρουν

Επιλέξτε ετικέτες

1ο Εξάμηνο x

- ☐ Test board
- ☐ Τηλε-εκπαίδευση
- ☐ Διδακτορικά
- ☐ Σπουδαστικά Θέματα ΠΠΣ
 - ☐ Πτυχιακές Εργασίες
 - ☐ Παρουσιάσεις πτυχιακών
 - ☐ Πτυχιακή/Διπλωματική Εργασία
- ☐ 1ο Έτος
 - ☒ 1ο Εξάμηνο
 - ☒ 1101 Μαθηματικά I
 - ☒ 1102 Δομημένος Προγραμματισμός
 - ☒ 1103 Εισαγωγή στην Επιστήμη των Υπολογιστών
 - ☒ 1104 Ηλεκτρονική Φυσική
 - ☒ 1105 Κυκλώματα Συνεχούς Ρεύματος
 - ☐ 2ο Εξάμηνο
- ☐ 2ο Έτος
- ☐ 3ο Έτος

Επιλεγμένες ετικέτες

1ο Εξάμηνο

Εικόνα 4.8 Ομαδοποίηση επιλεγμένων στοιχείων

Επίσης ο λογαριασμός περιέχει μία ενότητα αναφοράς προβλήματος.

Aboard
Ανακοινώσεις
Διαχείριση ανακοινώσεων
About
Apps

ΚΟΥΣΤΟΥΛΑΣ ΘΕΟΦΑΝΗΣ

Ετικέτες
Αναφορά προβλήματος

Αναφορά προβλήματος

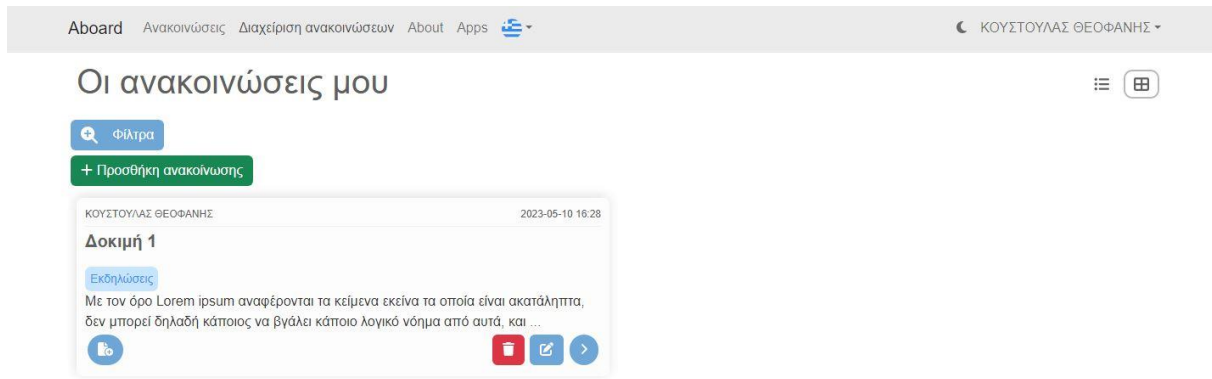
Για να αναφέρετε κάποιο πρόβλημα πατήστε

εδώ

Εικόνα 4.9 Προβολή αναφοράς προβλήματος

4.6 Author

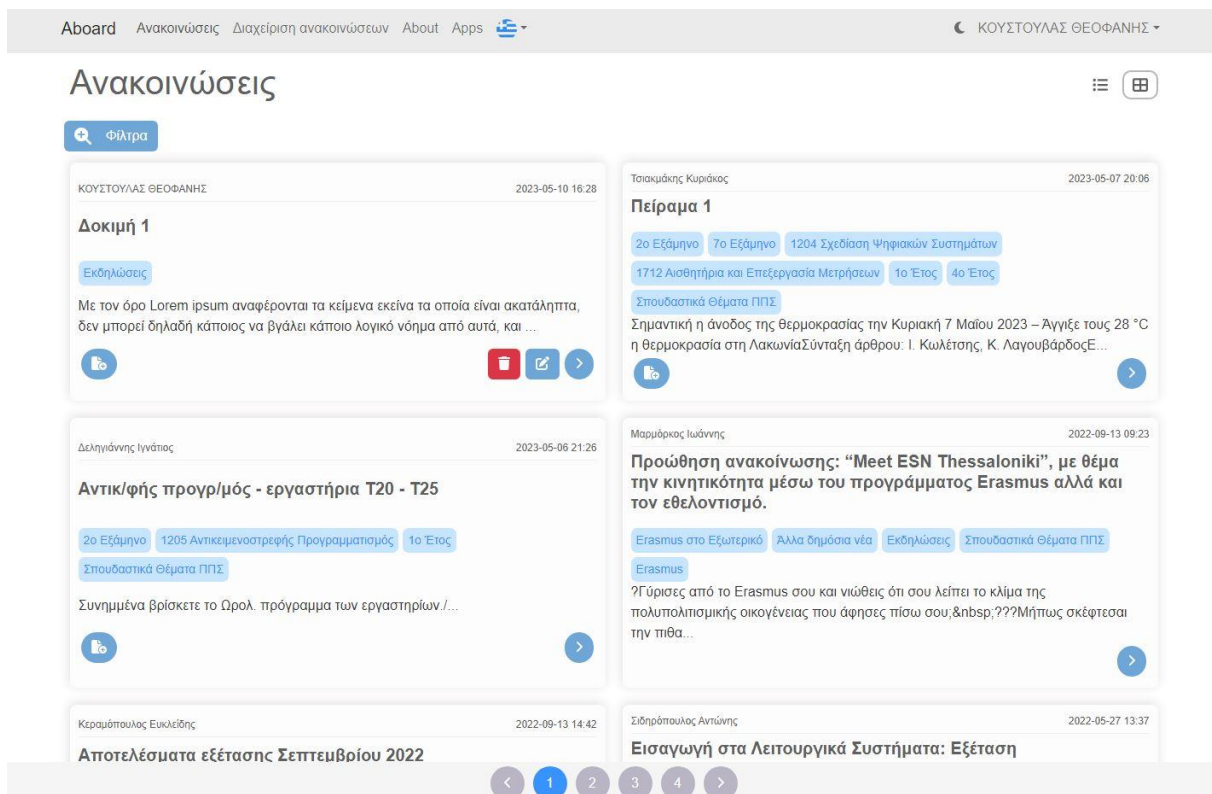
Ο author έχει πρόσβαση σε όλες τις προβολές έχει και ένας απλός χρήστης με επιπρόσθετη την προβολή την διαχείριση ανακοινώσεων. Στην διαχείριση ανακοινώσεων εμφανίζεται η λίστα των ανακοινώσεων του χρήστη με ίδιες επιλογές και φίλτρα όπως στην προβολή ανακοινώσεων και επιπρόσθετα υπάρχει το κουμπί Προσθήκη ανακοίνωσης. Επίσης οι ανακοινώσεις που ανήκουν στον συγκεκριμένο χρήστη εμφανίζουν το κουμπί Επεξεργασία.



Εικόνα 4.10 Προβολή Οι ανακοινώσεις μου

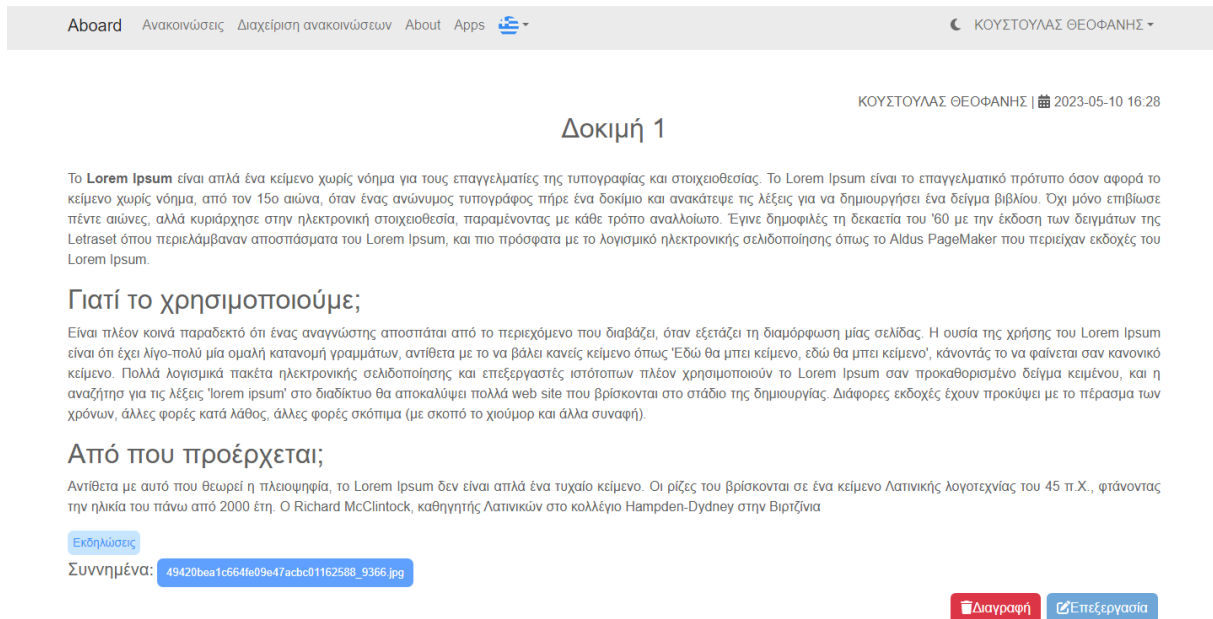
4.6.1 Actions ανακοινώσεων

Στην προβολή ανακοινώσεων ο author βλέπει παραπάνω action buttons στις ανακοινώσεις που του ανήκουν. Συγκεκριμένα βλέπει το button επεξεργασίας ανακοίνωσης και διαγραφής ανακοίνωσης.



Εικόνα 4.11 Actions ανακοινώσεων

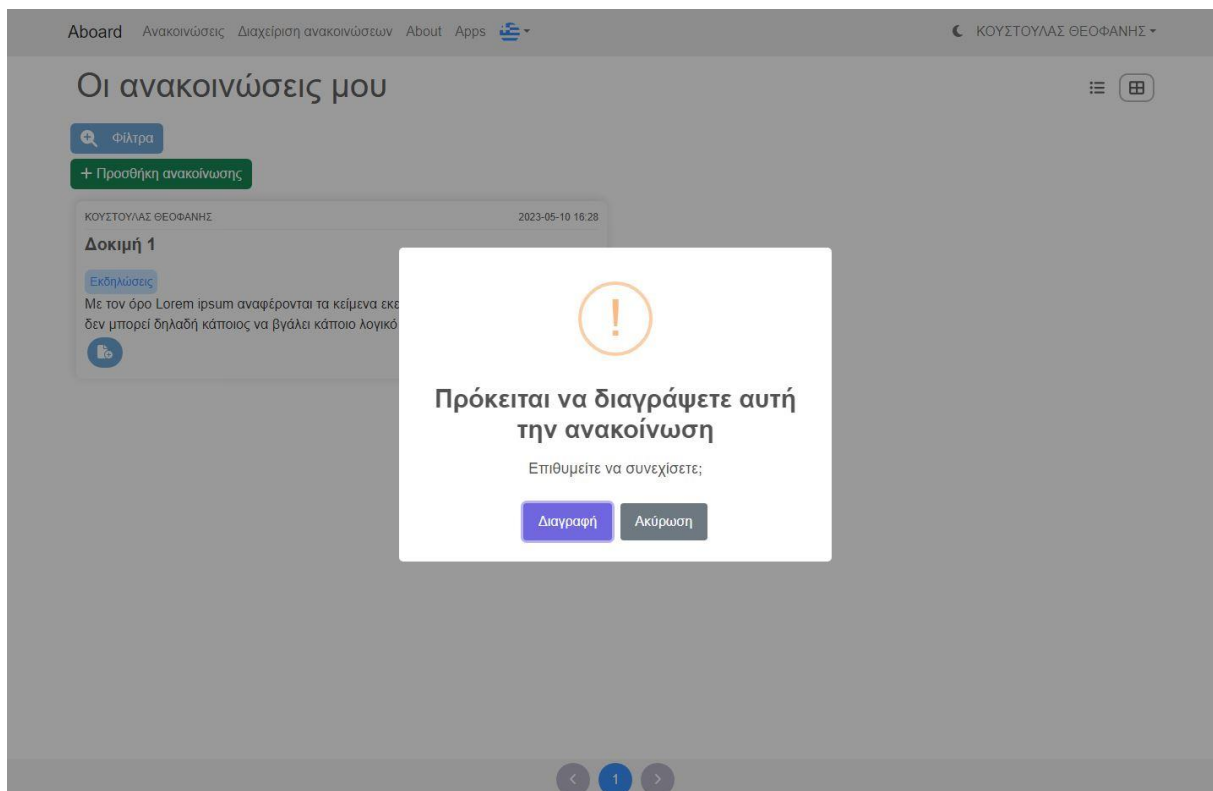
Επίσης τα ίδια action buttons εμφανίζονται και στην προβολή ανακοίνωσης.



Εικόνα 4.12 Action button στην προβολή μίας ανακοίνωσης

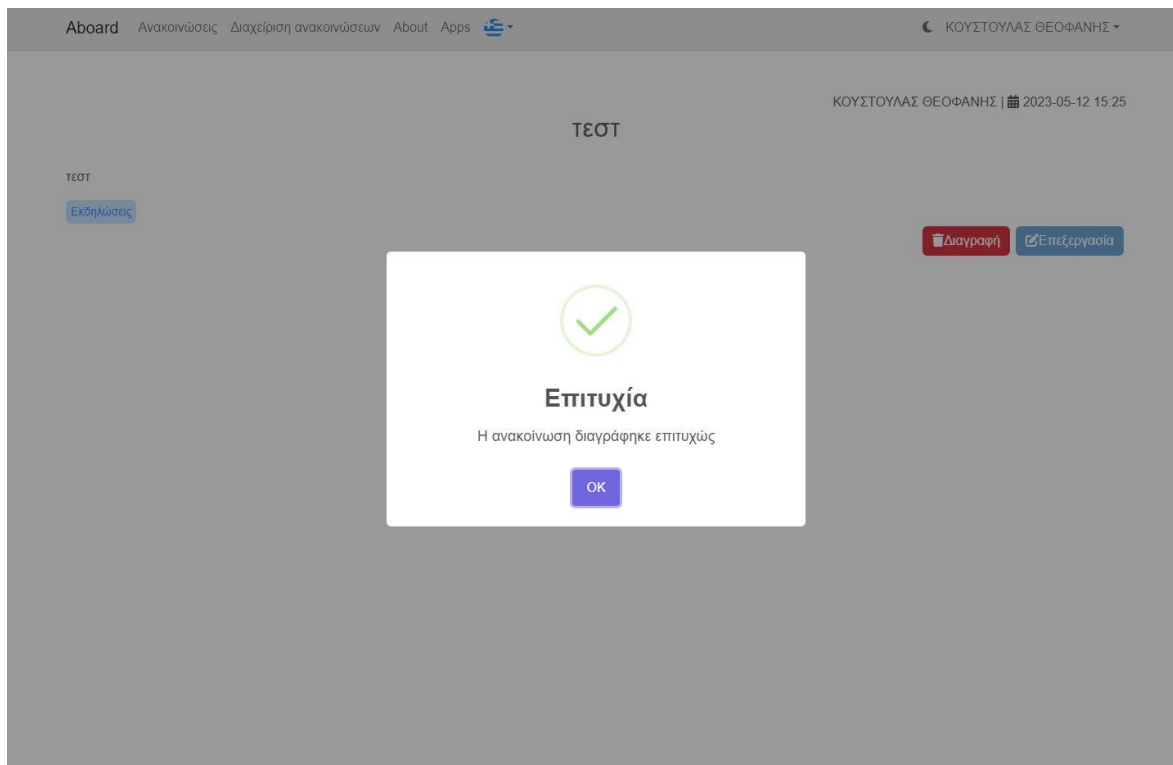
4.6.2 Διαγραφή ανακοίνωσης

Πατώντας το κουμπί διαγραφής θα εμφανιστεί ένα popup επιβεβαίωσης ενέργειας για λόγους ασφαλείας.



Εικόνα 4.13 Popup επιβεβαίωσης διαγραφής

Επιβεβαιώνοντας την διαγραφή εμφανίζεται δεύτερο popup που ενημερώνει τον χρήστη για την ολοκλήρωση της ενέργειας.



Εικόνα 4.14 Ολοκλήρωση διαγραφής

4.6.3 Εισαγωγή ανακοίνωσης

Από την φόρμα ανακοινώσεων ο χρήστης μπορεί να εισάγει μία νέα ανακοίνωση. Έχει επιλεχθεί να εμφανίζονται πάντα τα πεδία της αγγλικής μετάφρασης ως προτροπή προς τον χρήστη να εισάγει και στα αγγλικά τα στοιχεία της ανακοίνωσης για να εξασφαλιστεί σε μεγαλύτερο βαθμό η προσβασιμότητα. Τα σημεία εισαγωγής κειμένου υλοποιούν τις απαιτήσεις που αναφέρονται στο κεφάλαιο 1.3.

Προσθήκη ανακοίνωσης

Τίτλος
Προσθέστε έναν τίτλο ανακοίνωσης

Τίτλος στα αγγλικά
Προσθέστε έναν τίτλο ανακοίνωσης

Κείμενο

Κείμενο στα αγγλικά

Αποθήκευση

Εικόνα 4.15 Προβολή εισαγωγής κειμένων ανακοίνωσης

Επίσης δίνεται η δυνατότητα στον χρήστη να εισάγει ετικέτες, να ορίσει την ανακοίνωση ως σημαντική για συγκεκριμένο χρονικό διάστημα, να εισάγει πληροφορίες εκδήλωσης και επισυναπτόμενα αρχεία.

Ετικέτες
Ετικέτες που χρησιμοποιείτε συχνά
☐ Εκδηλώσεις
☐ Σπουδαστικά Θέματα ΠΠΣ
Επιλέξτε ετικέτες

☒ Επισήμανση ως σημαντική

Εμφάνιση μέχρι
31-05-2023

☒ Προσθήκη εκδήλωσης

Τοποθεσία
TEST ΤΟΠΟΘΕΣΙΑ

Ημ/νία έναρξης
25-05-2023 15:00

Ημ/νία λήξης
29-05-2023 15:00

Ανέβασμα αρχείων

Αποθήκευση

Εικόνα 4.16 Προβολή επιπρόσθετων στοιχείων ανακοίνωσης

4.6.4 Επεξεργασία ανακοίνωσης

Στην επεξεργασία ανακοίνωσης εμφανίζεται η φόρμα ανακοινώσεων με προ φορτωμένα τα δεδομένα της ανακοίνωσης. Ειδικά για τις φωτογραφίες στα κείμενα έγιναν πολλά trial and error ώστε να φτάσουμε στο επιθυμητό αποτέλεσμα της κανονικής φόρτωσης φωτογραφιών στο επιλεγμένο μέγεθος.

Aboard

Ανακοινώσεις

Διαχείριση ανακοινώσεων

About

Apps

ΚΟΥΣΤΟΥΛΑΣ ΘΕΟΦΑΝΗΣ

Επεξεργασία ανακοίνωσης

Τίτλος

Δοκιμή 1


Τίτλος στα αγγλικά

Test 1

Κείμενο

B I U <> H H1 H2 H3 H4 H5 H6 <>


Με τον όρο **Lorem ipsum** αναφέρονται τα κείμενα εκείνα τα οποία είναι ακατάληπτα, δεν μπορεί δηλαδή κάποιος να βγάλει κάποιο λογικό νόημα από αυτά, και έχουν δημιουργηθεί με σκοπό να παρουσιάσουν στον αναγνώστη μόνο τα γραφιστικά χαρακτηριστικά, αυτά καθ' εαυτά



Κείμενο στα αγγλικά

B I U <> H H1 H2 H3 H4 H5 H6 <>

Lorem ipsum is simply dummy text of the printing and typesetting industry. Lorem ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop



Ετικέτες

Εκδηλώσεις

☒ Μεταφορά ανακοίνωσης σε άλλο καθηγητή

Επιλέξτε καθηγητή

☐ Επισήμανση ως σημαντική

☒ Προσθήκη εκδήλωσης

Τοποθεσία

test

Ημ/νία έναρξης

18-05-2023 12:00

Ημ/νία λήξης

31-05-2023 12:00

Ανέβασμα αρχείων

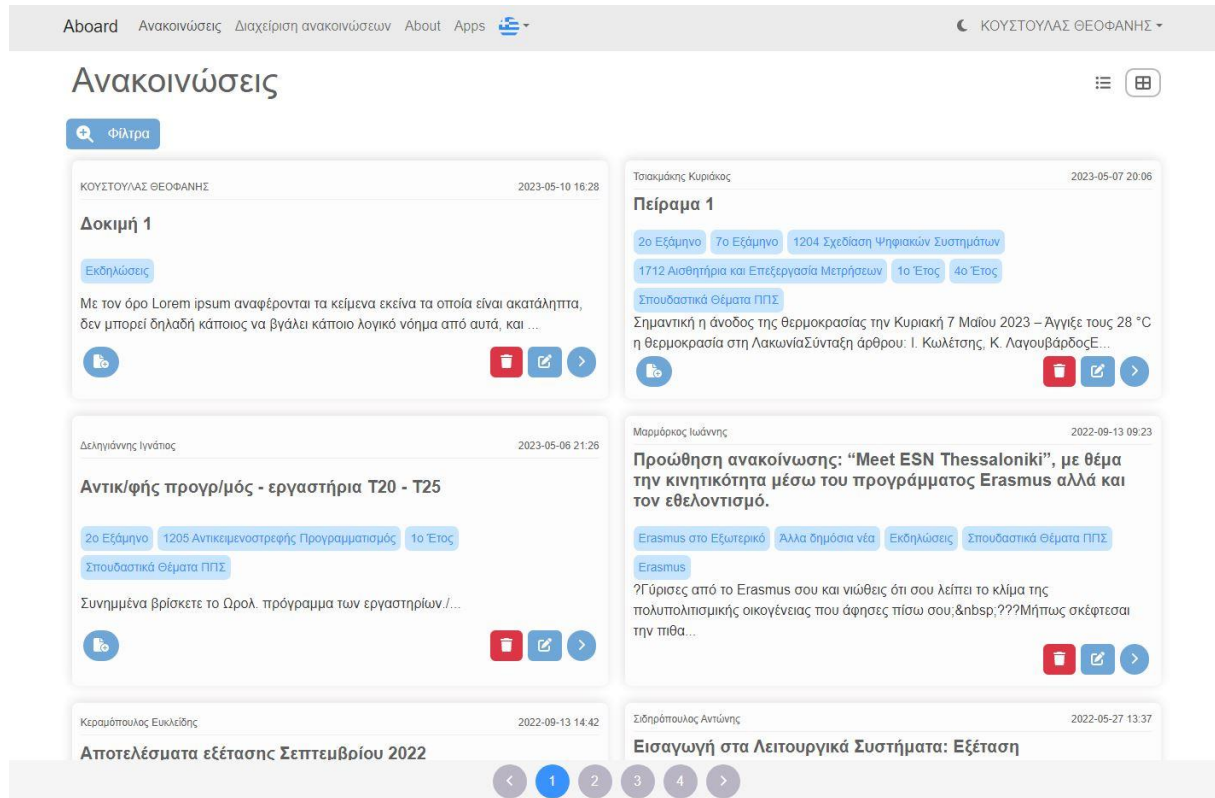
49420bea1c664fe09e47acbc01162588_9366.jpg

Αποθήκευση

Εικόνα 4.18 Προβολή επεξεργασίας ανακοίνωσης 2

4.7 Admin

Ο admin έχει πρόσβαση σε όλες τις παραπάνω προβολές με δύο επιπρόσθετα πλεονεκτήματα. Μπορεί να επεξεργαστεί όλες τις ανακοινώσεις και μπορεί να θέσει μία ανακοίνωση σε άλλο author.



Εικόνα 4.19 Προβολή δικαιωμάτων διαχειριστή σε ανακοινώσεις τρίτων

Κείμενο στα αγγλικά

B I U < > H

☰ ☲ ☱ ☳ ☴ ☵

Ετικέτες

Ετικέτες που χρησιμοποιείτε συχνά

☐ Εκδηλώσεις

☐ Σπουδαστικά Θέματα ΠΠΣ

Επιλέξτε ετικέτες

☒ Μεταφορά ανακοίνωσης σε άλλο καθηγητή

[352] Σιδηρόπουλος Αντώνης

☐ Επισήμανση ως σημαντική

☐ Προσθήκη εκδήλωσης

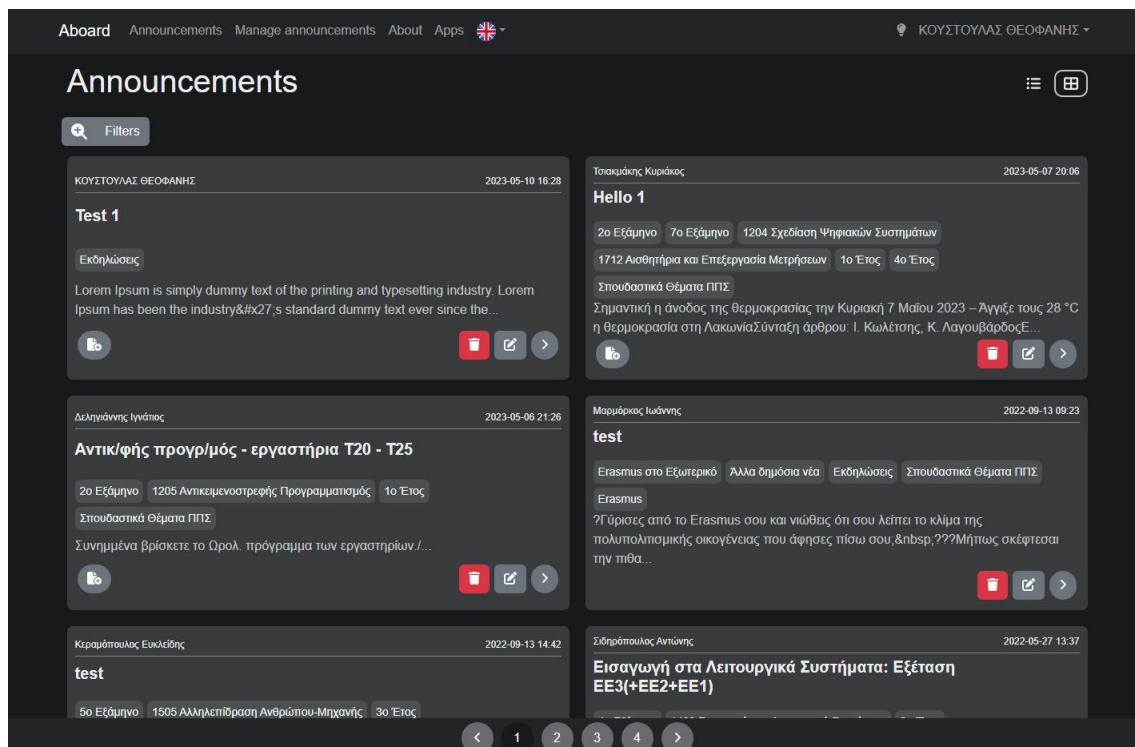
▶ Ανέβασμα αρχείων

Αποθήκευση

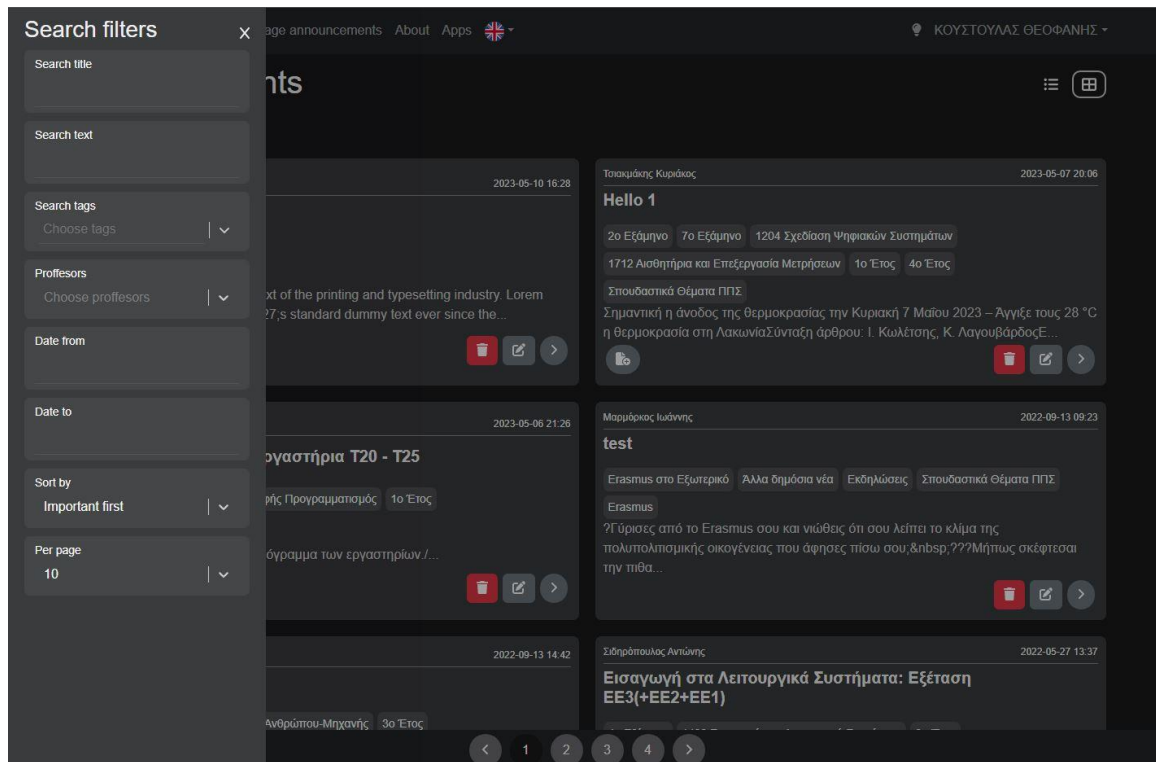
Εικόνα 4.20 Προβολή πεδίου μεταφοράς ανακοίνωσης σε άλλο author

4.8 Πολυθεματικότητα και πολυγλωσσικότητα

Στο header της εφαρμογής δίνεται η δυνατότητα στον χρήστη να επιλέξει το θέμα της εφαρμογής ανάμεσα σε σκούρο και φωτεινό. Επίσης μπορεί να αλλάξει την γλώσσα που προβάλλεται η εφαρμογή από Ελληνικά σε Αγγλικά.



Εικόνα 4.21 Προβολή με σκούρο θέμα και αγγλική γλώσσα



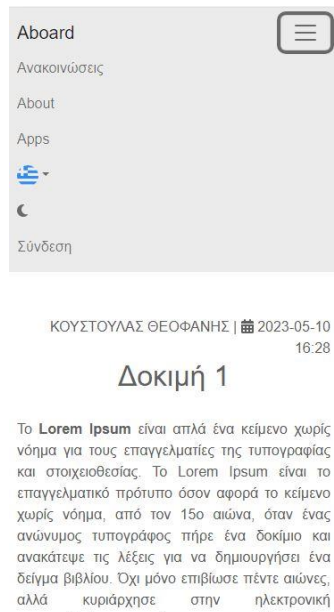
Εικόνα 4.22 Προβολή φίλτρων σε σκούρο θέμα

4.9 Responsive

Η εφαρμογή έχει αναπτυχθεί με τρόπο που διευκολύνει την πρόσβαση τόσο από desktop όσο και από tablet και smartphones. Τα στοιχεία των προβολών στοιχίζονται έτσι ώστε να διευκολύνεται η μετάδοση της πληροφορίας και ο χειρισμός.

4.9.1 Μενού πλοήγησης

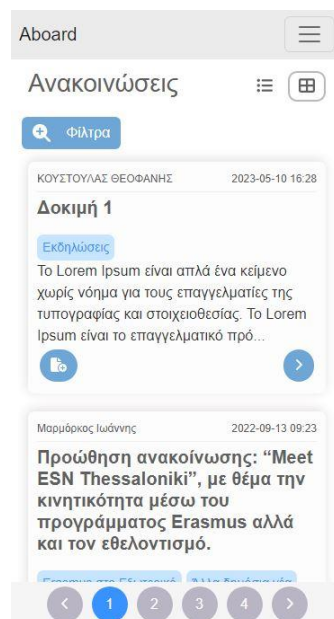
Το μενού πλοήγησης σε συσκευές με μικρή οθόνη μεταβάλλεται σε κρυφό μενού που εμφανίζεται στοιχισμένο κάθετα μετά από το πάτημα ενός κουμπιού όπως στα περισσότερα πρότυπα σχεδίασης responsive εφαρμογών.



Εικόνα 4.23 Responsive header

4.9.2 Ανακοινώσεις


Η διάταξη των ανακοινώσεων μεταβάλλεται σε σχέση με το πλάτος της οθόνης για να εμφανίζονται οι πληροφορίες με κατανοητό τρόπο. Συγκεκριμένα σε κάθε γραμμή εμφανίζεται μία ανακοίνωση αντί για δύο.



Εικόνα 4.24 Προβολή ανακοινώσεων σε smartphone

4.9.3 Φόρμα ανακοίνωσης

Η φόρμα ανακοίνωσης εμφανίζεται σε κάθετη μορφή με ένα input ανά σειρά.




Aboard 

Επεξεργασία ανακοίνωσης

Τίτλος
 Δοκιμή 1

Τίτλος στα αγγλικά
 Test 1


Κείμενο

B *I* U <> **H**   


<>

Το **Lorem Ipsum** είναι απλά ένα κείμενο χωρίς νόημα για τους επαγγελματίες της τυπογραφίας και στοιχειοθεσίας. Το Lorem Ipsum είναι το επαγγελματικό πρότυπο όσον αφορά το κείμενο χωρίς νόημα, από τον 15ο αιώνα, όταν ένας ανώνυμος τυπογράφος πήρε ένα δοκίμιο και ανακάτεψε τις λέξεις για να δημιουργήσει ένα δείγμα βιβλίου. Όχι μόνο επιβίωσε πέντε αιώνες, αλλά κυριάρχησε στην ηλεκτρονική στοιχειοθεσία, παραμένοντας με κάθε

Εικόνα 4.25 Φόρμα ανακοίνωσης responsive

Ετικέτες
 Εκδηλώσεις x 


☒ Μεταφορά ανακοίνωσης σε άλλο καθηγητή


Επιλέξτε καθηγητή 


☐ Επισήμανση ως σημαντική

☒ Προσθήκη εκδήλωσης

Τοποθεσία
 test

Ημ/νία έναρξης
 18/05/2023 12:00 μμ 

Ημ/νία λήξης
 31/05/2023 12:00 μμ 

 **Ανέβασμα αρχείων**

Αποθήκευση

Εικόνα 4.26 Φόρμα ανακοίνωσης λοιπά πεδία responsive

4.10 Επίλογος

Σε αυτό το κεφάλαιο παρουσιάστηκαν όλες οι προβολές της εφαρμογής από κάθε πιθανή κατάσταση χρήστη και συσκευής ώστε να γίνει κατανοητός ο τρόπος λειτουργίας.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει μία σύνοψη της εφαρμογής και θα προταθούν κάποιες μελλοντικές βελτιώσεις. Στην συνέχεια θα αναλυθούν τα τελικά συμπεράσματα από τον επανασχεδιασμό και την ανάπτυξη της εφαρμογής.

5.2 Σύνοψη

Η εφαρμογή aboard υλοποιεί ένα πλήρες σύστημα ανακοινώσεων με σκοπό την ενημέρωση των φοιτητών του τμήματος μηχανικών πληροφορικής και ηλεκτρονικών συστημάτων και αποτελείται από ένα API το οποίο είναι υπεύθυνο για την διατήρηση των πληροφοριών των χρηστών και των δικαιωμάτων τους, των ανακοινώσεων, των ετικετών, των subscriptions σε ετικέτες και των επισυναπτόμενων αρχείων ανακοινώσεων. Το API υλοποιεί όλο το CRUD (Create, Update, Delete) για τις παραπάνω οντότητες και τον συνδυασμό τους. Επίσης παρέχει σύστημα αυθεντικοποίησης εκμεταλλευόμενο των υπηρεσιών SSO που δίνονται από το σύστημα login.iee.ihu.gr. Η αυθεντικοποίηση του συστήματος είναι stateless που σημαίνει ότι η διατήρηση της σύνδεσης των χρηστών εφαρμόζεται με την χρήση και διάδοση JWT. Εκτός από το API η εφαρμογή παρέχει και μία ιστοσελίδα υλοποιημένη σε ReactJS. Η ιστοσελίδα περιέχει λίστα ανακοινώσεων με δυνατότητες φιλτραρίσματος, σελιδοποίησης και μορφοποίησης των αποτελεσμάτων. Επίσης δίνεται η δυνατότητα δήλωσης ετικετών που θα ήθελε ο χρήστης να ενημερώνεται για νέες ανακοινώσεις. Για τους authors υπάρχει το μενού διαχείρισης ανακοινώσεων όπου από εκεί μπορούν να εισάγουν, να επεξεργαστούν και να διαγράψουν τις ανακοινώσεις τους. Τέλος, για τον admin δίνεται πλήρης έλεγχος των ανακοινώσεων για εισαγωγή, επεξεργασία, διαγραφή και μετάθεση μιας ανακοίνωσης σε άλλο author.

5.3 Προτάσεις βελτίωσης

Με το τέλος του επανασχεδιασμού του έργου είναι σημαντικό να γίνει ένας απολογισμός ως προς τα σημεία που θα μπορούσαν να αλλάξουν ή τις προσθήκες που θα έπρεπε να γίνουν ώστε να βελτιωθεί ακόμα περισσότερο η εφαρμογή.

5.3.1 Αναβάθμιση έκδοσης Laravel

Η Laravel 6 όπως αναφέραμε και στο 2^ο κεφάλαιο δημοσιεύτηκε το 2019 και παρότι ακόμη αποτελεί ένα αρκετά χρήσιμο εργαλείο δεν μπορεί να συγκριθεί με τις δυνατότητες και τις διορθώσεις που έχουν η Laravel 9 και 10. Πιο συγκεκριμένα στις νεότερες εκδόσεις δίνεται η δυνατότητα για metrics σε Unit και Integration tests που κατατοπίζουν τον προγραμματιστή ως προς τα κομμάτια της εφαρμογής που θα μπορούσαν να βελτιωθούν. Επίσης οι διαδικασίες του query builder έχουν βελτιωθεί αρκετά, διορθώνοντας προβλήματα όπως για παράδειγμα το pagination με χρήση group by που δεν υποστηρίζεται στην έκδοση 6. Τέλος, από τον Μάρτιο του 2021 έχουν σταματήσει τα Security updates για την έκδοση 6 οπότε η αναβάθμιση σε νεότερη έκδοση θα ήταν η καλύτερη αφετηρία για την βελτίωση της εφαρμογής.

5.3.2 Υλοποίηση Unit και Integration tests

Ένα πολύ σημαντικό κομμάτι για την συντήρηση μίας εφαρμογής είναι το Unit testing και το Integration testing. Τα unit tests αφορούν δοκιμές συγκεκριμένων λειτουργιών όπως μία συγκεκριμένη μέθοδο για να διασφαλιστεί η εύρυθμη λειτουργία τους. Από την άλλη τα Integration tests αφορούν μεγαλύτερα

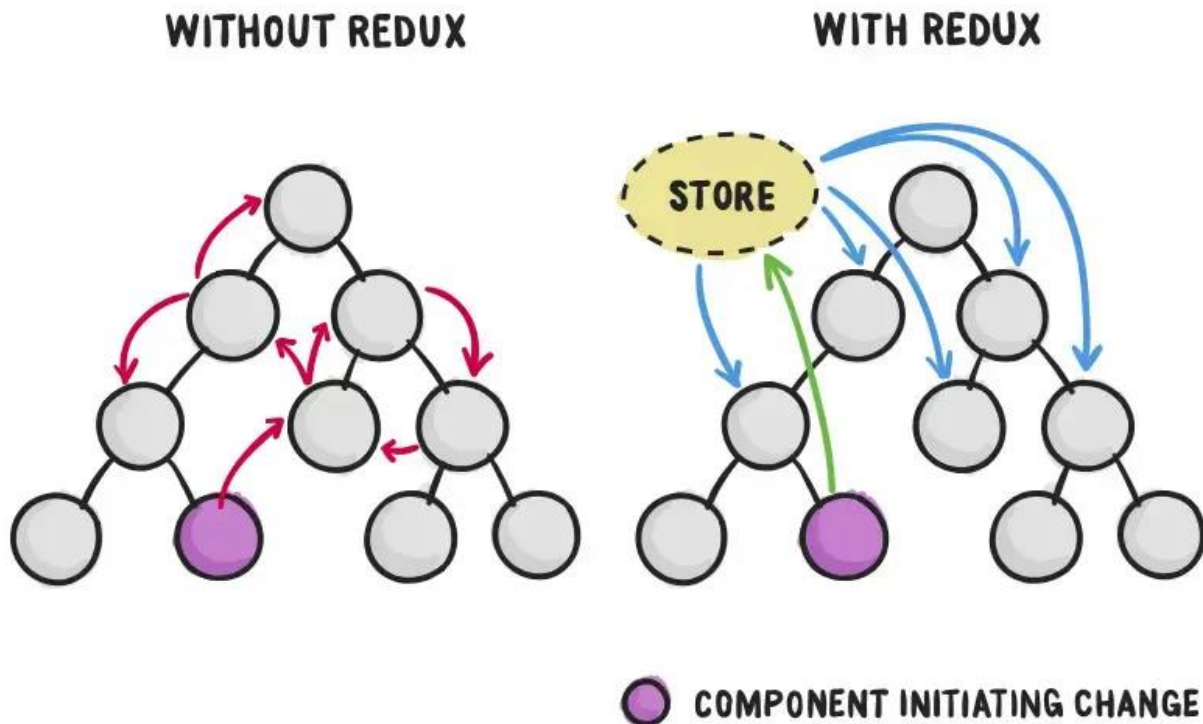
σύνολα λειτουργιών όπως μία δέσμη λειτουργιών για την εκπλήρωση ενός σκοπού. Κατά την αρχική ανάπτυξη του λογισμικού τα test βοηθάνε στο να βεβαιωθούμε ότι η νέα υλοποίηση είναι σωστή. Ωστόσο η μεγάλη χρησιμότητα των unit και integration tests εμφανίζεται όταν η εφαρμογή επεκτείνεται και ανανεώνονται οι λειτουργίες της. Σε αυτό το στάδιο πολλές φορές χρειάζεται να αλλάξουν πολλά σημεία με αποτέλεσμα να μην γνωρίζουμε με βεβαιότητα ότι οι αλλαγές που έχουν γίνει δεν επηρέασαν αρνητικά κάποια λειτουργία. Χωρίς τα test θα έπρεπε να δοκιμάσουμε χειροκίνητα κάθε κομμάτι της εφαρμογής έπειτα από την ολοκλήρωση αλλαγών. Σε μικρά project θα μπορούσαμε να κάνουμε τις παραπάνω δοκιμές χειροκίνητα αλλά σε μεγάλα project κάτι τέτοιο αποτελεί μία χρονοβόρα ή και αδύνατη διαδικασία. Έχοντας υλοποιήσει τα test όμως έπειτα από κάθε αλλαγή, με μία μόνο εντολή μπορούμε να βεβαιωθούμε ότι δεν έχει δημιουργηθεί κάποια δυσλειτουργία στο σύστημα.

5.3.3 React tests

Όπως στο back-end έτσι και στο front-end είναι χρήσιμο να δημιουργηθούν δοκιμές που να εξασφαλίζουν την ορθή λειτουργία του συστήματος μετά από κάποιο update. Για τον σκοπό αυτό υπάρχουν βιβλιοθήκες που υλοποιούν δοκιμές σε επίπεδο component, user event και μεθόδου. Μία από αυτές τις βιβλιοθήκες που θα πρότεινα είναι η Jest η οποία έχει δημιουργηθεί από την εταιρία Meta όπως και η ReactJS και διαθέτει μεγάλη γκάμα από εργαλεία για τις ανάγκες δοκιμών.

5.3.4 Redux

Όπως είπαμε και σε προηγούμενα κεφάλαια, το κάθε component της ReactJS περιέχει το state του και τα props του. Για εφαρμογές χαμηλής πολυπλοκότητας έχοντας μόνο τα δύο παραπάνω είμαστε καλυμμένοι. Ωστόσο όσο μία εφαρμογή επεκτείνεται, τόσο δυσκολότερη γίνεται η διαχείριση του state γιατί η πληροφορία είναι αποκεντροποιημένη και συντηρείται στο κάθε state ξεχωριστά. Αν θέλαμε για παράδειγμα να μοιράσουμε μία πληροφορία από ένα component δύο επίπεδα πιο πάνω θα έπρεπε να γράψουμε αρκετό επιπρόσθετο κώδικα και στο ενδιαμέσο component και στο component που βρίσκεται από πάνω. Επίσης με τον επιπρόσθετο κώδικα θα δεσμεύαμε ένα component να χειρίζεται με έναν πολύ συγκεκριμένο τρόπο. Όλη η παραπάνω διαδικασία δεν είναι συντηρήσιμη και δημιουργεί συνθήκες παραγωγής σφαλμάτων. Αυτό που θα έλυνε το παραπάνω πρόβλημα θα ήταν αν το state της εφαρμογής συνολικά συντηρούνταν σε ένα σημείο καθολικά. Αυτή την λύση μας παρέχει η βιβλιοθήκη Redux για την React. Μέσω της redux μπορούμε να διατηρούμε συνολικά το state της εφαρμογής σε ένα σημείο επιτυγχάνοντας την δημιουργία μίας μοναδικής πηγής αλήθειας ή αλλιώς Single Source Of Truth (SSOT). Επίσης κάθε component μπορεί να λάβει πληροφορίες σχετικά με το state καθώς και μεθόδους μεταβολής του state. Έχοντας όλα τα παραπάνω σαν γνώμονα, γίνεται εμφανές ότι με την προσθήκη της redux θα βελτιώναμε την εφαρμογή δραματικά.



Εικόνα 5.1 Σύγκριση διαχείρισης πληροφορίας χωρίς redux και με redux

5.3.5 Επιβολή ενός μόνο προτύπου συγγραφής κώδικα

Η ομοιομορφία του κώδικα δεν αποτελεί κρίσιμο στοιχείο για την τελική υλοποίηση του project αλλά η ύπαρξη ομοιομορφίας στον κώδικα παρέχει ευκολότερη κατανόηση τόσο από τους ανθρώπους που εργάζονται πάνω στο project όσο και στα εργαλεία συγγραφής κώδικα που θα μπορούν να αναγνωρίσουν ποια σχόλια περιγράφουν μία μέθοδο, μία κλάση και γενικότερα την δομή του έργου. Υπάρχουν πολλές εφαρμογές και extensions για δημοφιλή IDE που μορφοποιούν τον κώδικα σύμφωνα με κάποιο πρότυπο που έχουμε ορίσει, οπότε μία καλή πρακτική θα ήταν μελλοντικά να γίνει χρήση κάποιου από αυτά τα εργαλεία ώστε να επιτευχθεί ομοιομορφία του κώδικα καθολικά.

5.4 Συμπεράσματα

Το παρόν έργο αποτελεί μία προσπάθεια αρκετών ανθρώπων, όπου ο καθένας συνείσφερε θυσιάζοντας αρκετό από τον προσωπικό του χρόνο για να φτάσει στο σημείο που είναι σήμερα. Τελειώνοντας και εγώ με την σειρά μου τον επανασχεδιασμό της εφαρμογής, μέσω της διαδικασίας της ανάπτυξης εκτίμησα την υλοποίηση που έχει γίνει πριν από εμένα και μπορώ να πω πως πρόκειται για ένα σύστημα πλήρες και αποδοτικό. Αλλά όπως όλα έτσι και η τεχνολογία εξελίσσεται συνεχώς και μαζί με την τεχνολογία θα πρέπει να εξελιχθούν και τα συστήματα που την χρησιμοποιούν και όπως επανασχεδιάστηκε η εφαρμογή από εμένα θα επανασχεδιαστεί ξανά από κάποιον άλλον. Ωστόσο μέσα από την τριβή αυτή μπόρεσα να πάρω σημαντικές γνώσεις που θα αποτελέσουν δομικό λίθο για την πορεία μου στο μέλλον. Συγκεκριμένα κατά την υλοποίηση της εφαρμογής εξερεύνησα ένα καινούργιο για εμένα framework που είναι η Laravel. Τα κομμάτια της προηγούμενης υλοποίησης αποτέλεσαν και τα πρακτικά παραδείγματα συγγραφής που μείωσαν τον χρόνο εξοικείωσης σημαντικά.

5.5 Επίλογος

Σε αυτό το κεφάλαιο έγινε μία σύνοψη της τελικής εφαρμογής αναλύοντας τον σκοπό και τις λειτουργίες. Στην συνέχεια προτάθηκαν κάποιες βελτιώσεις που θα μπορούσαν να υλοποιηθούν μελλοντικά. Τέλος, τονίστηκαν τα συμπεράσματα που εξάχθηκαν από την υλοποίηση της εφαρμογής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] Lathbrige Tymothy C. and Laganriere Robert, *Object-Oriented Software Engineering*, Second edition. McGraw-Hill Education, 2005.
- [2] “History of PHP.” <https://www.php.net/manual/en/history.php.php> (accessed May 01, 2023).
- [3] M. I. Kausar Bagwan and P. D. Swati Ghule, “A Modern Review on Laravel-PHP Framework,” *IRE Journals*, vol. 2, no. 12, 2019.
- [4] “History of laravel.” <https://www.w3schools.in/laravel/history> (accessed May 01, 2023).
- [5] Arda. Kilicdagi, *Laravel Design Patterns and Best Practices*. Packt Publishing, 2014.
- [6] Parixit Patel, “Laravel request lifecycle,” Jun. 15, 2020. <https://dev.to/patelparixit07/laravel-request-lifecycle-195e> (accessed Apr. 29, 2023).
- [7] Jeel Patel, “PHP Framework Comparison – Which is The Best PHP Framework?,” Dec. 10, 2021. <https://www.monocubed.com/blog/php-framework-comparison/> (accessed Apr. 29, 2023).
- [8] “Introduction to composer.” <https://getcomposer.org/doc/00-intro.md> (accessed May 08, 2023).
- [9] A. Hickson, *HTML, The living standard*. Lulu Publishing Services, 2016.
- [10] A. Wirfs-Brock and B. Eich, “JavaScript: The first 20 years,” *Proceedings of the ACM on Programming Languages*, vol. 4, no. HOPL, Jun. 2020, doi: 10.1145/3386327.
- [11] Jordan Walke, “FaxJS Github page,” Dec. 04, 2011. <https://github.com/jordwalke/FaxJs> (accessed Apr. 29, 2023).
- [12] M. Fariz, S. Lazuardy, and D. Anggraini, “Modern Front End Web Architectures with React.Js and Next.Js,” *International Research Journal of Advanced Engineering and Science*, vol. 7, no. 1, pp. 132–141, 2022.
- [13] “ReactJS documentation.” <https://react.dev/learn/passing-props-to-a-component> (accessed Apr. 29, 2023).
- [14] “MySQL constraints,” May 01, 2023. https://www.w3schools.com/mysql/mysql_constraints.asp (accessed May 01, 2023).
- [15] “Understand and Evaluate Table Relationships,” May 01, 2023. <https://experienceleague.adobe.com/docs/commerce-business-intelligence/mbi/analyze/warehouse-manager/table-relationships.html?lang=en> (accessed May 01, 2023).
- [16] J. Cooper *et al.*, “Database Systems Journal,” *Database Systems Journal*, vol. 2, no. 5, 2011.