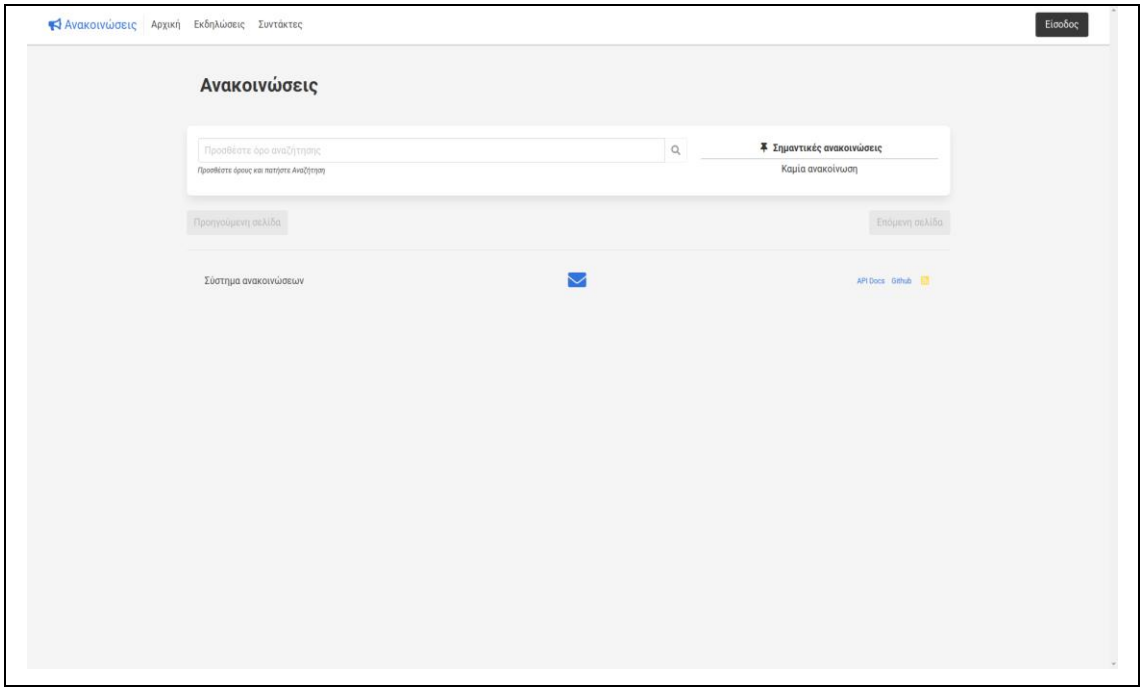
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«ΠΙΝΑΚΑΣ ΑΝΑΚΟΙΝΩΣΕΩΝ»



Του φοιτητή
Νικολαΐδη Νικόλαου - Χρήστου
Αρ. Μητρώου: 113763

Επιβλέπων
Ονοματεπώνυμο:
Σιδηρόπουλος Αντώνιος
Βαθμίδα: Επίκουρος Καθηγητής

ΘΕΣΣΑΛΟΝΙΚΗ, ΣΕΠΤΕΜΒΡΙΟΣ 2020

Τίτλος Δ.Ε.: ΠΙΝΑΚΑΣ ΑΝΑΚΟΙΝΩΣΕΩΝ
Κωδικός Δ.Ε.: 19005
Ονοματεπώνυμο φοιτητή: Νικολαΐδης Νικόλαος - Χρήστος
Ονοματεπώνυμο εισηγητή: Σιδηρόπουλος Αντώνιος
Ημερομηνία ανάληψης Δ.Ε.: 13/05/19
Ημερομηνία περάτωσης Δ.Ε.: 18/09/20

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 διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

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

«Η παρούσα πτυχιακή αφιερώνεται στους ανθρώπους που με στήριξαν όλα αυτά τα χρόνια των σπουδών μου. Χωρίς εσάς δε θα ήταν εφικτό αυτό το κείμενο.»

Πρόλογος

Την περίοδο επιλογής και υλοποίησης της παρούσης εργασίας εργαζόμουν ως διαχειριστής συστημάτων σε μία εταιρία. Δεν είχα ιδιαίτερη εμπειρία με τον προγραμματισμό. Για να πω την αλήθεια, μάλλον ήταν το χειρότερό μου. Η μόνη επαφή που είχα ήταν κάποιες εργασίες, μικρότερης ή μεγαλύτερης έκτασης, σε μαθήματα, στις οποίες δεν είχα να σκεφτώ εξ αρχής τα πάντα που αφορούσαν την εφαρμογή που θα έφτιαχνα. Οπότε, όταν επέλεγα το συγκεκριμένο θέμα, είχα αποφασίσει ήδη πως θα δώσω μία ευκαιρία στον τομέα του προγραμματισμού. Ασχολήθηκα με τεχνολογίες για το back-end και για το front-end που δεν είχα δει άλλη φορά, έφτιαξα το δικό μου API, έμαθα διάφορες καινούργιες τεχνολογίες. Επιπλέον, υπάρχει το ενδεχόμενο να χρησιμοποιηθεί από όλους τους καθηγητές και συναδέρφους μου στη σχολή. Όλα αυτά είναι πράγματα που με κάνουν να πιστεύω πως έκανα τη σωστή επιλογή για τη διπλωματική μου εργασία και, ίσως, να άνοιξα και έναν νέο τομέα για το επαγγελματικό μου μέλλον.

Περίληψη

Στα χρόνια των σπουδών μου, έχω συναντήσει δύο συστήματα ανακοινώσεων που χρησιμοποιήθηκαν. Το παλαιότερο σύστημα hydra και το τμήμα ανακοινώσεων στο νέο σύστημα με όνομα apps. Και τα δύο έχουν τα θετικά και τα αρνητικά τους. Όμως, το hydra πάλιωσε και αποφασίστηκε πως δεν είναι πλέον χρηστικό για το τμήμα. Έτσι, αποφασίστηκε να δημιουργηθεί το apps, το οποίο, καθώς αποτελεί ένα έργο δημιουργημένο από φοιτητές, πλέον δεν είναι εύκολο να συντηρηθεί, καθώς και οι συνάδερφοι «έχουν πάρει τους δρόμους τους», εργάζονται και δεν είναι σε θέση να σπαταλούν μεγάλο μέρος του χρόνου τους για το σύστημα.

Υπό αυτό το πλαίσιο, αποφασίστηκε πως χρειάζεται μία αλλαγή. Ένα σύστημα παραμετροποιημένο σύμφωνα με τις ανάγκες του τμήματος, που δε θα είχε τα αρνητικά των προηγούμενων συστημάτων, θα είναι εύκολα επεκτάσιμο και θα παρέχει API ώστε να είναι εύκολη η αλληλεπίδραση του με άλλα υπάρχοντα συστήματα (π.χ. website του τμήματος)..

«ΠΙΝΑΚΑΣ ΑΝΑΚΟΙΝΩΣΕΩΝ»

(ANNOUNCEMENT BOARD)

«Νικολαΐδης Νικόλαος - Χρήστος»

(Nikolaidis Nikolaos - Christos)

Abstract

In my time of studying, I have encountered two announcement board systems. The old system, named hydra, and the announcement section in the new system currently used, named apps. Both of them had their pros and cons. As hydra got old, it was decided that it was not usable for the department. Hence, apps was created, but since it was built by students, it is now very difficult to maintain, as my fellow students are now working full time and they are not in position to spend a big portion of their time maintaining the system.

Under these circumstances, a change was decided. A system built according to the needs of the department, which would not have the cons of the old systems, would be easily extensible and would provide an API so that interaction with other systems (e.g. department's website) would be very easy.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τους γονείς μου για την εμπιστοσύνη τους, για την οικονομική και ηθική στήριξη που μου παρείχαν καθ' όλη τη διάρκεια των σπουδών μου. Θα ήθελα, επίσης, να ευχαριστήσω τους κοντινούς μου ανθρώπους για τη στήριξη και τη βοήθειά τους.

Θα ήθελα, ακόμη, να ευχαριστήσω τον υπεύθυνο καθηγητή, κ. Αντώνη Σιδηρόπουλο, για την εμπιστοσύνη, την καθοδήγηση και τις ουσιώδεις συμβουλές που μου προσέφερε κατά τη διάρκεια εκπόνησης της παρούσης εργασίας. Τέλος, θα ήθελα να ευχαριστήσω και τα μέλη της εξεταστικής επιτροπής για τον χρόνο που αφιέρωσαν για την ανάγνωση και παρουσίαση της εργασίας..

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
API	Application Programming Interface
RSS	Rich Site Summary
LDAP	Lightweight Directory Access Protocol
IP	Internet Protocol
PHP	Hypertext Preprocessor
LTS	Long Term Support
MVC	Model View Controller
CLI	Command Line Interface
CSRF	Cross-site Request Forgery
ORM	Object-Relational Mapping
CMS	Content Management System
CRUD	Create, Read, Update, Delete
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
URL	Uniform Resource Locator
WSDL	Web Services Description Language
XML	eXtensible Markup Language
JSON	JavaScript Object Notation
RFC	Requests for Comments
DBMS	Database Management System
DB	Database
JWT	JSON Web Token
Ε.Λ.	Ενδιάμεσο Λογισμικό
CORS	Cross-origin Resource Sharing
CSS	Cascade Style Sheets
W3C	World Wide Web Consortium
JS	JavaScript
UI	User Interface
CDN	Content Delivery Network
SASS	Syntactically Awesome Style Sheets
SCSS	Sassy Cascade Style Sheets
npm	node package manager
DOM	Document Object Model
CPU	Central Processing Unit
TDD	Test-Driven Development

Κεφάλαιο 1ο: Ανάλυση απαιτήσεων

1.1 Ανάλυση απαιτήσεων

Στην παρακάτω λίστα αναλύονται οι απαιτήσεις που υπήρχαν κατά τη δημιουργία της εφαρμογής. Γίνεται παράθεσή τους και μία σύντομη ανάλυση της κάθε απαίτησης.

- **Υποστήριξη πολυμέσων και προεπισκόπησή τους κατά την εμφάνιση της ανακοίνωσης**:
  δυνατότητα να υπάρχουν πολυμέσα-αρχεία σε κάθε ανακοίνωση και τα οποία θα μεταφορτώνονται μαζί με την ανακοίνωση
- **Υποστήριξη API (Application Programming Interface):**
  χρειάζεται η εφαρμογή προσφέρει δυνατότητα αλληλεπίδρασης μέσω API για όλες τις λειτουργίες που θα εκτελούνται από αυτή
- **Υποστήριξη RSS/Atom Feed**:
  χρειάζεται η εφαρμογή να υποστηρίζει το πρότυπο RSS για το διαμοιρασμό μέρους των ανακοινώσεων
- **Αποθήκευση σε MySQL**:
  η εφαρμογή θα πρέπει να αποθηκεύει όλα τα δεδομένα της (χρήστες, ανακοινώσεις, πολυμέσα κ.ο.κ.) σε βάση MySQL
- **Δυνατότητα σήμανσης ανακοινώσεων ως σημαντικές**:
  κάθε ανακοίνωση θα πρέπει να έχει τη δυνατότητα να σημαίνεται ως σημαντική ώστε να εμφανίζεται στην αρχική σελίδα της εφαρμογής
- **Προσθήκη εκδηλώσεων**:
  κάθε ανακοίνωση θα πρέπει να έχει τη δυνατότητα να σημαίνεται ως εκδήλωση ώστε να φαίνεται σε ξεχωριστή σελίδα της εφαρμογής
- **Σύνδεση με το σύστημα αυθεντικοποίησης του τμήματος**:
  ο εκάστοτε χρήστης της εφαρμογής θα πρέπει να αυθεντικοποιείται με το σύστημα (LDAP) του τμήματος προτού του επιτραπεί η θέαση του συνόλου των ανακοινώσεων
- **Οργάνωση των ανακοινώσεων με tags**:
  κάθε ανακοίνωση θα πρέπει να συνδέεται με ένα ή περισσότερα tags για λόγους οργάνωσης και αναζήτησης
- **Διαχωρισμός tags σε public και private**:
  κάθε tag θα πρέπει να σημαίνεται ως public ή private αναλόγως με το κοινό στο οποίο θα επιτρέπεται η θέασή της
- **Ιεραρχική οργάνωση των tags**:
  τα tags θα πρέπει να υποστηρίζουν ιεραρχική οργάνωση, δηλαδή να μπορεί ένα tag να συνδέεται με ένα άλλο με τη μορφή parent-child
- **Εγγραφή χρήστη σε tags**:
  κάθε εγγεγραμμένος χρήστης θα πρέπει να μπορεί να εγγράφεται (subscribe) στα tags που τον ενδιαφέρουν ώστε να είναι σε θέση να λαμβάνει ενημερώσεις για τις ανακοινώσεις ενδιαφέροντος του

- **Αναζήτηση βάσει κριτηρίων**:
  κάθε χρήστης θα πρέπει να έχει τη δυνατότητα αναζήτησης στην εφαρμογή βάσει συγκεκριμένων κριτηρίων (τίτλος-περιεχόμενο ανακοίνωσης, tag, συντάκτης) και να γίνεται εμφάνιση μόνο των αποτελεσμάτων που είναι σε θέση να λάβει αναλόγως της κλάσης του (εγγεγραμμένος ή μη)
- **Εμφάνιση όλων των ανακοινώσεων κατά την είσοδο από το IP range του ΔΙΠΑΕ**:
  σε περίπτωση που ένας χρήστης εισέρχεται στην εφαρμογή από το δίκτυο του ΔΙΠΑΕ, θα πρέπει να «βλέπει» όλες τις ανακοινώσεις, τόσο αυτές που είναι χαρακτηρισμένες ως public αλλά και τις private. Δε θα πρέπει να έχει κάποια άλλη δυνατότητα χωρίς να έχει πραγματοποιήσει είσοδο στο σύστημα.

1.2 Επίλογος

Σ’ αυτό το κεφάλαιο αναλύθηκαν οι απαιτήσεις της παρούσης εφαρμογής. Λεπτομερής ανάλυση του τρόπου υλοποίησης της καθεμίας από αυτές θα ακολουθήσει στα επόμενα κεφάλαια.

Κεφάλαιο 2ο: Τεχνολογίες που χρησιμοποιήθηκαν

2.1 Εισαγωγή στις τεχνολογίες

Στο παρόν κεφάλαιο θα γίνει μία ανάλυση σε όλες τεχνολογίες που χρησιμοποιούνται στην εφαρμογή. Θα αναλυθεί το πώς χρησιμοποιείται το κάθε συστατικό μέσα στην εφαρμογή καθώς και ο λόγος που επιλέχτηκε η κάθε τεχνολογία.
Αναφορικά με την επιλογή του κάθε συστατικού, καλό είναι να υπενθυμίσουμε πως, καθώς η εφαρμογή έχει μεγάλες πιθανότητες να χρησιμοποιηθεί σε παραγωγικό περιβάλλον και θα συντηρηθεί από τρίτους, θα πρέπει να χρησιμοποιηθούν τεχνολογίες οι οποίες είναι κατανοητές, διαδεδομένες και ώριμες.

2.2 Τεχνολογίες Back-end

Η σημαντικότερη απόφαση για το συγκεκριμένο project είναι η τεχνολογία που θα χρησιμοποιηθεί για την υλοποίηση του back-end. Ο λόγος είναι πως το back-end είναι το συστατικό που επικοινωνεί με τη βάση και εκείνο που επιστρέφει τα δεδομένα με μία συγκεκριμένη μορφή ώστε να χρησιμοποιηθούν.
Υπάρχουν πολλές γλώσσες προγραμματισμού ικανές να φέρουν εις πέρας τις απαιτήσεις της εφαρμογής. Επιλέγοντας τη γλώσσα που θα χρησιμοποιήσουμε, θα πρέπει να λάβουμε υπόψιν μας τα παρακάτω:

- χαρακτηριστικά της γλώσσας
- τοπικό οικοσύστημα της γλώσσας
- πρόβλημα που καλούμαστε να επιλύσουμε
- υπάρχουσα γνώση της εκάστοτε γλώσσας
- διαθέσιμους πόρους

Βάσει των παραπάνω, επιλέχθηκε να γίνει χρήση της γλώσσας PHP (έκδοση 7.3) καθώς και ενός από τα πιο διαδεδομένα framework αυτής, του Laravel (έκδοση 6.x). Επιλέχθηκε η έκδοση 6.x, καθώς αποτελεί έκδοση Εκτεταμένης Υποστήριξης (LTS – Long Term Support), οπότε θα λαμβάνει ενημερώσεις ασφαλείας για μεγαλύτερο χρονικό διάστημα.

2.2.1 PHP

Εικόνα 1 Λογότυπο PHP

Η PHP είναι μία open source γλώσσα γενικού σκοπού και αποτελεί μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού διαδικτυακών εφαρμογών. Έχει ένα τεράστιο οικοσύστημα από προγραμματιστές, βιβλιοθήκες και frameworks. Ένα μεγάλο μέρος του παγκοσμίου ιστού είναι γραμμένο σε PHP.

Μεγάλες εταιρείες όπως το Facebook, η Wikipedia, το Yahoo και το Slack χρησιμοποιούν την PHP ως κύρια γλώσσα προγραμματισμού στις υπηρεσίες τους καταγράφοντας καθημερινά εκατομμύρια επισκέψεις.

Η PHP χρησιμοποιείται κυρίως για server-side εφαρμογές και για command-line scripting. Υποστηρίζει λειτουργικά συστήματα Windows, Linux, Unix καθώς και macOS. Για να τρέξει μία web based εφαρμογή σε PHP χρειάζεται κάποιον web server. Υποστηριζόμενοι web servers είναι οι Apache web server, IIS, Nginx, lighttpd καθώς και διάφοροι άλλοι. Ακόμη, με τη χρήση της PHP, υπάρχει η ελευθερία στον προγραμματιστή αναφορικά με το εάν θα χρησιμοποιήσει δομημένο (procedural), αντικειμενοστραφή (object oriented) προγραμματισμό ή μία μίξη των δύο.

Η PHP δεν έχει αυστηρούς κανόνες ή standards όπως άλλες γλώσσες προγραμματισμού (π.χ. Python), παρά μόνο guidelines, με αποτέλεσμα μεγάλα projects που δε χρησιμοποιούν κάποιο framework να μην έχουν σωστό στήσιμο και να γίνονται δύσκολα στην ανάγνωση και στη συντήρηση. Το συγκεκριμένο πρόβλημα είναι γνωστό και ως Spaghetti Code.

Διάφορα κείμενα και αναλύσεις γράφονται αναφορικά με την ασφάλεια σε PHP projects. Πράγματι, χρησιμοποιώντας την PHP, όπως και κάθε άλλη γλώσσα στο back-end, ένα σύστημα μπορεί να γίνει compromise. Οι κύριοι λόγοι είναι λάθος ρυθμισμένοι servers καθώς και έλλειψη input validation στον κώδικα.

Χαρακτηριστικά PHP:

- Δυναμική
- Πολλές βιβλιοθήκες
- Μεγάλη προγραμματιστική κοινότητα
- Μεγάλη ιστορία
- Έλλειψη αυστηρού ελέγχου στον κώδικα

2.2.2 Laravel

Εικόνα 2 Λογότυπο Laravel

Η Το Laravel είναι ένα open source PHP framework που βασίζεται στο Symfony. Χρησιμοποιεί το μοντέλο αρχιτεκτονικής MVC (model-view-controller) για το οποίο θα γίνει ανάλυση παρακάτω.

Το Symfony είναι ένα σετ από components και βιβλιοθήκες που έχει ως στόχο την επιτάχυνση της δημιουργίας και συντήρησης web εφαρμογών. Προάγει, επίσης, την επαναχρησιμοποίηση κώδικα. Δίνει πολλές ελευθερίες στον προγραμματιστή (οργάνωση των φακέλων και βιβλιοθηκών) και επιτρέπει την πλήρη παραμετροποίηση του project. Η λογική του βασίστηκε στο Spring framework.

Μεγάλες εταιρίες χρησιμοποιούν το Laravel ως την κύρια πλατφόρμα τους (λ.χ. Pfizer, BBC και 9gag).

Μερικά από τα πλεονεκτήματα του Laravel είναι τα εξής:

- Artisan CLI: command line interface που δίνει έλεγχο στον προγραμματιστή κατά τη δημιουργία της εφαρμογής. Ο έλεγχος των μοντέλων, των database migrations, του middleware καθώς και των περισσότερων λειτουργιών της εφαρμογής επιτυγχάνεται μέσω αυτού
- Αυθεντικοποίηση: παρέχεται out of the box αυθεντικοποίηση των χρηστών
- Input validation: πλήρης έλεγχος του συνόλου του input που στέλνει ο χρήστης στην εφαρμογή
- Ασφάλεια: μέσω των middleware μπορούν να ελέγχουν διάφορα στοιχεία του request
- CSRF tokens: παρέχεται out of the box csrf protection
- Προστατευόμενα routes: μέσω του ισχυρού μηχανισμού routing μπορούν να υπάρχουν public ή private routes αναλόγως με την κρίση του προγραμματιστή
- Database migrations: όλος ο έλεγχος των πινάκων της βάσης γίνεται μέσω migrations τα οποία γράφονται ως κώδικας PHP
- Αυτόματη εύρεση πακέτων: καθώς, όπως αναφέραμε παραπάνω, χρησιμοποιείται το Symfony, το framework μπορεί αυτόματα να βρίσκει τα πακέτα βάσει δηλώσεων του προγραμματιστή
- Eloquent ORM (Object Relational Mapping): το Eloquent αποτελεί τον μηχανισμό που αντιστοιχεί τα μοντέλα της εφαρμογής με πίνακες στη βάση
- API resources: out of the box υποστήριξη για API

Ακόμη, πολλά CMS (Content Management Systems – Συστήματα Διαχείρισης Περιεχομένου) είναι γραμμένα σε Laravel, όπως για παράδειγμα το October CMS.

Τέλος, αξίζει να σημειωθεί πως το Laravel παρέχει πολλές ακόμη ευκολίες για τον προγραμματιστή (error handling, blade templates, notifications, pagination κ.α.) οι οποίες θα αναλυθούν εκτενώς σε αργότερα κομμάτια της παρούσης εργασίας στα σημεία που χρησιμοποιούνται.

2.2.3 Composer

Εικόνα 3 Λογότυπο Composer

Το composer είναι ένα εργαλείο διαχείρισης εξαρτήσεων (dependency management) για την PHP. Επιτρέπει στον προγραμματιστή να ορίσει βιβλιοθήκες τις οποίες χρησιμοποιεί και να τις διαχειρίζεται (εγκατάσταση/ενημέρωση). Οι εξαρτήσεις είναι διαχειρίσιμες σε επίπεδο project και αποθηκεύονται σε έναν φάκελο (συνήθως vendor) στο καθένα από αυτά. Έχει παρόμοια φιλοσοφία με το npm το οποίο θα αναλυθεί παρακάτω. Επιτρέπει, επίσης, τη δήλωση συγκεκριμένων εκδόσεων βιβλιοθηκών και όχι απαραίτητα την τελευταία έκδοση για αποφυγή ασυμβατότητας με το εκάστοτε project. Για την εγκατάστασή του απαιτείται μόνο ένα εκτελέσιμο PHP, κάτι που επιτρέπει την εγκατάστασή του σε όσα λειτουργικά συστήματα υποστηρίζουν PHP. Η διαχείριση των εξαρτήσεων γίνεται μέσω ενός αρχείου με όνομα composer.json το οποίο περιέχει τις βιβλιοθήκες με τις εκδόσεις τους.

Ο συνήθης τρόπος εγκατάστασης μίας εξάρτησης είναι ο ακόλουθος:

```
composer require monolog/monolog
```

που έχει ως αποτέλεσμα το composer.json να αποκτήσει την παρακάτω μορφή:

```
{
    "require": {
        "monolog/monolog": "1.2.*"
    }
}
```

Εντολές

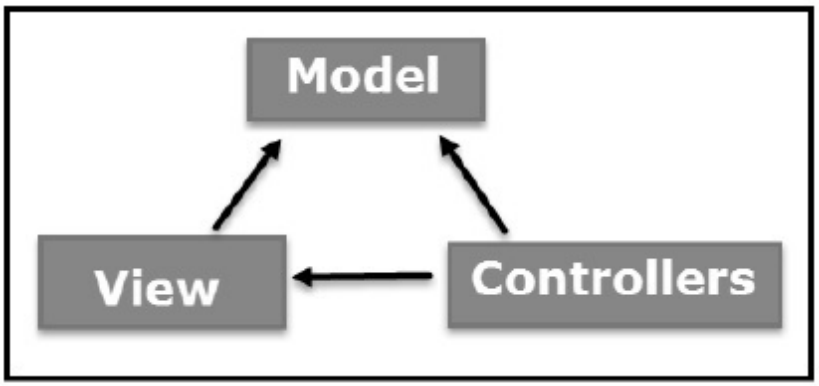
Οι παράμετροι/εντολές που υποστηρίζει το composer είναι οι εξής:

- require: προσθέτει τη βιβλιοθήκη στο composer.json και την εγκαθιστά
- install: εγκαθιστά όλες τις βιβλιοθήκες από ένα ήδη υπάρχον composer.json
- update: αναβαθμίζει όλες τις βιβλιοθήκες σύμφωνα με το composer.json και τους εκάστοτε περιορισμούς
- remove: απεγκαθιστά μία βιβλιοθήκη και την αφαιρεί από το composer.json

Υποστηριζόμενα frameworks

- Symfony
- Laravel
- CodeIgniter
- CakePHP
- Magento
- Yii
- Zend Framework κ.ά.

2.2.4 MVC



Εικόνα 4 Τα μέρη του MVC

Ο Το MVC (model-view-controller) αποτελεί ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για τη δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Το συγκεκριμένο μοντέλο

διαιρεί τη λογική του προγράμματος σε τρία αλληλένδετα μέρη. Αυτό γίνεται για να διαχωριστούν οι εσωτερικές αναπαραστάσεις των πληροφοριών από τον τρόπο με τον οποίο οι πληροφορίες παρουσιάζονται και γίνονται αποδεκτές από τον τελικό χρήστη. Πολλές γλώσσες προγραμματισμού (μεταξύ τους η JavaScript, η Python, η Ruby, η PHP, η Java και η C#) υποστηρίζουν frameworks που υλοποιούν το MVC.

Συστατικά

Model: Το κεντρικό συστατικό του μοντέλου, αποτελεί τη δυναμική δομή δεδομένων της εφαρμογής, ανεξάρτητη από τη διεπαφή χρήστη. Ελέγχει τα δεδομένα, τη λογική και τους κανόνες του προγράμματος.

View: Αναφέρεται στο οπτικό κομμάτι του μοντέλου και επιτρέπει πολλαπλές και διαφορετικές απεικονίσεις των ίδιων πληροφοριών.

Controller: Δέχεται είσοδο από τον χρήστη και τη μετατρέπει σε εντολές για το model ή το view.

Εκτός από τη διαίρεση της εφαρμογής σε αυτά τα συστατικά, ο σχεδιασμός του μοντέλου MVC καθορίζει τις αλληλεπιδράσεις μεταξύ τους. Συγκεκριμένα:

- Το model είναι υπεύθυνο για τη διαχείριση των δεδομένων της εφαρμογής. Λαμβάνει είσοδο χρήστη από τον controller.
- Το view σημαίνει παρουσίαση του μοντέλου σε συγκεκριμένη μορφή.
- Ο controller αποκρίνεται στην από τον χρήστη και εκτελεί αλληλεπιδράσεις στα αντικείμενα του μοντέλου δεδομένων. Ο controller λαμβάνει την είσοδο, την επικυρώνει προαιρετικά και μετά μεταβιβάζει την είσοδο στο model.

Στόχοι

Ταυτόχρονη ανάπτυξη: Καθώς το MVC αποσυνδέει τα διάφορα στοιχεία μιας εφαρμογής, οι προγραμματιστές μπορούν να εργάζονται παράλληλα σε διαφορετικά στοιχεία χωρίς να επηρεάζουν ή να αποκλείουν το ένα το άλλο.

Επαναχρησιμοποίηση κώδικα: Το ίδιο (ή παρόμοιο) view για μία εφαρμογή μπορεί να αναδιαμορφωθεί για άλλη εφαρμογή με διαφορετικά δεδομένα, επειδή το view απλώς χειρίζεται τον τρόπο με τον οποίο τα δεδομένα εμφανίζονται στον χρήστη.

2.2.5 Laravel και MVC

Το Laravel framework υλοποιεί το μοντέλο MVC, όπως αναφέραμε και παραπάνω. Στις επόμενες παραγράφους θα προσπαθήσουμε να εξηγήσουμε τον τρόπο με ένα παράδειγμα από το documentation του framework.

Στο Laravel, το model αναπαρίσταται ως μία PHP class η οποία έχει properties που αντιστοιχίζονται με τα ονόματα των columns στη βάση. Για να δημιουργήσουμε ένα νέο model, χρειάζεται να τρέξουμε την εντολή:

```
php artisan make:model Photo
```

Μετά την εκτέλεση αυτής της εντολής, θα δημιουργηθεί στον φάκελο app μία PHP class με όνομα Photo και αυτή θα είναι η κλάση μέσω της οποίας θα διαχειριζόμαστε το model στη βάση δεδομένων.

Προχωρώντας στο controller κομμάτι, αξίζει να σημειώσουμε το feature Resource Controller που δίνει το Laravel. Σύμφωνα με αυτό, κατά τη δημιουργία ενός Resource Controller, δημιουργούνται οι τυπικές διαδρομές CRUD (Create, Read, Update και Delete) σε έναν controller με μία μόνο εντολή. Για παράδειγμα, για τη δημιουργία του controller που χειρίζεται όλα τα HTTP αιτήματα για το παραπάνω μοντέλο, θα χρειαστεί να τρέξουμε την παρακάτω εντολή:

```
php artisan make:controller PhotoController –resource
```

Με αυτήν την εντολή δημιουργείται ένα αρχείο στον φάκελο app/Http/Controllers με όνομα PhotoController.php. Ο controller θα περιέχει μια μέθοδο για καθεμία από τις διαθέσιμες λειτουργίες πόρων CRUD που αναφέρθηκαν παραπάνω.
Τέλος, για να μπορέσει το Laravel να αναγνωρίσει τον Resource Controller, θα πρέπει να προστεθεί μία γραμμή αντίστοιχη με την παρακάτω στο αρχείο routes/web.php, το οποίο είναι το αρχείο υπεύθυνο για το web routing της εφαρμογής:

```
Route::resource('/photos', 'PhotoController');
```

Η αντιστοίχιση των μεθόδων είναι η ακόλουθη:

Πίνακας 1 Ενέργειες του Resource Controller

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

Εναλλακτικά, και όπως συμβαίνει στην παρούσα εφαρμογή/υλοποίηση, μπορούμε να έχουμε μία γραμμή route για καθεμία από τις λειτουργίες του controller. Εκτενέστερη ανάλυση αναφορικά με τις μεθόδους των controller καθώς και για το routing θα γίνει σε επόμενο κεφάλαιο.
Τέλος, για το view συστατικό του MVC, ας πάρουμε το παράδειγμα της μεθόδου create (C στο CRUD):

```
public function create()
    {
        return view('createphoto');
    }
```

Η συγκεκριμένη μέθοδος καλείται να φορτώσει το view με όνομα createphoto.blade.php από τον φάκελο resources/views. Όπως αναφέρθηκε και παραπάνω, το Laravel χρησιμοποιεί blade templates για το view κομμάτι της εφαρμογής. Τα blade templates είναι μία μίξη HTML και PHP και γίνονται render από τον web server ώστε να εμφανίζεται στον χρήστη το τελικό αποτέλεσμα. Για τη διευκόλυνση του έργου του προγραμματιστή, η σελίδα μπορεί να χωριστεί σε τμήματα (με τα annotation @section - @endsection). Υποστηρίζονται, επίσης, template inheritance, switch/case statements, loops, καθώς και ο ορισμός επαναχρησιμοποιήσιμων components.

2.2.6 API

Ο Το API είναι η συντομογραφία που σχηματίζεται από το Application Programming Interface (Διασύνδεση Προγραμματισμού Εφαρμογών). Το API είναι ένα σύνολο ορισμών και πρωτοκόλλων για τη δημιουργία και την ενσωμάτωση λογισμικού εφαρμογών.

Τα API επιτρέπουν σε μία υπηρεσία να επικοινωνεί με άλλες υπηρεσίες χωρίς να χρειάζεται να γνωρίζει πώς υλοποιούνται. Αυτό απλοποιεί την ανάπτυξη εφαρμογών, εξοικονομώντας χρόνο και χρήμα.

Μερικές φορές τα API θεωρούνται συμβόλαια, με τεκμηρίωση που αντιπροσωπεύει μια συμφωνία μεταξύ των μερών: Εάν το συμβαλλόμενο μέρος 1 στέλνει ένα απομακρυσμένο αίτημα δομημένο με έναν συγκεκριμένο τρόπο, το λογισμικό του συμβαλλόμενου μέρους 2 θα απαντήσει με έναν συγκεκριμένο τρόπο.

Πολύ γνωστά APIs αποτελούν μεταξύ άλλων αυτά του Google Maps, του YouTube και του Twitter.

Εικόνα 5 Αλληλεπίδραση στο API

Τύποι API ανάλογα με την προσβασιμότητα

- Ανοιχτά API (Open APIs): Γνωστά και ως Δημόσια (Public) APIs, δεν υπάρχουν περιορισμοί πρόσβασης σε αυτούς τους τύπους API, επειδή είναι διαθέσιμα στο κοινό.
- API συνεργατών (Partner APIs): Οι προγραμματιστές χρειάζονται συγκεκριμένα δικαιώματα ή άδειες για να έχει πρόσβαση σε αυτόν τον τύπο API, επειδή δεν είναι διαθέσιμα στο κοινό.
- Εσωτερικά API (Internal APIs): Γνωστά και ως Ιδιωτικά (Private) APIs, μόνο τα εσωτερικά συστήματα διαθέτουν αυτόν τον τύπο API. Είναι συνήθως σχεδιασμένα για εσωτερική χρήση σε έναν οργανισμό και χρησιμοποιούνται μεταξύ των διαφόρων εσωτερικών ομάδων για να βελτιώσουν τις υπηρεσίες τους.
- Σύνθετα API (Composite APIs): Αυτός ο τύπος API συνδυάζει διαφορετικά API δεδομένων και υπηρεσιών. Είναι μια ακολουθία εργασιών που εκτελούνται συγχρόνως ως αποτέλεσμα της εκτέλεσης και όχι κατόπιν αιτήματος μιας εργασίας. Οι κύριες χρήσεις του είναι η επιτάχυνση της διαδικασίας εκτέλεσης και η βελτίωση της απόδοσης των ακροατών στις διεπαφές Ιστού.

Υπηρεσίες Ιστού (Web Services)

Οι υπηρεσίες ιστού είναι ένα τυποποιημένο μέσο επικοινωνίας μεταξύ των εφαρμογών πελάτη και διακομιστή. Αποτελούν μία ενότητα λογισμικού που έχει σχεδιαστεί για να εκτελεί ένα συγκεκριμένο σύνολο εργασιών. Οι υπηρεσίες ιστού μπορούν να αναζητηθούν μέσω του δικτύου και να χρησιμοποιηθούν ανάλογα. Όταν καλείται, η διαδικτυακή υπηρεσία παρέχει τη λειτουργικότητα στον πελάτη, ο οποίος θεωρείται πως καλεί αυτήν την υπηρεσία ιστού.

Τύποι Υπηρεσιών Ιστού

Οι δύο κύριοι τύποι υπηρεσιών ιστού είναι τα SOAP και REST.

- SOAP (Simple Object Access Protocol): Το SOAP είναι ένα πρωτόκολλο που σχεδιάστηκε πριν από το REST. Η κύρια ιδέα πίσω από το σχεδιασμό του ήταν να διασφαλιστεί ότι τα προγράμματα που βασίζονται σε διαφορετικές πλατφόρμες και γλώσσες προγραμματισμού θα μπορούσαν να ανταλλάσσουν δεδομένα με ευκολία.
- REST (Representational State Transfer): Το REST σχεδιάστηκε ειδικά για εργασία με πολυμεσικά στοιχεία, αρχεία ή ακόμη και αντικείμενα σε ένα συγκεκριμένο υλικό. Οποιαδήποτε υπηρεσία ιστού που ορίζεται στις αρχές του REST μπορεί να ονομαστεί υπηρεσία ιστού RestFul. Μια υπηρεσία REST χρησιμοποιεί τα ρήματα HTTP των GET, POST, PUT και DELETE για εργασία με τα απαιτούμενα στοιχεία.

Πίνακας 2 Σύγκριση SOAP και REST

SOAP	**REST**
Το SOAP είναι πρωτόκολλο σχεδιασμένο με προδιαγραφές. Περιλαμβάνει ένα αρχείο WSDL που έχει τις απαιτούμενες πληροφορίες σχετικά με τον σκοπό της υπηρεσίας, καθώς και την τοποθεσία της.	Το REST είναι ένα αρχιτεκτονικό στυλ το οποίο χαρακτηρίζεται από τα ακόλουθα: □ είναι client-server (πελάτη-διακομιστή) □ είναι stateless □ γίνεται χρήση προσωρινής μνήμης □ διαθέτει ομοιόμορφη διεπαφή □ αποτελεί πολυεπίπεδο σύστημα
Το SOAP δεν μπορεί να κάνει χρήση του REST.	Το REST μπορεί να χρησιμοποιήσει το SOAP ως βασικό πρωτόκολλο υπηρεσιών.

SOAP	**REST**
Το SOAP χρησιμοποιεί διεπαφές (interfaces) υπηρεσίας για να εκθέσει τη λειτουργικότητά του σε εφαρμογές πελατών. Το αρχείο WSDL παρέχει στον πελάτη τις απαραίτητες πληροφορίες για τις προσφερόμενες υπηρεσίες.	Το REST χρησιμοποιεί τα URLs ως πόρους για την αναζήτηση των στοιχείων του.
Το SOAP απαιτεί περισσότερο εύρος ζώνης για τη χρήση του. Δεδομένου ότι τα μηνύματα SOAP περιέχουν πολλές πληροφορίες, ο όγκος δεδομένων είναι μεγάλος.	Το REST δε χρειάζεται μεγάλο εύρος ζώνης για την αποστολή μηνυμάτων στον διακομιστή. Τα μηνύματα REST αποτελούνται κυρίως από μηνύματα JSON.
Το SOAP μπορεί να λειτουργήσει μόνο με μορφή XML.	Το REST επιτρέπει διαφορετική μορφή δεδομένων, όπως απλό κείμενο, HTML, XML, JSON κ.λπ.

Καθώς η παρούσα εφαρμογή κάνει χρήση του REST, θα προχωρήσουμε σε περαιτέρω ανάλυση της αρχιτεκτονικής του.

Χαρακτηριστικά του REST

- Client-Server (Πελάτης-Διακομιστής): Διαχωρίζοντας τους προβληματισμούς της διεπαφής χρήστη από τους προβληματισμούς για την αποθήκευση δεδομένων, βελτιώνουμε τη φορητότητα της διεπαφής χρήστη σε πολλές πλατφόρμες και βελτιώνουμε την επεκτασιμότητα απλοποιώντας τα στοιχεία του διακομιστή.
- Stateless: Κάθε αίτημα από πελάτη σε διακομιστή πρέπει να περιέχει όλες τις απαραίτητες πληροφορίες για την κατανόηση του αιτήματος από τον διακομιστή και δεν μπορεί να κάνει χρήση από οτιδήποτε αποθηκευμένο σε αυτόν. Συνεπώς, η κατάσταση συνεδρίας (session state) διατηρείται αποκλειστικά στον πελάτη.
- Cacheable (Χρήση προσωρινής μνήμης): Οι περιορισμοί της προσωρινής μνήμης απαιτούν τα δεδομένα εντός μιας απόκρισης σε ένα αίτημα να επισημαίνονται έμμεσα ή ρητά ως προσωρινά αποθηκευόμενα ή μη προσωρινά αποθηκευόμενα. Εάν μια απόκριση είναι προσωρινά αποθηκευμένη, τότε η κρυφή μνήμη στον πελάτη έχει το δικαίωμα να επαναχρησιμοποιήσει αυτά τα δεδομένα για μεταγενέστερα, πανομοιότυπα αιτήματα.
- Uniform interface (Ομοιόμορφη διεπαφή): Εφαρμόζοντας την αρχή της γενικότητας του λογισμικού στη διεπαφή των στοιχείων, απλοποιείται η συνολική αρχιτεκτονική του συστήματος και βελτιώνεται η ορατότητα των αλληλεπιδράσεων. Για να επιτευχθεί μια ομοιόμορφη διεπαφή, απαιτούνται πολλοί αρχιτεκτονικοί περιορισμοί για την καθοδήγηση της συμπεριφοράς των συστατικών. Το REST ορίζεται από τέσσερις περιορισμούς διεπαφής: αναγνώριση πόρων, χειρισμός πόρων μέσω παραστάσεων, αυτο-περιγραφικά μηνύματα και υπερμέσα ως η μηχανή της κατάστασης εφαρμογής.
- Layered system (Πολυεπίπεδο σύστημα): Το πολυεπίπεδο σύστημα επιτρέπει σε μια αρχιτεκτονική να αποτελείται από ιεραρχικά στρώματα, περιορίζοντας τη συμπεριφορά των συστατικών έτσι ώστε το καθένα να μην μπορεί να «δει» πέρα από το άμεσο επίπεδο με το οποίο αλληλοεπιδρά.

HTTP και REST

Παραπάνω αναφέρθηκε πως το REST κάνει χρήση των HTTP ρημάτων για αλληλεπίδραση. Οι ενέργειες που πραγματοποιεί κάθε ρήμα ορίζονται σαφώς από το RFC του HTTP (7231) και είναι οι παρακάτω:

- GET: Η μέθοδος GET ζητά τη μεταφορά μιας τρέχουσας επιλεγμένης αναπαράστασης για τον ζητούμενο πόρο. Το GET είναι ο κύριος μηχανισμός ανάκτησης πληροφοριών και το επίκεντρο σχεδόν όλων των βελτιστοποιήσεων απόδοσης.
- PUT: Η μέθοδος PUT ζητά να δημιουργηθεί ή να αντικατασταθεί η κατάσταση του ζητούμενου πόρου με την κατάσταση που ορίζεται από την αναπαράσταση που περιλαμβάνεται στο ωφέλιμο φορτίο μηνύματος αποστολής.
- POST: Η μέθοδος POST ζητά από τον ζητούμενο πόρο να επεξεργαστεί την αναπαράσταση που περικλείεται στο αίτημα σύμφωνα με τη συγκεκριμένη σημασιολογία του πόρου.
- DELETE: Η μέθοδος DELETE ζητά από τον διακομιστή να καταργήσει τη συσχέτιση μεταξύ του πόρου προορισμού και της τρέχουσας λειτουργικότητάς του.
- PATCH: Η μέθοδος PATCH έχει παρόμοια λειτουργία με τη μέθοδο PUT. Η διαφορά μεταξύ τους αντικατοπτρίζεται στον τρόπο με τον οποίο ο διακομιστής επεξεργάζεται το εκάστοτε αίτημα για την τροποποίηση ενός πόρου. Σε ένα αίτημα PUT, η οντότητα θεωρείται τροποποιημένη έκδοση του πόρου που είναι αποθηκευμένος στον διακομιστή και ο πελάτης ζητά να αντικατασταθεί η αποθηκευμένη έκδοση. Με τη μέθοδο PATCH, ωστόσο, η οντότητα περιέχει ένα σύνολο οδηγιών που περιγράφουν πώς ένας πόρος που βρίσκεται επί του παρόντος στον διακομιστή πρέπει να τροποποιηθεί για να παράγει μια νέα έκδοση. Η συγκεκριμένη μέθοδος ορίζεται, μάλιστα, σε ξεχωριστό RFC, το υπ' αριθμόν 5789.

Ακόμη, σε αυτό το σημείο, αξίζει να αναφερθεί πως το REST χρησιμοποιεί και τους κωδικούς κατάστασης του HTTP ώστε να ενημερώσει τον παραλήπτη για την κατάσταση του αιτήματός του. Οι κωδικοί κατάστασης συνοψίζονται στις παρακάτω πέντε μεγάλες ομάδες:

- 1xx Informational (Πληροφορίες): Μεταφέρει πληροφορίες σε επίπεδο πρωτοκόλλου μεταφοράς.
- 2xx Success (Επιτυχία): Υποδεικνύει ότι το αίτημα του πελάτη έγινε αποδεκτό με επιτυχία.
- 3xx Redirection (Ανακατεύθυνση): Δηλώνει ότι ο πελάτης πρέπει να προβεί σε κάποια επιπλέον ενέργεια για να ολοκληρώσει το αίτημά του.
- 4xx Client Error (Σφάλμα πελάτη): Υποδεικνύει σφάλμα στην πλευρά του πελάτη.
- 5xx Server Error (Σφάλμα διακομιστή): Υποδεικνύει σφάλμα στην πλευρά του διακομιστή.

Όλοι οι κωδικοί κατάστασης ορίζονται και αυτοί από το RFC 7231.

Laravel και API

Όπως αναφέρθηκε παραπάνω, το Laravel παρέχει εγγενώς υποστήριξη για τη δημιουργία API και θεωρείται μία από τις καλύτερες διαθέσιμες επιλογές για τη δημιουργία API στην PHP.

Ακολουθεί τη λογική που περιεγράφηκε παραπάνω στο κεφάλαιο του MVC, μόνο που αυτή τη φορά οι διαδρομές (routes) γράφονται στο αρχείο routes/api.php. Έτσι, το πλαίσιο γνωρίζει πως τα συγκεκριμένα resources θα είναι προσβάσιμα μέσω διαδρομών της μορφής http(s)://domain.com/api/path/to/resource. Προσθήκη αποτελεί το /api/ μετά το domain name. Φυσικά αυτό είναι configurable στις ρυθμίσεις του πλαισίου και είναι στην κρίση του προγραμματιστή εάν θα το χρησιμοποιήσει ως έχει ή εάν θα το αλλάξει.

Για το API χρησιμοποιούνται αποκλειστικά οι διαδρομές για CRUD χωρίς να γίνεται υλοποίηση κάποιας διαδρομής για εμφάνιση γραφικών.

2.2.7 JSON

Το JSON (JavaScript Object Notation) είναι ένα lightweight πρότυπο μεταφοράς δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και να το γράφουν, καθώς και για τα μηχανήματα να το αναλύσουν και να το δημιουργήσουν. Το JSON είναι μια μορφή κειμένου που είναι εντελώς ανεξάρτητη

από τη γλώσσα, αλλά χρησιμοποιεί συμβάσεις που είναι πανομοιότυπες σε όλες τις γλώσσες προγραμματισμού (π.χ. PHP, Java, Python κ.λπ).

Το JSON έχει δύο τύπους απεικόνισης:

- ζευγάρια ονόματος/τιμής: σε διαφορετικές γλώσσες αυτός ο τύπος μεταφράζεται ως object, record, struct, dictionary, hash table, keyed list ή associative array
- ταξινομημένη λίστα τιμών: σε διαφορετικές γλώσσες αυτός ο τύπος μεταφράζεται ως array, vector, list ή sequence

Σχεδόν όλες οι σύγχρονες γλώσσες προγραμματισμού υποστηρίζουν τη μία ή την άλλη μορφή.

Παράδειγμα JSON

```
{
   "people": [
      {
         "firstName":"Nikolaos Christos",
         "lastName":"Nikolaidis"
      },
      {
         "firstName":"Antonis",
         "lastName":"Sidiropoulos"
      }
   ]
}
```

JSON και API

Η ολοένα αυξανόμενη χρησιμοποίηση του JSON στα APIs οδήγησε στη δημιουργία ενός specification αναφορικά με τη χρήση του. Το εν λόγω specification ονομάζεται JSON:API και καθορίζει τον τρόπο με τον οποίο ένας πελάτης (client) πρέπει να ζητά την ανάκτηση ή την τροποποίηση πόρων και τον τρόπο με τον οποίο ένας διακομιστής (server) πρέπει να ανταποκρίνεται σε αυτά τα αιτήματα.

Έχει σχεδιαστεί για να ελαχιστοποιεί τόσο τον αριθμό των αιτημάτων όσο και τον όγκο των δεδομένων που μεταδίδονται μεταξύ πελατών και διακομιστών. Αυτή η αποτελεσματικότητα επιτυγχάνεται χωρίς να διακυβεύεται η αναγνωσιμότητα, η ευελιξία ή η δυνατότητα εντοπισμού.

Τόσο οι πελάτες όσο και διακομιστές υποχρεούνται να δηλώσουν ρητά πως επιθυμούν την ανταλλαγή JSON δεδομένων μεταξύ τους μέσω κεφαλίδων (headers) του HTTP. Στην περίπτωση του πελάτη, γίνεται χρήση του Accept:application/json, ενώ σε αυτήν του διακομιστή, Content-Type:application/json.

JSON και Laravel

Καθώς δημιουργούμε ένα API με Laravel, εμφανίζεται η ανάγκη να υπάρχει ένας ενδιάμεσος μεταξύ των models μας και των JSON απαντήσεων που επιστρέφονται από την εφαρμογή μας, όπως συμβαίνει και με την παρούσα εφαρμογή. Για αυτόν τον σκοπό, έγινε η χρήση του feature API Resources. Μέσω αυτού, μπορούμε να ελέγχουμε πλήρως τα επιστρεφόμενα δεδομένα από το model μας, καθώς και να μετασχηματίζουμε τις απαντήσεις μας (για παράδειγμα, να προσθέτουμε διαφορετικές HTTP κεφαλίδες). Ακόμη, μέσω αυτού του μηχανισμού, ένα model μπορεί να περιέχει και πληροφορίες από τις συσχετίσεις του.

Το ισχυρό αυτό feature μας δίνει, επίσης, τη δυνατότητα να δημιουργήσουμε collections από resources. Μέσω της συγκεκριμένης δυνατότητας μπορούμε να συμπεριλάβουμε pagination στις απαντήσεις που επιστρέφει το API. Το pagination επιτυγχάνεται μέσω δύο values στην απάντηση (links και meta) στα οποία περιλαμβάνεται πληροφορία όπως ο αριθμός των διαθέσιμων σελίδων, σε ποια σελίδα βρίσκεται ο χρήστης στην παρούσα απάντηση, πόσες είναι οι διαθέσιμες σελίδες, πόσα model περιέχει το collection κ.α.

Ακόμη, αξίζει να σημειωθεί πως το Laravel δίνει τη δυνατότητα για μετατροπή οποιουδήποτε PHP array σε JSON περιλαμβάνοντας την αντίστοιχη απαιτούμενη HTTP κεφαλίδα, όπως στο παρακάτω παράδειγμα:

```
return response()->json([
    'name' => 'Nikolaos Christos',
    'surname' => 'Nikolaidis',
]);
```

Τέλος, υπάρχει το serialization σε JSON. Στον συγκεκριμένο τρόπο, αφού έχουμε κάνει retrieve τις πληροφορίες ενός ή περισσότερων model, μπορούμε να τις επιστρέψουμε ως JSON, όπως στο παράδειγμα που ακολουθεί:

```
$user = App\User::find(1);

return $user->toJson();

return $user→toJson(JSON_PRETTY_PRINT);
```

Στον παραπάνω τρόπο, υποστηρίζονται και όλα τα options του PHP function json_encode (π.χ. JSON_FORCE_OBJECT, JSON_HEX_QUOT, JSON_HEX_TAG κ.α.).

2.2.8 MariaDB

Εικόνα 6 Λογότυπο MariaDB

Βάση δεδομένων

Μια βάση δεδομένων είναι μια οργανωμένη συλλογή δεδομένων. Μια σχεσιακή βάση δεδομένων είναι μια συλλογή σχημάτων, πινάκων, ερωτημάτων, αναφορών, προβολών και άλλων στοιχείων. Οι σχεδιαστές βάσεων δεδομένων συνήθως οργανώνουν τα δεδομένα για να μοντελοποιούν πτυχές της πραγματικότητας κατά τρόπο που να υποστηρίζει διαδικασίες που απαιτούν πληροφορίες, όπως (για παράδειγμα) μοντελοποίηση της διαθεσιμότητας δωματίων σε ξενοδοχεία με τρόπο που να υποστηρίζει την εύρεση ενός ξενοδοχείου με κενές θέσεις.

Ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS) είναι μια εφαρμογή υπολογιστή-λογισμικού που αλληλοεπιδρά με τους τελικούς χρήστες, άλλες εφαρμογές και την ίδια τη βάση δεδομένων για την καταγραφή και ανάλυση δεδομένων. Ένα DBMS γενικού σκοπού επιτρέπει τον ορισμό, τη δημιουργία, την αναζήτηση, την ενημέρωση και τη διαχείριση των βάσεων δεδομένων.

Η MariaDB είναι μια βάση δεδομένων SQL κατάλληλη για μικρές έως μεσαίες ιστοσελίδες. Ξεκίνησε ως fork της MySQL από τον ιδιοκτήτη της όταν αυτή πουλήθηκε στην Oracle. Η MariaDB έχει διατηρήσει υψηλή συμβατότητα με τη MySQL κι, ως εκ τούτου, μπορεί να χρησιμοποιηθεί ως drop-in replacement αυτής. Οι κοινές εφαρμογές της περιλαμβάνουν εφαρμογές web που βασίζονται σε διάφορες γλώσσες προγραμματισμού που απαιτούν back-end αποθήκευσης DB.

Η MariaDB χρησιμοποιείται συνήθως με 2 διαφορετικούς μηχανισμούς αποθήκευσης. Ο ένας ονομάζεται MyISAM, δεν υποστηρίζει συναλλαγές και αποθηκεύει κάθε πίνακα σε ένα σύνολο 3 αρχείων. Ο δεύτερος λέγεται InnoDB και υποστηρίζει συναλλαγές. Αυτός ο μηχανισμός αποθήκευσης αποθηκεύει όλα τα δεδομένα σε ένα ενιαίο σύνολο από bytes ή χρησιμοποιεί ένα σύνολο bytes ανά κατάλογο βάσεων δεδομένων. Η MariaDB έχει ένα μεγάλο πλεονέκτημα, καθώς είναι δωρεάν, είναι συνήθως διαθέσιμη σε πακέτα κοινής φιλοξενίας (shared hosting) και μπορεί εύκολα να εγκατασταθεί σε περιβάλλον Linux, Unix ή Windows.

Laravel και Database

Το Laravel δίνει τρεις επιλογές για αλληλεπίδραση με τη βάση: raw SQL queries, το query builder και το Eloquent ORM.

Raw SQL queries μπορούν να τρέξουν με τον παρακάτω τρόπο:

```
$users = DB::select( DB::raw("SELECT * FROM users ") );
```

Το query builder συνδέει πίνακες της βάσης με functions που αντιστοιχούν σε database queries. Για παράδειγμα:

```
$users = DB::table('users')->where('age','>=','20')→get();
```

Η παραπάνω γραμμή θα έχει ως αποτέλεσμα στη μεταβλητή $users να επιστραφεί το σύνολο των στοιχείων του πίνακα users όπου η ηλικία είναι μεγαλύτερη ή ίση με 20. Αντίστοιχα, υπάρχουν και άλλα functions (π.χ. count(), max() κλπ) τα οποία εκτελούν τις αντίστοιχες λειτουργίες.

Το Eloquent ORM, από την άλλη, καθώς ήδη υπάρχει η αντιστοιχία πίνακα και model, κάνει χρήση αυτής ώστε να λάβει τα αποτέλεσμα από τη βάση. Για παράδειγμα:

```
$users = App\Users::all();
```

Η παραπάνω γραμμή θα κάνει χρήση του model Flight και θα επιστρέψει όλα τα records της βάσης στη μεταβλητή $users. Παρόμοια με το query builder παραπάνω, προσφέρει διάφορα functions που εκτελούν τις αντίστοιχες λειτουργίες.

Το Laravel υποστηρίζει τους παρακάτω τύπους βάσεων:

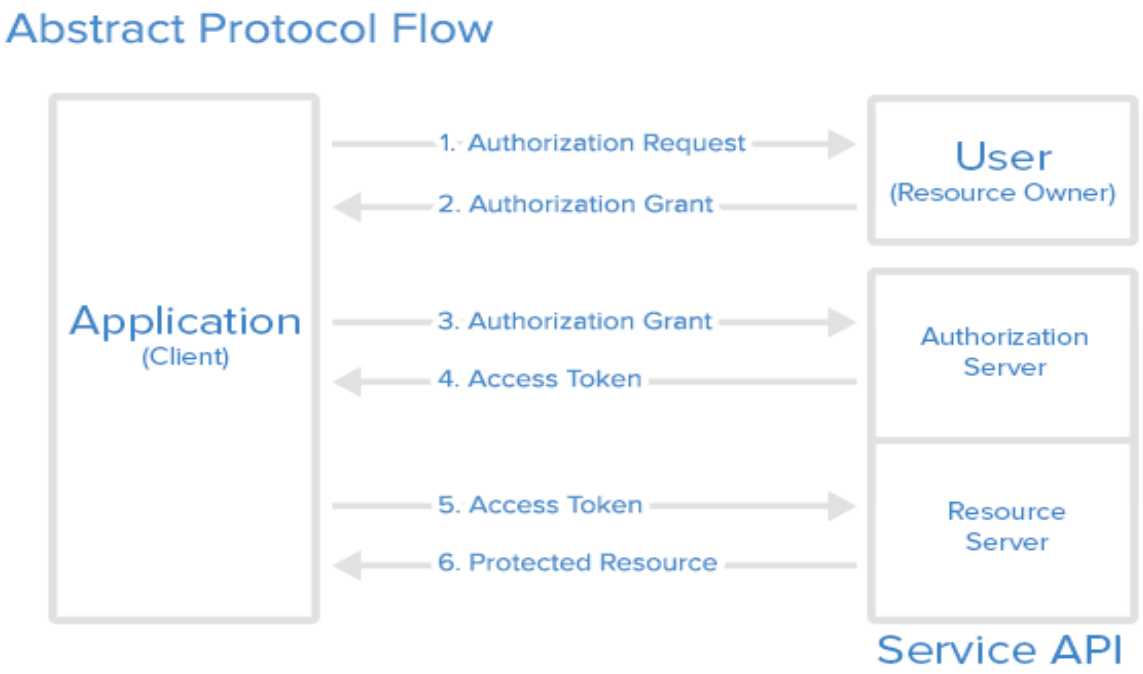
- MySQL 5.6+
- PostgreSQL 9.4+
- SQLite 3.8.8+
- SQL Server 2017+

Αξίζει να σημειωθεί πως μέσα στα υποστηριζόμενα configurations, δίνεται και η δυνατότητα για διαφορετικά connections με διαφορετικές βάσεις δεδομένων τόσο αναφορικά με τον τύπο (π.χ. MariaDB και MySQL), αλλά και απλώς κάνοντας χρήση άλλων διαπιστευτηρίων στο connection. Δίνεται, ακόμη, η δυνατότητα πλήρους διαχείρισης των transactions αλλά και των deadlocks.

2.2.9 OAuth 2

Εικόνα 7 Λογότυπο OAuth

Το OAuth 2 είναι ένα πλαίσιο εξουσιοδότησης (authorization framework) που επιτρέπει στις εφαρμογές να αποκτούν περιορισμένη πρόσβαση σε λογαριασμούς χρηστών σε μια υπηρεσία HTTP, όπως το Facebook. Λειτουργεί μεταβιβάζοντας τον έλεγχο ταυτότητας χρήστη στην υπηρεσία που φιλοξενεί τον λογαριασμό χρήστη και εξουσιοδοτώντας εφαρμογές τρίτων να έχουν πρόσβαση στον λογαριασμό χρήστη. Το OAuth 2 παρέχει ροές εξουσιοδότησης για εφαρμογές ιστού και επιτραπέζιων υπολογιστών και κινητές συσκευές. Το OAuth 2 περιγράφεται από το RFC 6749.



Εικόνα 8 Ροή του OAuth

Ορολογία

- Διακριτικό πρόσβασης (Access token): Ένα διακριτικό που χρησιμοποιείται για πρόσβαση σε προστατευμένους πόρους.
- Κωδικός εξουσιοδότησης (Authorization code): Ένα διαμεσολαβητικό διακριτικό που δημιουργείται όταν ένας χρήστης εξουσιοδοτεί έναν πελάτη να έχει πρόσβαση σε προστατευμένους πόρους για λογαριασμό του. Ο πελάτης λαμβάνει αυτό το διακριτικό και το ανταλλάσσει με ένα διακριτικό πρόσβασης.
- Διακομιστής εξουσιοδότησης (Authorization server): Ένας διακομιστής που εκδίδει διακριτικά πρόσβασης μετά τον επιτυχημένο έλεγχο ταυτότητας ενός πελάτη και ενός κατόχου πόρου και εξουσιοδοτεί το αίτημα.
- Πελάτης (Client): Μια εφαρμογή που αποκτά πρόσβαση σε προστατευμένους πόρους για λογαριασμό του κατόχου των πόρων (όπως ένας χρήστης). Ο πελάτης θα μπορούσε να φιλοξενείται σε διακομιστή, επιτραπέζιο υπολογιστή, κινητό ή άλλη συσκευή.
- Χορήγηση (Grant): Η χορήγηση είναι μια μέθοδος απόκτησης διακριτικού πρόσβασης.
- Διακομιστής πόρων (Resource server): Ένας διακομιστής που βρίσκεται μπροστά από προστατευμένους πόρους και είναι σε θέση να δέχεται και να ανταποκρίνεται σε αιτήματα για αυτούς χρησιμοποιώντας διακριτικά πρόσβασης.
- Κάτοχος πόρου (Resource owner): Ο χρήστης που εξουσιοδοτεί μια εφαρμογή να έχει πρόσβαση στον λογαριασμό του. Η πρόσβαση της εφαρμογής στον λογαριασμό του χρήστη περιορίζεται στο «εύρος» της εξουσιοδότησης που παρέχεται (π.χ. πρόσβαση ανάγνωσης ή εγγραφής).
- Πεδίο εφαρμογής (Scope): Μια άδεια.

- JWT: Ένα JSON Web Token είναι μια μέθοδος για την ασφαλή εκπροσώπηση αξιώσεων μεταξύ δύο μερών.

Ελλείψει επίσημης μετάφρασης, χρησιμοποιήθηκαν οι όροι που, κατά τον γράφοντα, είναι καταλληλότεροι για τον εκάστοτε αγγλικό όρο.

Πεδία εφαρμογής OAuth (Scopes)

Το πεδίο εφαρμογής είναι ένας μηχανισμός στο OAuth 2 για τον περιορισμό της πρόσβασης μιας εφαρμογής στον λογαριασμό ενός χρήστη. Μια εφαρμογή μπορεί να ζητήσει ένα ή περισσότερα πεδία, αυτές οι πληροφορίες στη συνέχεια παρουσιάζονται στον χρήστη στην οθόνη συναίνεσης και το διακριτικό πρόσβασης που εκδίδεται στην εφαρμογή θα περιορίζεται στα παραχωρούμενα πεδία.

Η προδιαγραφή του OAuth επιτρέπει στον διακομιστή εξουσιοδότησης ή στον χρήστη να τροποποιεί τα πεδία που παρέχονται στην εφαρμογή σε σύγκριση με αυτό που ζητείται αν και, στην πράξη, αυτό δεν υλοποιείται από πολλές υπηρεσίες.

Το OAuth δεν καθορίζει συγκεκριμένες τιμές για τα πεδία, καθώς αυτά εξαρτούνται σε μεγάλο βαθμό από την εσωτερική αρχιτεκτονική και τις ανάγκες της υπηρεσίας.

JSON Web Tokens

Το JSON Web Token (JWT) είναι ένας τρόπος κωδικοποίησης αξιώσεων (claims) σε ένα έγγραφο JSON που στη συνέχεια υπογράφεται. Περιγράφεται από το RFC 7519. Τα JWT μπορούν να χρησιμοποιηθούν ως OAuth 2 Bearer Tokens για την κωδικοποίηση όλων των σχετικών τμημάτων ενός διακριτικού πρόσβασης στο ίδιο το διακριτικό πρόσβασης αντί να χρειάζεται να αποθηκεύονται σε βάση δεδομένων.

Bearer Tokens

Το Bearer Tokens είναι ο πιο ευρέως χρησιμοποιούμενος τύπος διακριτικού πρόσβασης που χρησιμοποιείται με το OAuth 2. Το Bearer Token είναι μια αδιαφανής συμβολοσειρά, που δεν προορίζεται να έχει νόημα για τους χρήστες που τη χρησιμοποιούν. Ορισμένοι διακομιστές εκδίδουν διακριτικά που είναι μια σύντομη σειρά δεκαεξαδικών χαρακτήρων, ενώ άλλοι ενδέχεται να χρησιμοποιούν δομημένα διακριτικά όπως τα JSON Web Tokens που αναφέρθηκαν παραπάνω.

OAuth Grant Types

Το OAuth καθορίζει διάφορους τύπους χορηγήσεων για διαφορετικές περιπτώσεις χρήσης, καθώς και ένα πλαίσιο για τη δημιουργία νέων τύπων χορήγησης.

Οι συχνότερα χρησιμοποιούμενοι τύποι είναι οι ακόλουθοι:

- Κωδικός Εξουσιοδότησης (Authorization Code): Χρησιμοποιείται από εμπιστευτικούς και δημόσιους πελάτες για την ανταλλαγή κωδικού εξουσιοδότησης για διακριτικό πρόσβασης. Αφού ο χρήστης επιστρέψει στον πελάτη μέσω του URL ανακατεύθυνσης, η εφαρμογή θα λάβει τον κωδικό εξουσιοδότησης από τη διεύθυνση URL και θα τον χρησιμοποιήσει για να ζητήσει ένα διακριτικό πρόσβασης.
- Διαπιστευτήρια πελάτη (Client Credentials): Χρησιμοποιείται από τους πελάτες για απόκτηση ενός διακριτικού πρόσβασης εκτός του περιβάλλοντος ενός χρήστη. Χρησιμοποιείται συνήθως από πελάτες για πρόσβαση σε πόρους για τον εαυτό τους και όχι για πρόσβαση σε πόρους ενός χρήστη.
- Κωδικός συσκευής (Device Code): Χρησιμοποιείται από συσκευές χωρίς πρόγραμμα περιήγησης ή με περιορισμούς εισόδου στη ροή της συσκευής για την ανταλλαγή κωδικού συσκευής που αποκτήθηκε προηγουμένως με ένα διακριτικό πρόσβασης.

- Ανανέωση διακριτικού (Refresh Token): Χρησιμοποιείται από τους πελάτες για την ανταλλαγή ενός διακριτικού ανανέωσης για ένα διακριτικό πρόσβασης όταν αυτό έχει λήξει, κάτι που επιτρέπει στους πελάτες να συνεχίσουν να έχουν ένα έγκυρο διακριτικό πρόσβασης χωρίς περαιτέρω αλληλεπίδραση με τον χρήστη.
- Παραχώρηση κωδικού (Password grant): Ο τύπος παραχώρησης κωδικού πρόσβασης είναι ένας τρόπος ανταλλαγής διαπιστευτηρίων χρήστη με διακριτικό πρόσβασης.

Laravel Passport

Τα API χρησιμοποιούν συνήθως διακριτικά για έλεγχο ταυτότητας χρηστών και δεν διατηρούν την κατάσταση συνεδρίας (session state) μεταξύ των αιτημάτων. Γι' αυτόν τον σκοπό, το Laravel χρησιμοποιεί το Laravel Passport, το οποίο προσφέρει μια πλήρη υλοποίηση διακομιστή OAuth 2. Το Passport βασισμένο στο πακέτο «League OAuth2 server». Εγκαθίσταται μέσω του composer με τη διαδικασία που περιεγράφηκε παραπάνω.

Στην παρούσα εφαρμογή γίνεται χρήση του τελευταίου grant type. Έγινε επιλογή αυτού του τρόπου χορήγησης, καθώς η διαδικτυακή εφαρμογή αποτελεί πρώτης τάξης πελάτη (first-party client) του API. Επίσης, υπήρχε η ανάγκη να γίνει σύνδεση με το σύστημα αυθεντικοποίησης του τμήματος. Περαιτέρω πληροφορίες για τον τρόπο αυθεντικοποίησης θα αναφερθούν σε επόμενο κεφάλαιο.

Η χρήση του Laravel Passport στο API της παρούσας εφαρμογής δίνει τη δυνατότητα αργότερα, σε περίπτωση που κριθεί σκόπιμο, να γίνει χρήση άλλου τύπου χορήγηση για τη διαδικτυακή εφαρμογή, καθώς, επίσης, και να γίνει χρήση οποιουδήποτε τύπου χορήγησης για άλλες, τρίτες, εφαρμογές.

2.2.10 LDAP

Το LDAP (Lightweight Directory Access Protocol) είναι ένα ανοιχτό πρωτόκολλο που χρησιμοποιείται για την αποθήκευση και ανάκτηση δεδομένων από μια ιεραρχική δομή καταλόγου. Χρησιμοποιείται συνήθως για την αποθήκευση πληροφοριών σχετικά με έναν οργανισμό, τα στοιχεία και τους χρήστες του.

Μια υπηρεσία καταλόγου χρησιμοποιείται για την αποθήκευση, οργάνωση και παρουσίαση δεδομένων σε μορφή τύπου κλειδιού-τιμής. Συνήθως, οι κατάλογοι βελτιστοποιούνται για αναζητήσεις και λειτουργίες ανάγνωσης σε σχέση με τις λειτουργίες εγγραφής. Έτσι λειτουργούν εξαιρετικά καλά για δεδομένα που χρησιμοποιούνται συχνά αλλά αλλάζουν σπάνια.

Τα δεδομένα που αποθηκεύονται σε μια υπηρεσία καταλόγου είναι συχνά περιγραφικής φύσης και χρησιμοποιούνται για τον καθορισμό των ιδιοτήτων μιας οντότητας. Ένα παράδειγμα φυσικού αντικειμένου που θα μπορούσε να αναπαρασταθεί καλά σε μια υπηρεσία καταλόγου είναι ένα βιβλίο διευθύνσεων. Κάθε άτομο μπορεί να εκπροσωπείται από μια καταχώριση στον κατάλογο, με ζεύγη κλειδιών-τιμών που περιγράφουν τα στοιχεία επικοινωνίας τους, τον τόπο της επιχείρησης κ.λπ.

Το LDAP είναι ένα πρωτόκολλο επικοινωνίας που καθορίζει τις μεθόδους στις οποίες είναι δυνατή η πρόσβαση σε μια υπηρεσία καταλόγου. Διαμορφώνει τον τρόπο με τον οποίο τα δεδομένα μιας υπηρεσίας καταλόγου θα πρέπει να προβάλλονται στους χρήστες, καθορίζει τις απαιτήσεις για τα στοιχεία που χρησιμοποιούνται για τη δημιουργία καταχωρίσεων δεδομένων σε μια υπηρεσία καταλόγου και περιγράφει τον τρόπο με τον οποίο χρησιμοποιούνται διαφορετικά στοιχεία για τη σύνθεση καταχωρίσεων. Είναι ένα ανοιχτό πρωτόκολλο, συνεπώς υπάρχουν πολλές διαφορετικές εφαρμογές διαθέσιμες. Το OpenLDAP είναι μία από τις πιο διαδεδομένες από αυτές.

Βασικά συστατικά

- Attributes: Τα ίδια τα δεδομένα σε ένα σύστημα LDAP αποθηκεύονται κυρίως σε στοιχεία που ονομάζονται χαρακτηριστικά (attributes). Τα χαρακτηριστικά είναι ζεύγη τιμών-κλειδιών. Τα

κλειδιά έχουν προκαθορισμένα ονόματα που υπαγορεύονται από τα objectClasses που έχουν επιλεγεί. Επιπλέον, τα δεδομένα σε ένα χαρακτηριστικό πρέπει να ταιριάζουν με τον τύπο που ορίζεται στον αρχικό ορισμό του χαρακτηριστικού.

- Entries: Μια καταχώριση (entry) είναι μια συλλογή πληροφοριών σχετικά με μια οντότητα.
- Object Classes: Οι κλάσεις αντικειμένων (object classes) είναι στοιχεία που καθορίζουν συλλογές τύπων χαρακτηριστικών που ενδέχεται να σχετίζονται με έναν συγκεκριμένο τύπο αντικειμένου, διαδικασίας ή άλλης οντότητας. Κάθε καταχώρηση έχει μια δοκιμή κλάση αντικειμένου, η οποία υποδεικνύει το είδος του αντικειμένου που αντιπροσωπεύει μια καταχώρηση και μπορεί, επίσης, να έχει καμία ή περισσότερες βοηθητικές κλάσεις αντικειμένων που προσθέτουν χαρακτηριστικά για αυτήν την καταχώρηση.
- Search Filters: Τα φίλτρα αναζήτησης χρησιμοποιούνται για τον καθορισμό κριτηρίων για τον προσδιορισμό καταχωρίσεων που περιέχουν συγκεκριμένα είδη πληροφοριών. Υπάρχουν διάφοροι τύποι φίλτρων αναζήτησης μεταξύ των οποίων παρουσία χαρακτηριστικού, ισότητα χαρακτηριστικού με τιμή, μεγαλύτερη/μικρότερη τιμή χαρακτηριστικού από προκαθορισμένη κ.ά.

2.2.11 RSS

Εικόνα 9 Λογότυπο RSS

Τα αρχικά RSS προκύπτουν από τον όρο Rich Site Summary (Σύνοψη Πλούσιας Σελίδας). Αναφέρεται σε μία μέθοδο μετάδοσης διαδικτυακού περιεχομένου που αλλάζει συχνά. Για τη μετάδοση των δεδομένων χρησιμοποιείται κείμενο σε μορφή XML. Πολλές ιστοσελίδες ενημερωτικού περιεχομένου παρέχουν αυτά που ονομάζουμε Ροές RSS (RSS Feeds), οι οποίες αποτελούν το εν λόγω ενημερωτικό περιεχόμενο. Για την ανάγνωση μίας ροής απαιτείται λογισμικό Αναγνώστη RSS (RSS Reader). Τέτοιο λογισμικό υποστηρίζουν όλοι οι σύγχρονοι φυλλομετρητές είτε εγγενώς είτε μέσω κάποιου πρόσθετου. Το RSS λύνει ένα πρόβλημα ανθρώπων που πλοηγούνται συχνά στο Διαδίκτυο. Τους επιτρέπει να μένουν ενήμεροι για τα ενδιαφέροντά τους, κατεβάζοντας το περιεχόμενο από ιστοσελίδες που παρέχουν ροές. Αυτό σημαίνει πως σώζουν χρόνο, καθώς δε χρειάζεται να πλοηγούνται σε σελίδες. Επίσης, κρατάνε την ιδιωτικότητά τους, καθώς δε χρειάζεται να δώσουν τη διεύθυνση e-mail τους για κάποιο newsletter.

Laravel και RSS

Το Laravel δεν παρέχει εγγενώς υποστήριξη για τη δημιουργία ροής RSS. Στην παρούσα υλοποίηση χρησιμοποιήθηκε ένα εξωτερικό πακέτο, το οποίο έδωσε τη δυνατότητα στα δεδομένα της ανακοίνωσης να μπορούν να μετασχηματιστούν σε μορφή για ροή RSS. Το πακέτο ονομάζεται spatie/laravel-feed και η εγκατάστασή του έγινε με το εργαλείο composer. Περισσότερες πληροφορίες για την υλοποίηση της συγκεκριμένης λειτουργίας θα αναφερθούν παρακάτω.

2.2.12 Λοιπά Laravel Features

Στο παρόν κεφάλαιο θα γίνει αναφορά σε ορισμένες λειτουργίες του Laravel που δεν μπορούσαν να ενταχθούν σε κάποιο από τα υπόλοιπα κεφάλαια, αλλά χρησιμοποιούνται στην εφαρμογή.

Laravel Notifications

Το Laravel εγγενώς υποστηρίζει αποστολή e-mails στους χρήστες. Υποστηρίζεται, επίσης, η αποστολή ειδοποιήσεων μέσω διαφόρων καναλιών επικοινωνίας, όπως τα SMS και το Slack (γνωστή εφαρμογή επικοινωνίας που χρησιμοποιείται κυρίως από οργανισμούς ή εταιρείες). Παρέχεται, ακόμη, η δυνατότητα για αποθήκευση των ειδοποιήσεων σε βάση δεδομένων, έτσι ώστε να γίνεται εμφάνισή τους σε περιβάλλον εφαρμογής. Οι ειδοποιήσεις έχουν, συνήθως, μικρό μέγεθος και περιλαμβάνουν μηνύματα με πληροφορίες που αφορούν κάποια ενέργεια που πραγματοποιήθηκε στην εφαρμογή.

Η δημιουργίας μίας ειδοποίησης γίνεται με την παρακάτω εντολή:

```
php artisan make:notification NotificationName
```

Μετά την εκτέλεσή της δημιουργείται στον φάκελο app/Notifications μία κλάση PHP, η κλάση της συγκεκριμένης ειδοποίησης. Σε αυτήν, υπάρχει μία μέθοδος με όνομα via και έναν αριθμό άλλων μεθόδων που μεταποιούν το μήνυμα της ειδοποίησης σε μορφή αναγνώσιμη από τα διαφορετικά κανάλια επικοινωνίας που έχουμε ορίσει.

Υποστηρίζεται, επίσης, η αποστολή ειδοποίησης κατ' ευθείαν μέσω του χρήστη ή η καθολική αποστολή ειδοποίησης σε μερίδα χρηστών. Είναι εφικτή και η προσθήκη των ειδοποιήσεων σε ουρά, ώστε να στέλνονται ανά ομάδες που ορίζονται από τον προγραμματιστή. Τέλος, μπορεί να οριστεί και χρονοκαθυστέρηση κατά την αποστολή μίας ειδοποίησης σε κάποιον χρήστη (π.χ. να ειδοποιηθεί μετά από 10 λεπτά για κάποιο συμβάν).

Στην παρούσα εφαρμογή, για την ώρα, υποστηρίζονται δύο ειδών ειδοποιήσεις: ειδοποίηση κατά την είσοδο του χρήστη ώστε να διατηρείται ιστορικό σύνδεσης και ειδοποίηση κατά τη δημιουργία μίας νέας ανακοίνωσης που περιλαμβάνει κάποια ετικέτα στην οποία ο χρήστης έχει κάνει εγγραφή. Περισσότερα για αυτές τις δύο ανακοινώσεις θα αναλυθούν σε επόμενο κεφάλαιο.

Laravel Events

Τα events (συμβάντα) στο Laravel παρέχουν μία υλοποίηση του προτύπου παρατηρητή (observer), που επιτρέπουν στην εφαρμογή να εγγράφεται σε διάφορα συμβάντα και να ακούει πότε αυτά συμβαίνουν. Τα events βρίσκονται στον φάκελο app/Events, ενώ οι Ακροατές (Listeners) στον φάκελο app/Listeners. Τα συμβάντα χρησιμεύουν ως ένας πολύ καλός τρόπος για την αποσύνδεση των διαφόρων πτυχών μίας εφαρμογής, καθώς ένα μεμονωμένο συμβάν μπορεί να έχει πολλούς ακροατές που δεν εξαρτώνται ο ένας από τον άλλο. Αυτό, πρακτικά, σημαίνει πως ο κώδικας για την αποστολή ειδοποιήσεων δε χρειάζεται να είναι συνδεδεμένος με τους controllers των διάφορων μοντέλων της εφαρμογής, αλλά σε ξεχωριστό σημείο.

Εγγραφή Συμβάντων και Ακροατών

Στο αρχείο EventServiceProvider.php στο φάκελο app/Providers γίνεται η δήλωση των συμβάντων και των αντίστοιχων ακροατών τους στη μορφή κλειδιού-πίνακα, δηλαδή κάθε συμβάν αντιστοιχίζεται σε έναν πίνακα από ακροατές. Η ιδιότητα που αυτό γίνεται είναι η $listen.

Η δημιουργία ενός συμβάντος είναι εύκολη και γίνεται με την παρακάτω εντολή:

```
php artisan event:generate
```

Υποστηρίζεται, επίσης, η χειροκίνητη εγγραφή συμβάντων, καθώς και η εγγραφή συμβάντων με σύμβολο υποκατάστασης (wildcard), που επιτρέπει στην εφαρμογή να «πιάνει» διάφορα συμβάντα και να πράττει αναλόγως. Τέλος, υποστηρίζεται και η αυτόματη ανακάλυψη συμβάντων κατά την οποία εγγράφονται όλοι οι ακροατές που βρίσκονται στον φάκελο Listeners που αναφέρθηκε παραπάνω και η αυτόματη ανακάλυψη συμβάντων.

Laravel Middleware

Το Middleware είναι γνωστό στα ελληνικά ως Ενδιάμεσο Λογισμικό. Είναι μία λειτουργία που δεν παρέχεται μονάχα από το Laravel, αλλά και από τα περισσότερα (αν όχι όλα) διαθέσιμα πλαίσια λογισμικού σε όλες τις γλώσσες προγραμματισμού (π.χ. Java, Python κ.ά.). Τα Ε.Λ. στο Laravel παρέχουν έναν μηχανισμό για το φιλτράρισμα όλων των HTTP αιτημάτων που περνάνε από την εφαρμογή.

Το Laravel παρέχει εγγενώς ορισμένα Ε.Λ. Για παράδειγμα, παρέχεται ένα Ε.Λ. το οποίο ελέγχει εάν ο χρήστης είναι συνδεδεμένος στην εφαρμογή. Εάν δεν είναι, τον ανακατευθύνει στην οθόνη σύνδεσης, εάν είναι, το αίτημα προχωράει. Φυσικά, περαιτέρω Ε.Λ. μπορούν να γραφτούν ώστε να πραγματοποιούνται και άλλες λειτουργίες. Ένα Ε.Λ. που αφορά το Cross-origin resource sharing (CORS) μπορεί να προσθέτει τις αντίστοιχες κεφαλίδες στα αιτήματα που φεύγουν από την εφαρμογή ή ένα άλλο να πραγματοποιεί μέτρηση της συχνότητας επίσκεψης μίας σελίδας. Όλα τα Ε.Λ. σε μία Laravel εφαρμογή βρίσκονται στον φάκελο app/Http/Middleware.

Τύποι Ενδιάμεσου Λογισμικού στο Laravel

Οι υποστηριζόμενοι από το Laravel τύποι ενδιάμεσου λογισμικού είναι οι εξής:

- Global Middleware (Καθολικό Ε.Λ.): Αυτός ο τύπος Ε.Λ. εκτελείται σε κάθε HTTP αίτημα που περνάει από την εφαρμογή. Ορίζεται στην ιδιότητα $middleware στην κλάση app/Http/Kernel.php.
- Route Middleware (Ε.Λ. για συγκεκριμένη διαδρομή): Ο συγκεκριμένος τύπος Ε.Λ. αφορά μία ή περισσότερες διαδρομές της εφαρμογής. Ορίζονται στην ίδια κλάση με τα παραπάνω, αλλά σε αυτήν την περίπτωση, χρησιμοποιείται η ιδιότητα $routeMiddleware. Η εν λόγω ιδιότητα έχει τη μορφή κλειδιού-τιμής, όπου το κλειδί είναι ένα όνομα που δίνει ο προγραμματιστής για να αναγνωρίζει το Ε.Λ., ενώ η τιμή είναι η κλάση του Ε.Λ. Τέλος, ορίζεται η χρήση του Ε.Λ. είτε στο αρχείο με τις διαδρομές της διαδικτυακής εφαρμογής είτε στο αρχείο με τις διαδρομές του API.
- Middleware Groups (Ομάδες Ε.Λ.): Σε ορισμένες περιπτώσεις, έχει νόημα τα Ε.Λ. να κατηγοριοποιούνται σε ομάδες ανάλογα με τη χρήση τους. Αυτή ακριβώς είναι και η χρήση του συγκεκριμένου τύπου Ε.Λ. Το Laravel εγγενώς περιλαμβάνει δύο ομάδες Ε.Λ., μία με όνομα web που αφορά τη διαδικτυακή εφαρμογή και μία με όνομα API που αφορά το API. Ορίζονται στην ιδιότητα $middlewareGroups.

Προτεραιότητα και Παράμετροι στα Ε.Λ.

Σε ορισμένες περιπτώσεις μπορεί να χρειαστεί τα Ε.Λ. να τρέξουν με μία ορισμένη σειρά για λόγους που επιβάλλει ο προγραμματιστής. Η σειρά μπορεί να οριστεί στην ιδιότητα $middlewarePriority στην κλάση app/Http/Kernel.php.

Ακόμη, τα Ε.Λ. στο Laravel μπορούν να λάβουν παραμέτρους. Για παράδειγμα, μπορεί να χρειάζεται να ελεγχθεί πως ο χρήστης έναν συγκεκριμένο ρόλο πριν του δοθεί η άδεια να προσπελάσει μία σελίδα. Για αυτόν τον σκοπό, κατά τον ορισμό ενός Ε.Λ. σε μία διαδρομή, χρησιμοποιείται ο χαρακτήρας «:» ανάμεσα στο όνομα και στις παραμέτρους που μεταφέρονται στο Ε.Λ. Τέλος, μπορούν να περαστούν πολλές παράμετροι με τη χρήση του χαρακτήρα «,».

2.3 Τεχνολογίες Front-end

Μία ακόμη σημαντική απόφαση που κληθήκαμε να λάβουμε, ήταν η επιλογή του πλαισίου που θα χρησιμοποιηθεί για το front-end. Πλέον, υπάρχουν διάφορα πλαίσια που προσφέρουν τις απαραίτητες λειτουργίες που θα έκαναν την εφαρμογή τόσο χρηστική όσο και εύκολα συντηρήσιμη και επεκτάσιμη.

Επιλέχθηκε το Vue.js καθώς ήταν το μόνο πλήρες πλαίσιο στο οποίο ο γράφον είχε εμπειρία κατά την επιλογή της συγκεκριμένης εργασίας. Η εφαρμογή θα μπορούσε κάλλιστα να γραφτεί και χωρίς τη χρήση κάποιου πλαισίου, αλλά θα την έκανε πολύ δύσκολη στη συντήρηση.

2.3.1 HTML/ CSS/ JavaScript

Εικόνα 10 Λογότυπα HTML/ CSS/ JS

Το HTML (HyperText Markup Language)είναι το βασικό δομικό στοιχείο του ιστού (web). Καθορίζει την έννοια και τη δομή του περιεχομένου του ιστού. Πέραν του HTML, χρησιμοποιείται το CSS (Cascading Style Sheets) το οποίο είναι υπεύθυνο για την εμφάνιση μίας ιστοσελίδας, καθώς και το JavaScript που είναι υπεύθυνο για τη λειτουργικότητα.

Το "Hypertext" αναφέρεται σε συνδέσμους που συνδέουν ιστοσελίδες μεταξύ τους, είτε μέσα σε έναν ιστότοπο είτε μεταξύ ιστότοπων. Οι σύνδεσμοι είναι μια θεμελιώδης πτυχή του ιστού. Το HTML χρησιμοποιεί "σήμανση" για να διαχωρίσει κείμενο, εικόνες και άλλο περιεχόμενο για προβολή σε κάποιο πρόγραμμα περιήγησης (π.χ. Mozilla Firefox, Google Chrome κ.α.). Η σήμανση HTML περιλαμβάνει ειδικά "στοιχεία" όπως για τα παράδειγμα τα: <head>, <title>, <body>, <header>,

<footer> και πολλά άλλα. Τα διάφορα στοιχεία του HTML ξεχωρίζουν μεταξύ τους μέσω των "tags", τα οποία αποτελούνται από το όνομα του στοιχείου που περιβάλλεται από "<" και ">".

Το Cascading Style Sheets (CSS) είναι μια stylesheet γλώσσα που χρησιμοποιείται για να περιγράψει την παρουσίαση ενός εγγράφου γραμμένο σε HTML ή XML . Το CSS περιγράφει πώς τα στοιχεία πρέπει να αποδίδονται στην οθόνη, σε χαρτί, σε ομιλία ή σε άλλα μέσα. Το CSS είναι μια από τις βασικές γλώσσες του ανοιχτού ιστού και τυποποιείται σε προγράμματα περιήγησης σύμφωνα με τις προδιαγραφές του W3C (Κοινοπραξία του Παγκοσμίου Ιστού, WWW Consortium ή W3C). Μέχρι πρότινος, η ανάπτυξη των διαφόρων τμημάτων των προδιαγραφών CSS έγινε συγχρονισμένα, γεγονός που επέτρεψε την έκδοση της τελευταίας σύστασης. Από το CSS3, το εύρος των προδιαγραφών αυξήθηκε σημαντικά και η πρόοδος σε διαφορετικές ενότητες του CSS άρχισε να διαφέρει τόσο πολύ, ώστε έγινε πιο αποτελεσματική η ανάπτυξη συστάσεων ξεχωριστά ανά ενότητα.

Το JavaScript (JS) είναι μια lightweight, ερμηνευμένη ή just-in-time γλώσσα προγραμματισμού με υποστήριξη για first-class functions. Αν και είναι πιο γνωστή ως γλώσσα προγραμματισμού για ιστοσελίδες, χρησιμοποιείται, επίσης, από διάφορα μη browser based περιβάλλοντα όπως το Node.js, το Apache CouchDB και το Adobe Acrobat. Το JavaScript είναι μία prototype-based, multi-paradigm, single-threaded, δυναμική γλώσσα προγραμματισμού, που υποστηρίζει αντικειμενοστραφή, imperative και declarative στυλ προγραμματισμού.
Το πρότυπο για το JavaScript ονομάζεται ECMAScript. Από το 2012, όλα τα σύγχρονα προγράμματα περιήγησης υποστηρίζουν πλήρως το ECMAScript 5.1. Τα παλαιότερα προγράμματα περιήγησης υποστηρίζουν τουλάχιστον ECMAScript 3. Το 2015 η ECMA International δημοσίευσε την έκτη major έκδοση του ECMAScript, η οποία ονομάζεται επίσημα ECMAScript 2015, και αρχικά αναφέρεται ως ECMAScript 6 ή ES6.

CSS framework

CSS framework ονομάζεται κάθε βιβλιοθήκη που επιτρέπει την ευκολότερη και πιο επικείμενη στα διάφορα web standards (π.χ. mobile-first development κ.α.) ανάπτυξη διαδικτυακών διεπαφών χρήστη με τη χρήση CSS. Πολύ γνωστά και ευραίως χρησιμοποιούμενα CSS frameworks είναι το Bootstrap, το Foundation και το Semantic UI. Τα CSS frameworks παρέχουν συνήθως styles για τις φόρμες, τα εικονίδια, τις εικόνες, τα headers και footers και πολλά άλλα elements. Παρέχουν, ακόμη, και υποστήριξη για διαφόρων ειδών grids και layouts.

Εικόνα 11 Λογότυπο Bulma

Στην παρούσα εφαρμογή χρησιμοποιήθηκε το Bulma CSS framework. Η κύρια διαφορά του συγκεκριμένου με τα προαναφερθέντα είναι πως δεν παρέχει build-in υποστήριξη για JavaScript, δηλαδή όλη η λογική είναι αναγκαίο να γραφτεί εξ αρχής. Για παράδειγμα, για το άνοιγμα και το

κλείσιμο κάποιου dropdown χρειάζεται να γράψουμε κάποιον watcher που θα ελέγχει το state για το πάτημα του κουμπιού του dropdown ώστε να προσθέτει ή να αφαιρεί το αντίστοιχο CSS class. Για τη συγκεκριμένη εφαρμογή μας βολεύει αυτή η λειτουργικότητα, καθώς τα πάντα ελέγχονται από το Vue.js framework για το οποίο θα γίνει ανάλυση παρακάτω. Είναι, επίσης, πολύ ελαφρύ και εξ ολοκλήρου ανοιχτού κώδικα.

Laravel και Blade Templates

Όπως αναφέρθηκε και παραπάνω, το Laravel χρησιμοποιεί τα blade templates για τα views του. Όπως, επίσης, αναφέρθηκε, τα blade templates δίνουν τη δυνατότητα για inheritance και sections. Μία τυπική Laravel εφαρμογή που τα χρησιμοποιεί, έχει τη δυνατότητα να κρατά ένα σταθερό layout για την εφαρμογή και να αλλάζει δυναμικά το περιεχόμενο της εκάστοτε σελίδας αναλόγως με τις επιθυμίες του προγραμματιστή. Άξιο αναφοράς είναι ακόμη το γεγονός πως στα blade templates μπορεί να συμπεριληφθεί και plain PHP κώδικας.

Στα blade templates γίνεται συνήθως το import στατικών αρχείων (css και js) είτε από κάποιο public CDN (Content Delivery Network) είτε έπειτα από μετάφραση SASS/SCSS και js αρχείων. Στην παρούσα εφαρμογή γίνεται χρήση του δεύτερου τρόπου μέσω του Laravel Mix. Για το τελευταίο θα γίνει παρουσίαση σε επόμενη ενότητα.

2.3.2 npm

Εικόνα 12 Λογότυπο npm

Το npm (Node Package Manager) είναι ένα αποθετήριο (repository) αποτελούμενο κυρίως από open source projects που αφορούν το Node.js και το JavaScript. Το Node.js, περιληπτικά, είναι μια πλατφόρμα ανάπτυξης λογισμικού (κυρίως διακομιστών) χτισμένη σε περιβάλλον JavaScript. Σε αντίθεση από τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών δικτύων, μία διεργασία node δεν στηρίζεται στην πολυνηματικότητα, αλλά σε ένα μοντέλο ασύγχρονης επικοινωνίας. Καθώς αποτελεί τεχνολογία back-end, ο γράφον δε θεωρεί πως χρήζει περαιτέρω αναφοράς στην παρούσα εργασία, καθώς, όπως γνωρίζουμε, για τη συγκεκριμένη εφαρμογή έχει χρησιμοποιηθεί το Laravel ως τεχνολογία back-end.

Το npm αποτελεί ένα command line tool το οποίο εγκαθίσταται μαζί με το Node.js και οι κύριες λειτουργίες του είναι δύο:

- διαχείριση dependencies: υποστήριξη για εγκατάσταση dependencies με versioning
- διαχείριση build scripts: δυνατότητα για διαφορετικά build scripts ανάλογα με το environment (production ή staging/testbed) με τις ανάλογες ευκολίες που αυτό δίνει στον προγραμματιστή

Για να γίνει δυνατή η χρήση των δυνατοτήτων του npm αρκεί ένα αρχείο JSON (package.json) στον φάκελο στον οποίο θα τρέξουμε την εντολή για εγκατάσταση των dependencies (npm install). Κατόπιν, θα δημιουργηθεί ένας φάκελος με όνομα node_modules μέσα στον οποίο θα υπάρχουν όλα εκείνα τα

modules που έχουμε ορίσει, καθώς και πιθανά τρίτα modules που χρειάζονται τα πακέτα που εγκαταστήσαμε.
Ένα διάσημο εναλλακτικό του npm άξιο αναφοράς είναι το yarn (ανεπτυγμένο από το Facebook), το οποίο χρησιμοποιεί ίδιο repository με το npm και θεωρείται πως προσφέρει ορισμένα performance enchancements κατά τη λειτουργία του.

Laravel Mix

Το Laravel Mix είναι ένα package για το Laravel που δίνει τη δυνατότητα στον προγραμματιστή να ορίζει τα βήματα για τη δημιουργία των static αρχείων μίας εφαρμογής. Ακολουθεί ένα παράδειγμα:

```
mix.js('resources/js/app.js', 'public/js')
    .sass('resources/sass/app.scss', 'public/css');
```

Σύμφωνα με το παραπάνω, το Mix θα μεταγλωττίσει το αρχείο στον φάκελο resources/js/app.js με φάκελο προορισμού τον public/js. Αντίστοιχα θα πράξει με το αρχείο scss. Για να τρέξει το Mix χρειάζεται το npm, άλλωστε και το ίδιο είναι npm package. Μπορεί να χρησιμοποιηθεί για compilation αρχείων less, sass, stylus, postcss καθώς και plain css σε css. Τέλος, παρέχει τη δυνατότητα για διάφορες custom επιλογές αναφορικά με το build, όπως versioning, merge πολλών αρχείων σε ένα, compilation Vue/React components και διάφορες άλλες που, όμως, δεν έχουν χρησιμοποιηθεί στο παρόν project.

2.3.3 Vue.js

Εικόνα 13 Λογότυπο Vue.js

Το Vue είναι ένα προοδευτικό framework για τη δημιουργία διεπαφών χρήστη. Με τον όρο προοδευτικό εννοείται πως μπορεί να χρησιμοποιηθεί αυτούσιο σε εφαρμογές σε συνεργασία με απλή HTML. Η βασική βιβλιοθήκη εστιάζεται μόνο στο επίπεδο προβολής (view) και είναι εύκολο να ενσωματωθεί σε άλλες βιβλιοθήκες ή υπάρχοντα project. Από την άλλη πλευρά, το Vue είναι επίσης απόλυτα ικανή να χρησιμοποιηθεί σε εξελιγμένες εφαρμογές μίας σελίδας (single page applications) μαζί με σύγχρονα εργαλεία και βιβλιοθήκες. Η βιβλιοθήκη έχει γραφτεί από τον Evan You και η πιο πρόσφατη σταθερή της έκδοση της κατά τη συγγραφή του παρόντος είναι η 2.6.11.

Κύρια χαρακτηριστικά

Virtual DOM

Το VueJS χρησιμοποιεί τη χρήση εικονικού DOM (virtual DOM), το οποίο χρησιμοποιείται και από άλλα frameworks. Οι αλλαγές δεν γίνονται στο DOM, αλλά δημιουργείται ένα αντίγραφο του DOM που υπάρχει με τη μορφή δομών δεδομένων JavaScript. Κάθε φορά που χρειάζεται να γίνουν αλλαγές, γίνονται στις δομές δεδομένων JavaScript και η τελευταία συγκρίνεται με την αρχική δομή δεδομένων. Οι τελικές αλλαγές στη συνέχεια ενημερώνονται στο πραγματικό DOM, το οποίο ο χρήστης θα δει να αλλάζει. Αυτό είναι καλό όσον αφορά τη βελτιστοποίηση, είναι λιγότερο ακριβό σε χρόνο CPU και οι αλλαγές μπορούν να γίνουν με ταχύτερο ρυθμό.

Data Binding

Η δυνατότητα δέσμευσης δεδομένων (data binding) βοηθά στο χειρισμό ή την εκχώρηση τιμών σε HTML attributes, την αλλαγή στυλ, την εκχώρηση κλάσεων με τη βοήθεια της οδηγίας που ονομάζεται v-bind.

Components

Τα components είναι ένα από τα σημαντικότερα χαρακτηριστικά του VueJS που βοηθά στη δημιουργία προσαρμοσμένων στοιχείων, τα οποία μπορούν να επαναχρησιμοποιηθούν. Αυτό δίνει στον προγραμματιστή τη δυνατότητα να «σπάει» τη διεπαφή του σε μικρότερα στοιχεία που μπορούν να επαναχρησιμοποιηθούν. Για να γίνει χρήση ενός component από το VueJS, θα πρέπει πρώτα να γίνει register.

Event Handling

Μέσω της οδηγίας v-on τα διάφορα στοιχεία μπορούν να περιμένουν για συγκεκριμένα events και να τρέξουν κάποιο κομμάτι κώδικα όταν το event γίνει trigger.

Animation/Transition

Το VueJS παρέχει διάφορους τρόπους εφαρμογής κίνησης/μετάβασης σε στοιχεία HTML όταν αυτά προστίθενται, ενημερώνονται ή αφαιρούνται από το DOM. Μπορούμε εύκολα να προσθέσουμε τρίτες βιβλιοθήκες κίνησης ώστε να προσθέσουμε περισσότερη διαδραστικότητα στη διεπαφή.

Computed Properties

Από τα σημαντικότερα χαρακτηριστικά του VueJS. Δίνει τη δυνατότητα να ακούμε για τις αλλαγές που έγιναν στα στοιχεία διεπαφής χρήστη και να εκτελεί τους απαραίτητους υπολογισμούς. Δεν χρειάζεται επιπλέον κώδικας για αυτό.

Templates

Το VueJS παρέχει πρότυπα που βασίζονται σε HTML και συνδέουν το DOM με τα δεδομένα του Vue instance.

Directives

Το VueJS έχει ενσωματωμένες οδηγίες όπως v-if, v-else, v-show, v-on, v-bind και v-model, οι οποίες χρησιμοποιούνται για την εκτέλεση διαφόρων ενεργειών στο frontend.

Watchers

Οι watchers εφαρμόζονται σε δεδομένα που μεταβάλλονται. Για παράδειγμα, στα στοιχεία φορμών. Φροντίζουν, ακόμη, τον χειρισμό τυχόν αλλαγών που ορίσει ο προγραμματιστής σε αυτά τα δεδομένα, κάνοντας τον προγραμματισμό πιο απλό και γρήγορο.

Vue instance

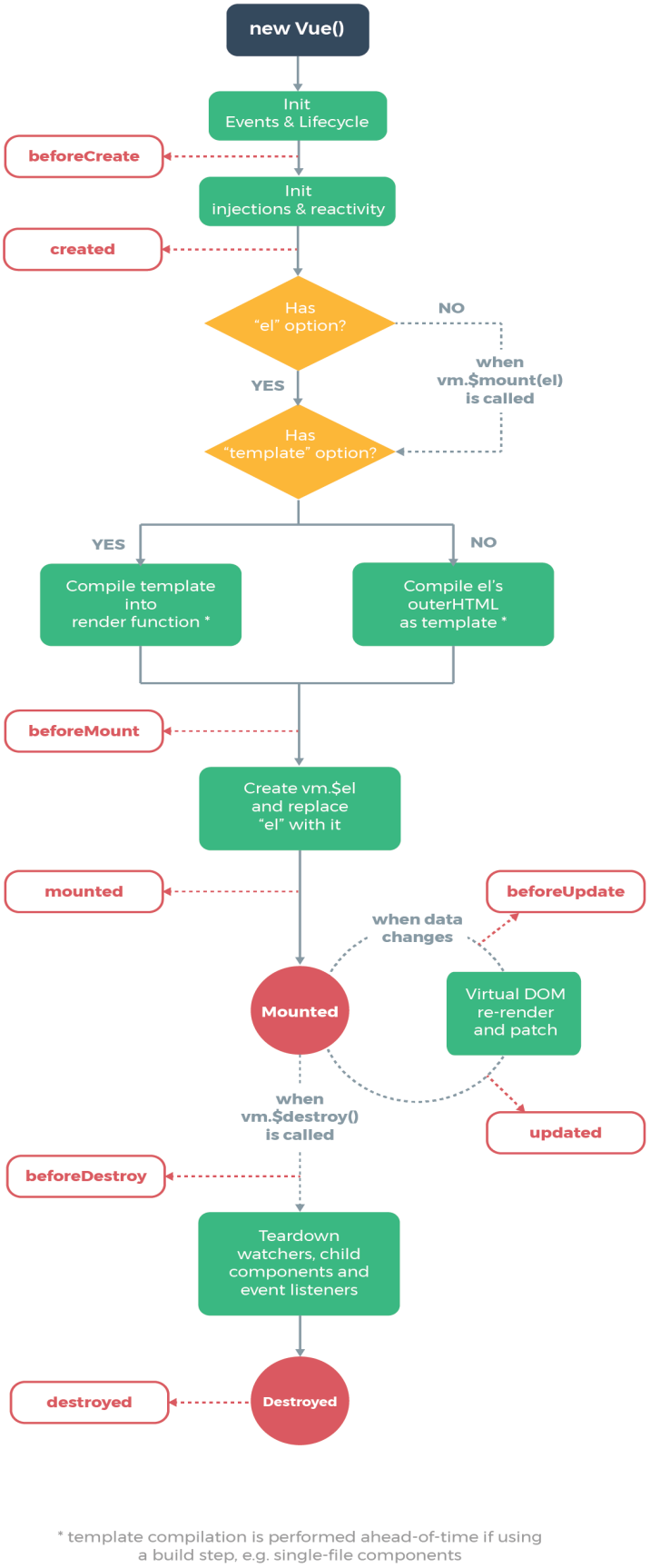
Στις παραπάνω γραμμές, έγινε αναφορά στο Vue instance. Το Vue instance ορίζει μία μεταβλητή πάνω στην οποία δημιουργείται το αντικείμενο Vue. Αυτό ονομάζεται root Vue instance. Όταν δημιουργείται ένα instance, προστίθενται στο σύστημα διαδραστικότητας (reactivity system) του Vue όλα τα data που έχει αυτό το αντικείμενο. Έτσι, όταν κάποιο από αυτά αλλάζει, φαίνεται και η αντίστοιχη αλλαγή στη διεπαφή χρήστη. Το Vue instance συνήθως γίνεται bind σε κάποιο HTML element, div κατά κύριο λόγο, και μέσα σε αυτό το element κάνει render τα δεδομένα ή το template που έχουμε ορίσει. Είναι σημαντικό να έχουμε μόνο ένα root Vue instance.

```JS
var vm = new Vue({
  // options
})
```

Εικόνα 14 Ορισμός νέου Vue instance

Instance Lifecycle Hooks

Κάθε instance περνάει από διάφορα στάδια όταν δημιουργείται. Τα κύρια στάδια είναι τα created, mounted, updated και destroyed.



Εικόνα 15 Διάγραμμα ροής Vue component

Components

Όπως αναφέρθηκε και παραπάνω, τα components είναι προσαρμοσμένα στοιχεία που μπορούν να επαναχρησιμοποιηθούν. Τα components είναι ονοματισμένα Vue instances που μπορούν να είναι κομμάτι του root Vue instance. Κάθε φορά που χρησιμοποιείται ένα component, ένα νέο instance δημιουργείται, πράγμα που δίνει τη δυνατότητα κάθε component να περιέχει το δικό του state. Αυτό είναι χρήσιμο όταν χρησιμοποιούμε περισσότερες από μία φορές το ίδιο component μέσα σε ένα Vue instance.

```JS
// Define a new component called button-counter
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button v-on:click="count++">Clicked: {{ count }} times.</button>'
})
```

Εικόνα 16 Παράδειγμα Vue component

Κάθε component θα πρέπει να έχει ένα κεφαλικό (root) HTML element. Σε διαφορετική περίπτωση, δε θα γίνει render από το Vue, εμφανίζοντας σφάλμα κατά το compile. Όπως για παράδειγμα, στην παρακάτω εικόνα:

```HTML
<h3>{{ title }}</h3>
<div v-html="content"></div>
```

Εικόνα 17 Παράδειγμα λάθος Vue component

Props

Σχεδόν κανένα component δεν είναι χρήσιμο εάν δεν υπάρχει ένας μηχανισμός να περάσει δεδομένα σε αυτό. Αυτή είναι η δυνατότητα που προσδίδουν στα components τα props.

Τα props είναι προσαρμοσμένα χαρακτηριστικά που μπορούν να εγγραφούν στα components. Όταν μία τιμή περνάει σε ένα prop, γίνεται χαρακτηριστικό του instance του component και μπορεί να χρησιμοποιηθεί σε αυτό. Δεν υπάρχει περιορισμός για τον αριθμό των props που ένα component μπορεί να έχει και, εξ ορισμού, οποιαδήποτε τιμή μπορεί να περάσει σε οποιοδήποτε prop. Αυτό, βέβαια, είναι παραμετροποιήσιμο, καθώς ο προγραμματιστής μπορεί να ορίσει τι τύπου είναι το prop και σε περίπτωση που περαστεί λάθος τύπος σε αυτό, να εμφανίζεται κάποιο error. Τέλος, τα props μπορούν να οριστούν ως απαιτούμενα (required) για την εμφάνισή τους στο component.

```JS
Vue.component('blog-post', {
  // camelCase in JavaScript
  props: ['postTitle'],
  template: '<h3>{{ postTitle }}</h3>'
})
```

Εικόνα 18 Παράδειγμα Vue prop

Events

Καθώς δημιουργούμε τα components και προσθέτουμε σε αυτά props, πολλές φορές εγείρεται η ανάγκη επικοινωνίας με το γονικό (parent) component. Για παράδειγμα, μπορεί να θέλουμε να αυξήσουμε το μέγεθος της γραμματοσειράς σε μόνο ένα σημείο της σελίδας μας, αφήνοντας την υπόλοιπη ως έχει. Σε αυτό έρχονται να βοηθήσουν τα custom events που αναφέραμε παραπάνω.

Το γονικό component περιμένει για ένα συγκεκριμένο event που έχει οριστεί από τον προγραμματιστή ώστε να πραγματοποιήσει κάποια συγκεκριμένη ενέργεια. Αυτό γίνεται με την οδηγία v-on. Το εμφωλευμένο component με τη σειρά του, θα χρειαστεί να εκπέμψει το event. Αυτό γίνεται με τη χρήση της ενσωματωμένης μεθόδου $emit περνώντας ως όρισμα το όνομα που έχουμε δώσει στο event. Έτσι, το γονικό component θα λάβει την κλήση του event και θα τρέξει τις εντολές ή τη μέθοδο που έχουμε ορίσει.

```JS
Vue.component('base-checkbox', {
  model: {
    prop: 'checked',
    event: 'change'
  },
  props: {
    checked: Boolean
  },
  template: `
    <input
      type="checkbox"
      v-bind:checked="checked"
      v-on:change="$emit('change', $event.target.checked)"
    >
  `
})
```

Εικόνα 19 Παράδειγμα Vue custom event

2.3.4 Nginx

Εικόνα 20 Λογότυπο Nginx

Το Nginx είναι ένα open source λογισμικό web server που μπορεί, επίσης, να χρησιμοποιηθεί ως reverse proxy, load balancer, mail proxy και caching server. Το λογισμικό είναι γνωστό για την αποδοτικότητα, τη σταθερότητα, την εύκολη διαχείρισή και τη μικρή του κατανάλωση σε πόρους συστήματος. Λόγω αυτού, χρησιμοποιείται από πολλά από τα διασημότερα sites του ιστού, όπως τα: Netflix, Hulu, Pinterest, Cloudflare, Airbnb και πολλά άλλα.

Το Nginx γράφτηκε με αρχικό σκοπό να παρέχει καλύτερες επιδόσεις συγκριτικά με τον Apache web server που για πολλά χρόνια αποτελούσε τον κύριο web server στον ιστό. Υπάρχουν διάφοροι λόγοι για να γίνει επιλογή του ενός web server έναντι του άλλου, αλλά ξεφεύγουν από τη σκοπιά της συγκεκριμένης εργασίας. Περιλαμβάνονται σύνδεσμοι στη βιβλιογραφία που έχουν ασχοληθεί με το συγκεκριμένο ζήτημα και εξάγουν ασφαλή συμπεράσματα.

Laravel και Nginx

Η παρούσα εφαρμογή χρησιμοποιεί το Nginx ως web server για τους λόγους που αναφέρθηκαν παραπάνω. Ακόμη, αξίζει να σημειωθεί πως και το ίδιο το documentation του Laravel προτείνει τη χρήση Nginx, δίνοντας μάλιστα και ένα αρχικό sample configuration file. Από την εμπειρία του γράφοντος, δε θεωρείται πως απαιτούνται εκτενείς αλλαγές σε αυτό ώστε να είναι σε θέση να ανταπεξέλθει σε production environments. Οι μόνες προσθήκες που έγιναν αφορούσαν το caching στατικών αρχείων στον browser του εκάστοτε χρήστη που πραγματοποιεί σύνδεση στην εφαρμογή, ώστε να υπάρχει το performance boost που αυτό δίνει.

Για να είναι το Nginx σε θέση να τρέξει κώδικα PHP χρειάζεται κάποιον handler. Το PHP Fast Process Manager (php-fpm) είναι ένας τέτοιος χειριστής FastCGI για scripts και εφαρμογές PHP. Συνήθως συνδυάζεται με τους διακομιστές ιστού για την εξυπηρέτηση εφαρμογών που απαιτούν ένα PHP framework, ενώ ο διακομιστής επιστρέφει HTML, JavaScript και άλλο περιεχόμενο εκτός της PHP.

2.4 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκαν όλες οι τεχνολογίες που χρησιμοποιούνται για την υλοποίηση της εφαρμογής. Αναλύθηκαν τόσο οι τεχνολογίες για το back-end όσο και για το front-end. Ακόμη, σε πολλές περιπτώσεις, αναφέρθηκε ο τρόπος δημιουργίας των διαφόρων συστατικών που απαρτίζουν την εφαρμογή.

Κεφάλαιο 3ο: Σχεδίαση του συστήματος

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει ανάλυση του σχήματος της βάσης της εφαρμογής και του τρόπου με τον οποίο τα πεδία συνδέονται μεταξύ τους. Θα αναλυθούν, επίσης, οι σχέσεις μεταξύ των αντικειμένων/μοντέλων, όπως, επίσης, και το ενδιάμεσο λογισμικό που χρησιμοποιείται στο front-end και στο back-end.

3.2 MySQL Schema

Εικόνα 21 Σχήμα βάσης δεδομένων

Για τη δημιουργία της παραπάνω εικόνας, χρησιμοποιήθηκε η εφαρμογή https://dbdiagram.io/.

Ανάλυση πεδίων πινάκων

Σε αυτό το σημείο, θα γίνει μία ανάλυση των σημαντικότερων πινάκων στο σχήμα της βάσης δεδομένων της εφαρμογής.

Announcements (ανακοινώσεις)

- id: bigint(20) και auto_increment. Αποτελεί το αναγνωριστικό κάθε ανακοίνωσης και είναι το πεδίο βάσει του οποίου γίνονται οι αλλαγές και οι αναζητήσεις. Είναι το κύριο κλειδί.
- title: varchar(255). Αποτελεί τον τίτλο κάθε ανακοίνωσης στα Ελληνικά. Βάσει αυτού του πεδίου γίνονται και αναζητήσεις από τον χρήστη.
- body: mediumtext. Αποτελεί το κείμενο της κάθε ανακοίνωσης στα Ελληνικά.
- is_event: boolean. Ελέγχει εάν η ανακοίνωση θα σημανθεί και ως εκδήλωση.
- event_start_time: timestamp. Προαιρετικό πεδίο που ορίζει την ώρα έναρξης μίας εκδήλωσης.
- event_end_time: timestamp. Προαιρετικό πεδίο που ορίζει την ώρα λήξης μίας εκδήλωσης.
- event_location: varchar(255). Προαιρετικό πεδίο που ορίζει την τοποθεσία μίας εκδήλωσης.
- created_at: timestamp. Ορίζει την ώρα δημιουργίας μίας ανακοίνωσης.
- updated_at: timestamp. Ορίζει την ώρα της τελευταίας ενημέρωσης μίας ανακοίνωσης.
- deleted_at: timestamp. Ορίζει την ώρα διαγραφής μίας ανακοίνωσης. Σε περίπτωση που είναι null, θεωρείται πως η ανακοίνωση είναι σε ισχύ.
- is_pinned: boolean. Ορίζει εάν μία ανακοίνωση θα σημανθεί ως σημαντική ώστε να είναι μονίμως στην αρχική οθόνη της εφαρμογής.
- gmaps: boolean. Ορίζει, σε μία εκδήλωση, εάν θα υπάρχει σύνδεσμος της εφαρμογής στο Google Maps.
- has_eng: boolean. Ορίζει εάν μία ανακοίνωση είναι διαθέσιμη στα αγγλικά.
- eng_title: varchar(255). Αποτελεί τον τίτλο κάθε ανακοίνωσης στα Αγγλικά. Βάσει αυτού του πεδίου γίνονται και αναζητήσεις από τον χρήστη.
- eng_body: mediumtext. Αποτελεί το κείμενο της κάθε ανακοίνωσης στα Αγγλικά.
- pinned_until: timestamp. Ορίζει, σε μία ανακοίνωση που έχει σημανθεί ως σημαντική, το χρονικό περιθώριο εμφάνισής της στην αρχική της εφαρμογής.
- user_id: bigint(20). Αναφορά στο αναγνωριστικό του χρήστη που έχει δημιουργήσει την ανακοίνωση.

Tags (ετικέτες)

- id: bigint(20) και auto_increment. Αποτελεί το αναγνωριστικό κάθε ετικέτας και είναι το πεδίο βάσει του οποίου γίνονται οι αλλαγές και οι αναζητήσεις. Είναι το κύριο κλειδί.
- title: varchar(255). Αποτελεί τον τίτλο κάθε ετικέτας.
- parent_id: bigint(20). Αποτελεί το αναγνωριστικό της γονικής ετικέτας για καθεμία από τις ετικέτες. Εάν είναι null, η ετικέτα θεωρείται κεφαλική (root).
- is_public: boolean. Ορίζει εάν μία ετικέτα θα είναι δημόσια προσβάσιμη ή όχι.
- created_at: timestamp. Ορίζει την ώρα δημιουργίας μίας ετικέτας.
- updated_at: timestamp. Ορίζει την ώρα της τελευταίας ενημέρωσης μίας ετικέτας.
- deleted_at: timestamp. Ορίζει την ώρα διαγραφής μίας ετικέτας. Σε περίπτωση που είναι null, θεωρείται πως η ετικέτα είναι ενεργεί.

Announcement_Tag (ανακοίνωση – ετικέτα)

Συσχετίζει τις ανακοινώσεις με τις ετικέτες.

- announcement_id: bigint(20). Αναφορά σε id μίας ανακοίνωσης.
- tag_id: bigint(20). Αναφορά σε id μίας ετικέτας.

Στον συγκεκριμένο πίνακα το κύριο κλειδί είναι και τα δύο πεδία.

Attachments (επισυνάψεις)

- id: bigint(20) και auto_increment. Αποτελεί το αναγνωριστικό κάθε επισύναψης. Είναι το κύριο κλειδί.
- announcement_id: bigint(20). Αποτελεί το αναγνωριστικό της ανακοίνωσης στην οποία «ανήκει» η επισύναψη.
- filename: varchar(255). Το όνομα του αρχείου επισύναψης.
- content: mediumblob. Είναι το περιεχόμενο της κάθε ανακοίνωσης σε μορφή blob.
- filesize: int(10). Αναπαριστά το μέγεθος του αρχείου σε bytes.
- mime_type: varchar(255). Ορίζει τον τύπο του αρχείου κάθε επισύναψης.
- created_at: timestamp. Ορίζει την ώρα δημιουργίας μίας επισύναψης.
- updated_at: timestamp. Ορίζει την ώρα της τελευταίας ενημέρωσης μίας επισύναψης.
- deleted_at: timestamp. Ορίζει την ώρα διαγραφής μίας επισύναψης. Σε περίπτωση που είναι null, θεωρείται πως η επισύναψη εμφανίζεται στην ανακοίνωσή της.

Notifications (ειδοποιήσεις)

- id: char(36). Αποτελεί το αναγνωριστικό της κάθε ανακοίνωσης. Είναι το κύριο κλειδί.
- type: varchar(255). Ορίζει τον τύπο της κάθε ειδοποίησης όπως τον έχουμε ορίσει προγραμματιστικά (π.χ. UserLoggedIn).
- notifiable_type: varchar(255). Ορίζει το μοντέλο το οποίο αφορά η ειδοποίηση.
- notifiable_id: bigint(20). Ορίζει το αναγνωριστικό του μοντέλου που αφορά η ανακοίνωση.
- data: text. Το περιεχόμενο της ανακοίνωσης. Έχει μορφή JSON.
- created_at: timestamp. Ορίζει την ώρα δημιουργίας της ειδοποίησης.
- updated_at: timestamp. Ορίζει την ώρα αλλαγής της ειδοποίησης.
- read_at: timestamp. Ορίζει την ώρα ανάγνωσης της ειδοποίησης.

Users (χρήστες)

- id: bigint(20). Το αναγνωριστικό του κάθε χρήστη στη βάση. Χρησιμοποιείται για αναγνώριση των ανακοινώσεων του χρήστη, καθώς και στις ειδοποιήσεις. Αποτελεί το κύριο κλειδί.
- name: varchar(255). Το πλήρες όνομα του χρήστη όπως έρχεται από τον LDAP server του τμήματος.
- email: varchar(255). Το email του χρήστη όπως έρχεται από τον LDAP server του τμήματος.
- created_at: timestamp. Ορίζει την ώρα δημιουργίας του χρήστη.
- updated_at: timestamp. Ορίζει την ώρα τελευταίας ενημέρωσης του χρήστη.
- is_author: boolean. Ορίζει εάν ο χρήστης αποτελεί συντάκτη. Η αρχική τιμή του είναι ψευδής. Ένας συντάκτης έχει δικαίωμα προσθήκης ανακοίνωσης.
- is_admin: boolean. Ορίζει εάν ο χρήστης είναι διαχειριστής. Η αρχική τιμή του είναι ψευδής. Ένας διαχειριστής έχει δικαίωμα να προσθέτει, να αφαιρεί και να τροποποιεί ετικέτες.
- last_login_at: timestamp. Αποτελεί την τελευταία χρονική στιγμή που πραγματοποίησε είσοδο στην εφαρμογή ο χρήστης.
- uid: varchar(255). Αποτελεί το uid του χρήστη όπως έρχεται από το σύστημα LDAP του τμήματος.

Ετικέτα_Χρήστης (tag_user)

Συσχετίζει τους χρήστες με τις ετικέτες στις οποίες έχουν εγγραφεί.

- user_id: bigint(20). Αναφορά σε αναγνωριστικό χρήστη.
- tag_id: bigint(20). Αναφορά σε αναγνωριστικό ανακοίνωσης.

Στον συγκεκριμένο πίνακα το κύριο κλειδί αποτελείται και από τα δύο πεδία.

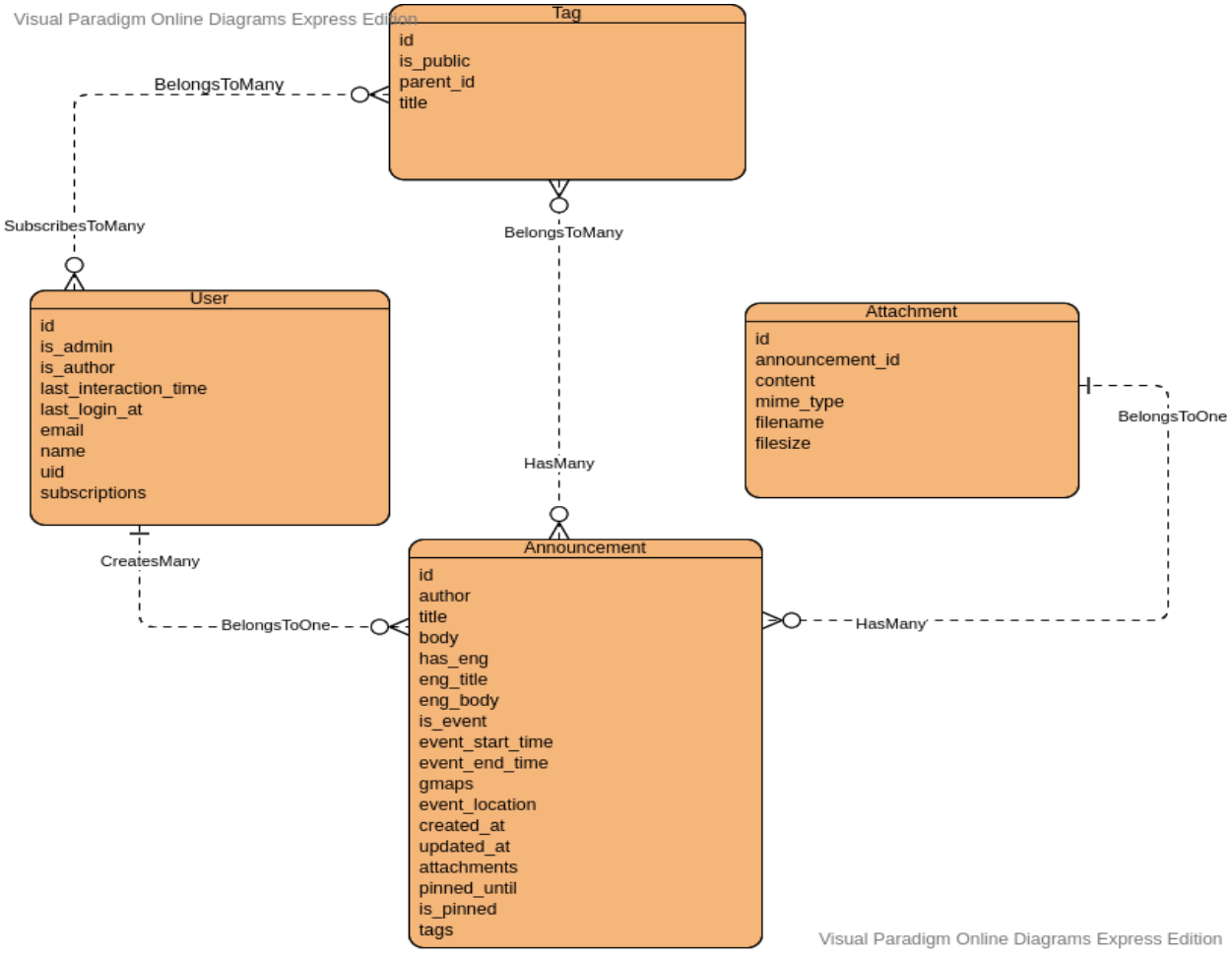
Πίνακες Laravel Passport

Το Laravel Passport με τη σειρά του περιέχει κάποιους πίνακες, από τους οποίους άξιος αναφοράς για τη συγκεκριμένη εφαρμογή είναι ο παρακάτω:

Oauth_Access_Tokens

- id: varchar(100). Αποτελεί το αναγνωριστικό του κάθε τεκμηρίου πρόσβασης. Είναι το κύριο κλειδί του πίνακα.
- user_id: bigint(20). Αναφορά στο αναγνωριστικό του χρήστη τον οποίο αφορά το εκάστοτε τεκμήριο.
- client_id: bigint(20). Αναφορά στο αναγνωριστικό του πελάτη που έχει εκδώσει το τεκμήριο.
- name: varchar(255). Το όνομα του παραπάνω πελάτη.
- scopes: text. Αφορά του πεδίο εφαρμογής του τεκμηρίου.
- revoked: boolean. Ορίζει εάν το τεκμήριο είναι ενεργό ή ανενεργό.
- created_at: timestamp. Ορίζει τη χρονική στιγμή δημιουργίας του τεκμηρίου.
- updated_at: timestamp. Ορίζει τη χρονική στιγμή ανανέωσης του τεκμηρίου.
- expires_at: timestamp. Ορίζει τη χρονική στιγμή λήξης του τεκμηρίου.

3.3 Σχέσεις μεταξύ αντικειμένων



Εικόνα 22 Διάγραμμα σχέσεων οντοτήτων

Ανάλυση

Όπως βλέπουμε και την παραπάνω εικόνα, τα κύρια συστατικά της εφαρμογής είναι 4: η Ανακοίνωση (Announcement), η Ετικέτα (Tag), η Επισύναψη (Attachment) και ο Χρήστης (User).

- Συσχέτιση Ανακοίνωσης – Επισύναψης: Κάθε επισύναψη αντιστοιχίζεται με μία ανακοίνωση. Κάθε ανακοίνωση μπορεί να περιέχει πολλές επισυνάψεις.
- Συσχέτιση Ανακοίνωσης – Ετικέτας: Κάθε ετικέτα μπορεί να αντιστοιχιστεί με πολλές ανακοινώσεις. Ομοίως, κάθε ανακοίνωση μπορεί να περιέχει πολλές ετικέτες.
- Συσχέτιση Ανακοίνωσης – Χρήστη: Κάθε χρήστης που έχει σημανθεί ως συντάκτης, μπορεί να δημιουργήσει πολλές ανακοινώσεις. Κάθε ανακοίνωση μπορεί να ανήκει μόνο σε έναν χρήστη.
- Συσχέτιση Χρήστη – Ετικέτας: Κάθε χρήστης μπορεί να εγγραφεί σε πολλές ετικέτες. Αντίστοιχα, πολλές ετικέτες μπορούν να ανήκουν σε έναν χρήστη. Κάθε χρήστη που έχει σημανθεί ως διαχειριστής, μπορεί να δημιουργήσει πολλές ετικέτες.

3.4 Ανάλυση ενδιάμεσου λογισμικού (middleware)

Παραπάνω αναλύθηκε τι είναι το ενδιάμεσο λογισμικό και ποια η χρησιμότητά του στις εφαρμογές. Στις επόμενες παραγράφους θα γίνει ανάλυση της χρήσης του ενδιάμεσου λογισμικού στην παρούσα εφαρμογή. Όπως αναφέρθηκε και παραπάνω, στο Laravel υποστηρίζεται ενδιάμεσο λογισμικό τόσο για το front-end όσο και για το back-end (το API). Ελλείψει εξωτερικού δρομολογητή κίνησης (θα γίνει αναφορά για το συγκεκριμένο στο κεφάλαιο με τις προτάσεις βελτίωσης), έγινε χρήση αυτού που εγγενώς περιλαμβάνεται στο Laravel.

Web Middleware (front-end)

Πέρα από τα middleware που περιλαμβάνονται εγγενώς, έχει γίνει προσθήκη ορισμένων για την καλύτερη απόδοση της εφαρμογής και για την ασφάλειά της από άτομα που δε διαθέτουν την επαρκή εξουσιοδότηση.

- Έλεγχος ύπαρξης ετικέτας/ανακοίνωσης: Σε αυτό το ενδιάμεσο λογισμικό ελέγχεται εάν υπάρχει ετικέτα/ανακοίνωση με το αναγνωριστικό που έχει δοθεί. Σε περίπτωση που δεν υπάρχει, επιστρέφεται σελίδα λάθους με κωδικό κατάστασης 404 (Δε βρέθηκε). Πρόκειται για δύο ξεχωριστούς ενδιάμεσους.
- Έλεγχος αναγνωριστικού: Ελέγχεται εάν οι χαρακτήρες που έχουν δοθεί αποτελούν όντως αριθμό. Σε περίπτωση που δεν αποτελούν, επιστρέφεται στον χρήστη σελίδα με κωδικό κατάστασης 400 (Λάθος αίτημα).
- Έλεγχος τύπου χρήστη: Ελέγχεται εάν ο χρήστης είναι όντως συντάκτης κατά την προσπάθεια για προσπέλαση της σελίδας δημιουργίας ανακοίνωσης. Σε περίπτωση που δεν είναι, επιστρέφεται σελίδα με κωδικό κατάστασης 401 (Μη εξουσιοδότηση).
- Έλεγχος αναγνωριστικού στην επεξεργασία: Σε συνέχεια του παραπάνω ενδιάμεσου, υπάρχει έλεγχος εάν ο χρήστης που προσπαθεί να επεξεργαστεί την ανακοίνωση, είναι ο συγγραφέας αυτής ή είναι διαχειριστής. Εάν δεν ισχύει κάποια από τις δύο συνθήκες, επιστρέφεται σελίδα με κωδικό σφάλματος 401.

API Middleware (back-end)

Αντίστοιχα με τα παραπάνω ενδιάμεσα λογισμικά για το front-end, υπάρχουν για το back-end. Στην προκειμένη περίπτωση, όμως, δεν επιστρέφεται κάποια σελίδα, αλλά ένα μήνυμα με τους αντίστοιχους κωδικούς κατάστασης σε μορφή JSON.

Web και API Middleware

Βάσει της ανάλυσης των απαιτήσεων της εφαρμογής, υπήρχε η ανάγκη όλες οι ανακοινώσεις να φαίνονται όταν οι χρήστες εισέρχονται στην εφαρμογή από IP διευθύνσεις του τοπικού δικτύου του τμήματος. Για να πραγματοποιηθεί το συγκεκριμένο, έχει γραφτεί ενδιάμεσο λογισμικό που εκτελεί έλεγχο της IP και ορίζει μία μεταβλητή στη συνεδρία (session) ώστε να είναι εφικτό για τον controller να ελέγχει εάν η μεταβλητή αυτή είναι αληθής ή ψευδής. Στην περίπτωση που είναι αληθής, εμφανίζει όλες τις ανακοινώσεις στον χρήστη. Το συγκεκριμένο ενδιάμεσο λογισμικό δεν εκτελεί καμία άλλη ενέργεια ούτε παρέχει κάποια άλλη δυνατότητα στον χρήστη.

3.5 Τρόπος λειτουργίας

Όπως έχει αναφερθεί και παραπάνω, η εφαρμογή χωρίζεται σε δύο κομμάτια, το front-end και το back-end, εξαιρετικά συνδεδεμένα μεταξύ τους στην προκειμένη υλοποίηση. Ο λόγος είναι πως δεν υπάρχει εξωτερικός δρομολογητής κίνησης και όλη τη δρομολόγηση για το front-end την έχει αναλάβει ο δρομολογητής που παρέχει το Laravel εγγενώς.

Όταν έρχεται ένα αίτημα, αναλαμβάνει ο δρομολογητής του Laravel να το «κατευθύνει» στο σωστό μέρος. Εκεί, μέσω των views, που αναλύθηκαν παραπάνω, γίνεται φόρτωση ενός component του Vue. Αυτό το component είναι υπεύθυνο να κάνει τα απαραίτητα αιτήματα στο API της εφαρμογής και, στη συνέχεια, να φορτώσει άλλα compoenent και να εμφανίσει τα δεδομένα με τη μορφή που έχουν δηλωθεί στο πρότυπο (template). Αυτή η διαδικασία συμβαίνει ανεξάρτητα από το εάν ο χρήστης είναι συνδεδεμένος ή όχι. Επίσης, είναι ανεξάρτητη του εάν είναι απλός χρήστης (φοιτητής), συντάκτης ανακοινώσεων ή διαχειριστής. Αναλόγως του βαθμού του στη βάση της εφαρμογής, γίνονται και τα αντίστοιχα αιτήματα. Όπως είναι προφανές, όσο μεγαλύτερος ο βαθμός του χρήστη, τόσο μεγαλύτερη η πρόσβαση που έχει.

Το Vue.js απαιτεί έναν σχετικά σύγχρονο φυλλομετρητή, καθώς ο τρόπος που κάνει mount τα components στο DOM του HTML, καθώς και η δημιουργία του εσωτερικού virtual DOM, απαιτεί σύγχρονα εργαλεία που δεν υπάρχουν σε παλαιότερους φυλλομετρητές.

3.6 Επίλογος

Σε αυτό το κεφάλαιο είδαμε το σχήμα της βάσης και τους συσχετισμούς μεταξύ των μοντέλων της εφαρμογής. Τέλος, αναλύθηκαν τα ενδιάμεσα λογισμικά, ο τρόπος χρήσης και λειτουργίας τους.

Κεφάλαιο 4ο: Περιγραφή της υλοποίησης

4.1 Περιγραφή της υλοποίησης

Στο παρόν κεφάλαιο, θα γίνει μία αποδόμηση όλων των στοιχείων της εφαρμογής, καθώς και μία ανάλυση όλων των σημείων του API και της διαδικτυακής εφαρμογής χωρίς την παρουσίαση κώδικα. Η ανάλυση θα μείνει στη λογική πίσω από την υλοποίηση. Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο, για τη δρομολόγηση των αιτημάτων χρησιμοποιούνται τα αρχεία web.php και api.php (και τα δύο στο φάκελο routes) για τη διαδικτυακή εφαρμογή και το API αντίστοιχα.

Η αρχική σελίδα

Όταν γίνεται ένα αίτημα λήψης της σελίδας http://aboard.iee.ihu.gr/, γίνεται ανακατεύθυνση στη διαδρομή http://aboard.iee.ihu.gr/announcements. Εκεί, γίνεται φόρτωση του κορμού της σελίδας από το αρχείο blade layout/main και φορτώνεται στο κεντρικό τμήμα της εφαρμογής το component που αφορά όλες τις ανακοινώσεις.

Ο βασικός κορμός της εφαρμογής αποτελείται από τα HTML tags (html, head κ.λπ.) και από το body element. Το body element ορίζει ένα div πάνω στο οποίο γίνεται mount το Vue instance (υπάρχει αντίστοιχη ανάλυση στο κεφάλαιο 2, στον τομέα του Vue.js). Μέσα σε αυτό το div, γίνεται φόρτωση των components της γραμμής πλοήγησης (κεφαλίδα-header), του υποσέλιδου (footer) και ορίζεται ένα κομμάτι πάνω στο οποίο θα γίνονται mount τα διάφορα components.

Μέσα στο component των ανακοινώσεων, γίνεται φόρτωση των components για τον τίτλο της σελίδας, για την αναζήτηση, για τη σελιδοποίηση, τα κουμπιά προσθήκης ανακοίνωσης/διαχείρισης και ενός component ανακοίνωσης. Εκεί γίνεται η φόρτωση των ανακοινώσεων από το API, και μέσω ενός βρόχου επανάληψης, εμφάνισή τους στη σελίδα.

Το component αναζήτησης περιλαμβάνει την μπάρα αναζήτησης και τον χώρο εμφάνισης των σημαντικών ανακοινώσεων.

Εάν ο χρήστης είναι συνδεδεμένος και είναι φοιτητής, απλώς αλλάζει η γραμμή πλοήγησης και, αντί για κουμπί σύνδεσης, περιέχει δύο κουμπιά. Το ένα είναι για την εμφάνιση των πρόσφατων ειδοποιήσεων του χρήστη και το άλλο για τη διαχείριση των στοιχείων που του επιτρέπει η εφαρμογή. Εάν ο χρήστης είναι συνδεδεμένος και είναι συντάκτης, εμφανίζεται, επίσης, ένα κουμπί που τον ανακατευθύνει στη σελίδα δημιουργίας ανακοίνωσης, ενώ εάν είναι συνδεδεμένος και είναι admin, εμφανίζεται και ένα κουμπί το οποίο ανοίγει το παράθυρο διαχείρισης των ετικετών.

Για ευκολία καθώς και για μη επανάληψη του domain name της εφαρμογής, από αυτό το σημείο και για το υπόλοιπο του κεφαλαίου, θα θεωρήσουμε πως, όταν αναφέρεται /example, θα εννοείται http://aboard.iee.ihu.gr/example. Επίσης, θα θεωρήσουμε πως σε κάθε διαδρομή θα γίνεται φόρτωση του layout/main που αποτελεί τον βασικό κορμό της εφαρμογής.

Το component ανακοίνωσης

Στην εφαρμογή υπάρχουν, ουσιαστικά, δύο components για τις ανακοινώσεις. Το ένα είναι αυτό που φαίνεται στην αρχική σελίδα, ενώ το άλλο είναι αυτό που φορτώνεται όταν ο χρήστης ζητήσει συγκεκριμένη ανακοίνωση με αναγνωριστικό id στη διαδρομή /announcements/id. Και στις δύο περιπτώσεις, φορτώνονται συγκεκριμένα components (π.χ. τίτλος, περιεχόμενο, ετικέτες κ.ά.) και εμφανίζονται στον χρήστη.

Το component δημιουργίας/επεξεργασίας ανακοίνωσης

Για τη δημιουργία και την επεξεργασία μίας ανακοίνωσης φορτώνεται ένα κοινό component. Η μεν δημιουργία είναι προσπελάσιμη από τη διαδρομή /announcements/create, ενώ η επεξεργασία από τη διαδρομή /announcements/id/edit. Ενδιάμεσο λογισμικό ελέγχει εάν ο χρήστης μπορεί να δημιουργήσει ανακοίνωση στην πρώτη περίπτωση, ενώ στη δεύτερη ελέγχει εάν είναι ο συντάκτης της ανακοίνωσης ή είναι διαχειριστής. Το κοινό component κάνει χρήση της δυναμικής δέσμευσης εισαγωγής (dynamic input binding) καθώς και του v-model. Εδώ φαίνεται και η επαναχρησιμοποίηση components, ένα από τα βασικά χαρακτηριστικά του Vue.js. Από εδώ γίνεται, τέλος, και η διαγραφή της ανακοίνωσης.

Το σημείο δημιουργίας/επεξεργασίας των ανακοινώσεων στο API

Οι μέθοδοι δημιουργίας και επεξεργασίας ανακοινώσεων στον controller είναι οι store και update. Κατά τη δημιουργία μίας ανακοίνωσης, ελέγχονται οι περιορισμοί που έχουν υποβληθεί αναφορικά με τον τύπο των δεδομένων και, εφόσον είναι επιτυχείς όλοι οι έλεγχοι, το back-end προχωρά στη δημιουργία της ανακοίνωσης. Ακόμη, μετά την επιτυχή δημιουργία, υψώνεται ένα Laravel event, το οποίο δημιουργεί μία ειδοποίηση για κάθε χρήστη εγγεγραμμένο στις ετικέτες της, έτσι ώστε αυτή να εμφανιστεί την επόμενη φορά που θα πραγματοποιήσει είσοδο στην εφαρμογή. Από την άλλη, στην περίπτωση της επεξεργασίας, αφού ελέγχεται πάλι πως όσα έχει στείλει ο χρήστης είναι σωστά, γίνεται αντικατάσταση/συμπλήρωση των δεδομένων και, στη συνέχεια, αποθήκευση. Σε αυτήν την περίπτωση δεν υπάρχει κάποιο Laravel event.

Η σελίδα των εκδηλώσεων

Στη διαδρομή /events φορτώνεται η σελίδα με τις εκδηλώσεις. Πέραν των component για τον τίτλο της σελίδας και τη σελιδοποίηση, φορτώνεται και ένας βρόχος που περιλαμβάνει ένα component εκδήλωσης (ομοίως με τις ανακοινώσεις). Όταν ο χρήστης κάνει click σε μία εκδήλωση, μεταφέρεται στην αντίστοιχη σελίδα της ανακοίνωσης. Οι εκδηλώσεις, σε αντίθεση με τις ανακοινώσεις, φορτώνονται όλες, με περιορισμένες πληροφορίες, ενώ όταν γίνεται click σε μία, αναλαμβάνει το ενδιάμεσο λογισμικό να επιλέξει εάν ο χρήστης μπορεί να πραγματοποιήσει θέαση της ανακοίνωσης.

Η σελίδα με τους συντάκτες

Για προσπέλαση της σελίδας με τους συντάκτες απαιτείται σύνδεση με την εφαρμογή. Γίνεται φόρτωση όλων των συντακτών από το API και εμφανίζεται στον χρήστη μία καρτέλα για τον καθένα, πάνω στην οποία μπορεί να κάνει click και να του εμφανιστούν όλες οι ανακοινώσεις που έχει γράψει ο εκάστοτε συντάκτης. Αξιοσημείωτο είναι το γεγονός πως, καθώς η εφαρμογή διατηρεί δική της βάση η οποία δε συγχρονίζεται με τον LDAP server του τμήματος, για να εμφανιστεί ένας συντάκτης, θα πρέπει πρώτα να πραγματοποιήσει σύνδεση με την εφαρμογή και να γίνει ο έλεγχος πως πληροί την προϋπόθεση να οριστεί ως συντάκτης. Για παράδειγμα, μπορεί ο χρήστης Χ, που είναι φοιτητής, να ξέρει πως εκεί θα πρέπει να βρει τον χρήστη Ψ, που είναι καθηγητής, οπότε θα έπρεπε να εμφανίζεται, αλλά εάν ο δεύτερος δεν έχει πραγματοποιήσει σύνδεση, δε θα εμφανίζεται. Αυτός είναι ένας περιορισμός που υψώνεται από την πρόσβαση που έχει η ίδια η εφαρμογή στον LDAP server.

Η σελίδα με την τεκμηρίωση

Στη διαδρομή /docs υπάρχει μία σελίδα με όλη την τεκμηρίωση (documentation) όσον αφορά το API. Για θέαση της σελίδας είναι απαραίτητη η είσοδος στην εφαρμογή. Σε αυτήν τη σελίδα υπάρχει υλικό που αφορά όλα τα σημεία (endpoints) του API με παραδείγματα των απαντήσεων καθώς και ανάλυση των μοντέλων της εφαρμογής.

Αναζήτηση βάσει ετικέτας ή συντάκτη

Η αναζήτηση βάσει ετικέτας ή συντάκτη πραγματοποιείται στις διαδρομές /search/tag/id ή /search/author/id αντίστοιχα. Εκεί, αφού το αίτημα περάσει από τα ενδιάμεσα λογισμικά, η σελίδα πραγματοποιεί αναζήτηση βάσει του αντίστοιχου αναγνωριστικού που έχει δοθεί. Τα ενδιάμεσα λογισμικά περιλαμβάνουν τον έλεγχο εάν αυτό που έχει δοθεί ως id είναι αριθμός και στις δύο περιπτώσεις, ενώ στην περίπτωση των ετικετών, ελέγχεται και αν υπάρχει ετικέτα με το αναγνωριστικό που έχει ληφθεί.

Προσαρμοσμένη αναζήτηση

Το component της προσαρμοσμένης αναζήτησης βρίσκεται στην αρχική οθόνη. Σε αυτό ο χρήστης μπορεί να εισάγει λέξεις ή φράσεις ώστε να πραγματοποιήσει αναζήτηση στη βάση της εφαρμογής. Η αναζήτηση γίνεται μόνο στο πεδίο του τίτλου της ανακοίνωσης (τόσο στον ελληνικό όσο και στον αγγλικό) και, αφού γίνει το αίτημα στο API, εμφανίζεται μία σελίδα με τα αποτελέσματα. Ο λόγος που έχει επιλεγεί να γίνεται αναζήτηση μόνο στον τίτλο είναι το γεγονός πως, καθώς τυχόν εικόνες που περιλαμβάνονται στην ανακοίνωση είναι σε μορφή base64 κωδικοποίησης, υπάρχει η περίπτωση τα επιστρεφόμενα αποτελέσματα να μη συνάδουν με τους όρους αναζήτησης του χρήστη.

Η σελίδα εισόδου

Η σελίδα της εισόδου περιλαμβάνει ένα πλαίσιο εισαγωγής στο οποίο υπάρχουν δύο πεδία: το πεδίο για το όνομα χρήστη και το πεδίο για τον κωδικό του χρήστη. Μετά την εισαγωγή τους και το click από τον χρήστη το κουμπί εισόδου, πραγματοποιείται ένα αίτημα στο API στο σημείο /api/auth/login. Στην περίπτωση επιτυχημένης εισαγωγής στοιχείων, γίνεται αποθήκευση του τεκμηρίου πρόσβασης στην προσωρινή μνήμη του φυλλομετρητή και, έπειτα, ανακατεύθυνση στην αρχική σελίδα. Εκεί πλέον, γίνεται εκ νέου αίτημα στο API, που αυτήν τη φορά περιλαμβάνει το τεκμήριο που αποθηκεύτηκε, και ως εκ τούτου, το API επιστρέφει όλες τις ανακοινώσεις με χρονολογική σειρά, ώστε να πραγματοποιηθεί η διαδικασία που περιεγράφηκε πιο πάνω. Στην περίπτωση που εισήχθησαν λάθος στοιχεία, επιστρέφεται από το API μήνυμα λάθους που αντικατοπτρίζεται και στο front-end με αντίστοιχο μήνυμα.

Η γραμμή πλοήγησης

Η γραμμή πλοήγησης για έναν μη συνδεδεμένο χρήστη περιλαμβάνει, πέρα από τα κουμπιά ανακατεύθυνσης στην αρχική, τη σελίδα με τις εκδηλώσεις και τους συντάκτες, ένα κουμπί ανακατεύθυνσης στη σελίδα εισόδου. Εάν ο χρήστης έχει πραγματοποιήσει είσοδο, το τελευταίο αντικαθίσταται από δύο άλλα κουμπιά. Το ένα περιλαμβάνει τις δέκα τελευταίες ειδοποιήσεις του χρήστη, ενώ το άλλο περιέχει κάποιες πληροφορίες του χρήστη και ένα μενού ανακατεύθυνσης στη σελίδα με τις προτιμήσεις.

Η σελίδα προτιμήσεων του χρήστη

Η σελίδα προτιμήσεων του χρήστη περιλαμβάνει δύο πεδία. Το ένα είναι το πεδίο με τη δραστηριότητα του χρήστη και το άλλο με τις ετικέτες στις οποίες έχει πραγματοποιήσει εγγραφή. Στο πεδίο της δραστηριότητας περιλαμβάνεται ένας πίνακας που περιέχει αναλυτικά όλες τις ειδοποιήσεις που αφορούν τον χρήστη. Αυτές περιλαμβάνουν τις εισόδους που έχει πραγματοποιήσει ο χρήστης στην εφαρμογή, καθώς και τις ανακοινώσεις που έχουν γραφτεί και περιλαμβάνουν ετικέτες στις οποίες ο χρήστης έχει πραγματοποιήσει εγγραφή. Το δεύτερο πεδίο περιλαμβάνει έναν πίνακα με όλους τους

τίτλους των ετικετών και μία λίστα με όλες τις ετικέτες στις οποίες έχει πραγματοποιήσει εγγραφή ο χρήστης. Η δυνατότητα αυτή δίνεται σε όλους τους εγγεγραμμένους χρήστες ανεξαρτήτως ρόλου.

Το RSS Feed

Στη διαδρομή /feed υπάρχει το RSS Feed των ανακοινώσεων. Σε αυτήν τη σελίδα περιλαμβάνονται όλες οι ανακοινώσεις των τελευταίων επτά ημερών που περιέχουν δημόσιες ετικέτες καθώς και όλες έχουν σημανθεί από τους συντάκτες ως σημαντικές. Η επιλογή των επτά ημερών είναι πλήρως διαμορφώσιμη από το αρχείο .env του Laravel και μπορεί να αλλάξει πολύ εύκολα. Τέλος, αξίζει να σημειωθεί πως η επιλογή των κριτηρίων για την εμφάνιση των ανακοινώσεων ανήκει στον επιτηρητή καθηγητή. Φυσικά, και το συγκεκριμένο είναι εύκολα παραμετροποιήσιμο μέσω της μεθόδου ανάκτησης των ανακοινώσεων.

Οι σελίδες σφαλμάτων

Όπως έχει αναφερθεί και παραπάνω, στην εφαρμογή υπάρχουν ενδιάμεσα λογισμικά που ελέγχουν διάφορες συνθήκες αναφορικά με τη λειτουργικότητά της. Εάν εντοπιστεί κάποιο σφάλμα στη διαδικτυακή εφαρμογή, γίνεται ανακατεύθυνση στην αντίστοιχη σελίδα σφάλματος. Για παράδειγμα, για το σφάλμα 404 (δε βρέθηκε), υπάρχει αντίστοιχη σελίδα στη διαδρομή /errors/404 που το εμφανίζει.

Δημιουργία/Επεξεργασία/Διαγραφή ετικέτας

Η διαχείριση των ετικετών αφορά αποκλειστικά τους χρήστες της εφαρμογής που έχουν σημανθεί ως διαχειριστές. Η διαχείριση πραγματοποιείται μέσω ενός αναδυόμενου παραθύρου (modal) στο οποίο υπάρχει μία λίστα με όλες τις ετικέτες, καθώς και ένα κουμπί προσθήκης νέας. Επίσης, υπάρχουν κουμπιά για επεξεργασία και διαγραφή μίας ετικέτας. Πριν γίνει προσπάθεια για αποθήκευση ή αλλαγή ετικέτας, υπάρχουν δικλείδες που ελέγχουν αρχικά εάν ο χρήστης μπορεί να προσθέσει ετικέτα (δηλαδή εάν είναι διαχειριστής) και αν τα δεδομένα που έχουν σταλεί στο API είναι σωστά. Μετά την επικύρωση, το αίτημα προχωρά στον controller, ο οποίος είναι υπεύθυνος για τη διαχείριση. Σε περίπτωση σφάλματος, επιστρέφεται στον χρήστη μήνυμα σε μορφή JSON με τον κωδικό του σφάλματος και υπάρχει αντίστοιχη ενημέρωση στο front-end.

Προεπιλεγμένη διαδρομή για λάθος αιτήματα

Τόσο το front-end όσο και το back-end ορίζουν συγκεκριμένες διαδρομές που είναι προσπελάσιμες ανεξάρτητα από το είδος του χρήστη που προσπαθεί να πραγματοποιήσει την όποια ενέργεια. Σε περίπτωση που ζητηθεί διαδρομή που δεν υπάρχει, επιστρέφεται στον χρήστη σελίδα ή απάντηση σε μορφή JSON με κωδικό σφάλματος 404, που τον ενημερώνει πως δεν υπάρχει το σημείο που ζητήθηκε.

Ο κώδικας της εφαρμογής

Ο κώδικας της εφαρμογής υπάρχει σε ένα αποθετήριο (repository) στη διαδικτυακή πλατφόρμα φιλοξενίας εφαρμογών ανοιχτού λογισμικού Github και μπορεί να βρεθεί στον ακόλουθο σύνδεσμο: https://github.com/nickcn/aboard.iee.ihu.gr. Το αποθετήριο τη στιγμή της δημιουργίας του παρόντος κειμένου είναι ιδιωτικό, που σημαίνει πως μόνο οι χρήστες που έχουν ορισθεί από τον γράφον έχουν τη δυνατότητα προσπέλασής του. Ο επιτηρητής καθηγητής έχει δικαίωμα προσπέλασης και αλλαγών. Στο αποθετήριο, πέραν του κώδικα της εφαρμογής, υπάρχει και ένα αρχείο που περιέχει κατευθυντήριες οδηγίες για εγκατάσταση της εφαρμογής σε ένα δοκιμαστικό περιβάλλον σε Debian-based λειτουργικά συστήματα (π.χ. Debian, Ubuntu κ.λπ). Για μεταφορά του κώδικα σε παραγωγικό περιβάλλον ενδεχομένως να χρειαστούν κάποιες περαιτέρω αλλαγές αναφορικά με την ασφάλεια του λειτουργικού

συστήματος που, όμως, δεν ξεφεύγουν από γενικές καλές πρακτικές για τη διαχείριση ενός Linux server, αλλά δεν αποτελούν αντικείμενο της παρούσης ΠΕ.

4.2 Επίλογος

Σε αυτό το κεφάλαιο έγινε μία αποδόμηση της εφαρμογής και των επιμέρους κομματιών της. Αναλύθηκαν όλες οι λειτουργίες και οι επιμέρους σελίδες που περιέχει η εφαρμογή.

Κεφάλαιο 5ο: Περιγραφή της χρήσης

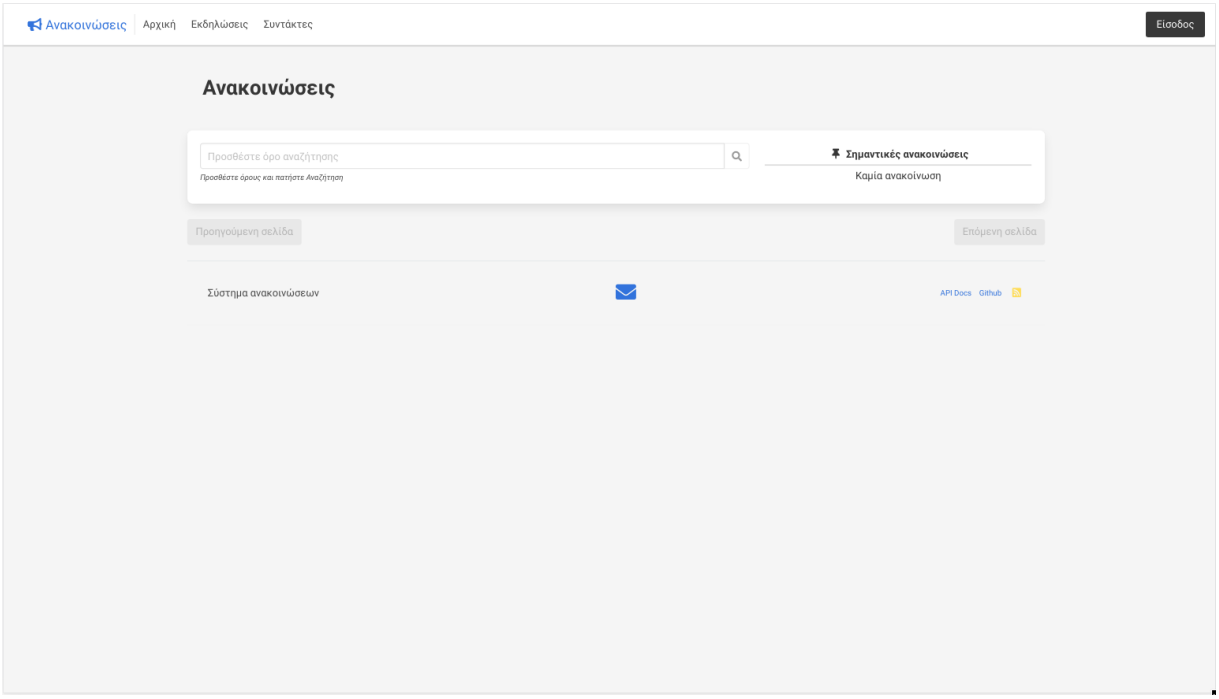
5.1 Περιγραφή της χρήσης

Οι παρακάτω δοκιμές και αναλύσεις θα γίνουν από τον λογαριασμό του γράφοντα στην εφαρμογή. Για να υπάρχει μία πιο σφαιρική όψη των λειτουργιών, θα θεωρηθεί πως ο λογαριασμός αποτελεί λογαριασμό διαχειριστή και σε κάθε διαφορετική λειτουργία θα επισημαίνονται οι διαφορές που υπάρχουν με τους απλούς λογαριασμούς.

Ακόμη, σε σημεία που δεν ακολουθήθηκε η πεπατημένη οδός του Laravel («the Laravel way», όπως συνηθίζεται να αποκαλείται από προγραμματιστές ο τρόπος με τον οποίο το Laravel επιλέγει να πραγματοποιεί ορισμένες ενέργειες) κατά τη δημιουργία των σημείων του API ή της διαδικτυακής εφαρμογής, θα υπάρχει ανάλυση κώδικα και περαιτέρω επεξήγησή του.

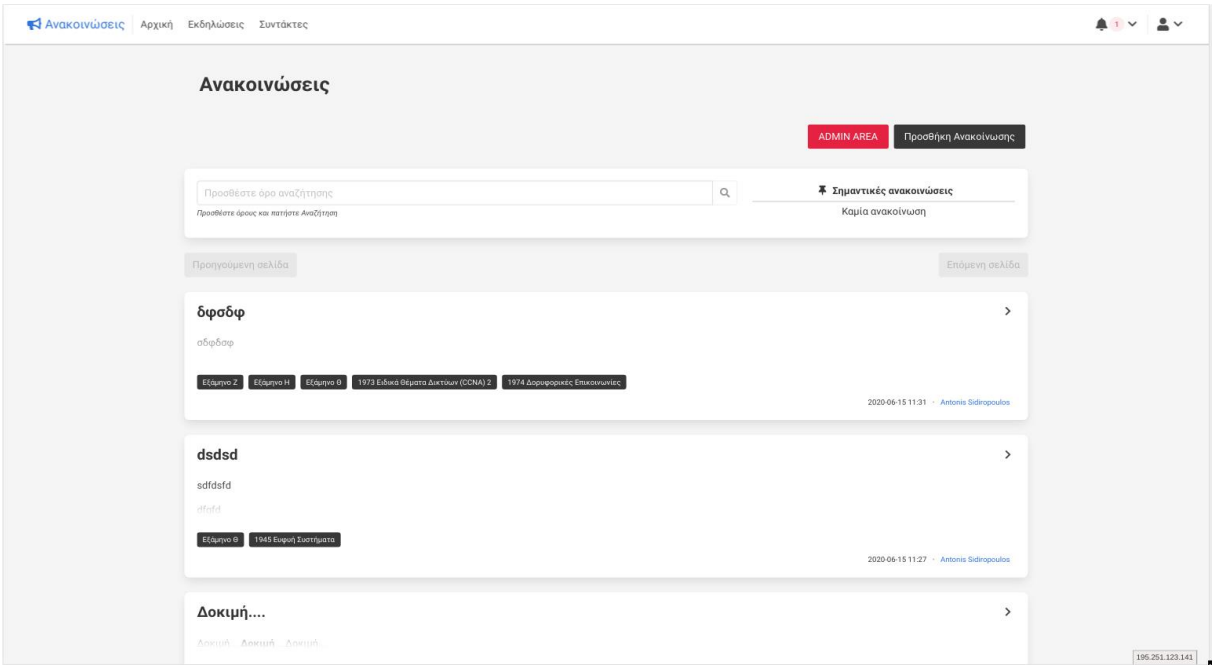
Τέλος, να σημειωθεί πως, για τη λήψη των screenshots, χρησιμοποιήθηκε η λειτουργία του φυλλομετρητή Google Chrome που επιτρέπει την αλλαγή της ανάλυσης οθόνης, οπότε το τελικό αποτέλεσμα ενδέχεται να διαφέρει μερικώς από αυτό που θα συναντήσει ένας χρήστης με διαφορετική ανάλυση.

Αρχική σελίδα



Εικόνα 23 Αρχική οθόνη μη αυθεντικοποιημένου χρήστη

Η παραπάνω εικόνα αποτελεί την αρχική οθόνη ενός χρήστη που δεν έχει πραγματοποιήσει σύνδεση στην εφαρμογή, η οποία αλλάζει στην επόμενη όταν ένας χρήστης πραγματοποιεί σύνδεση:

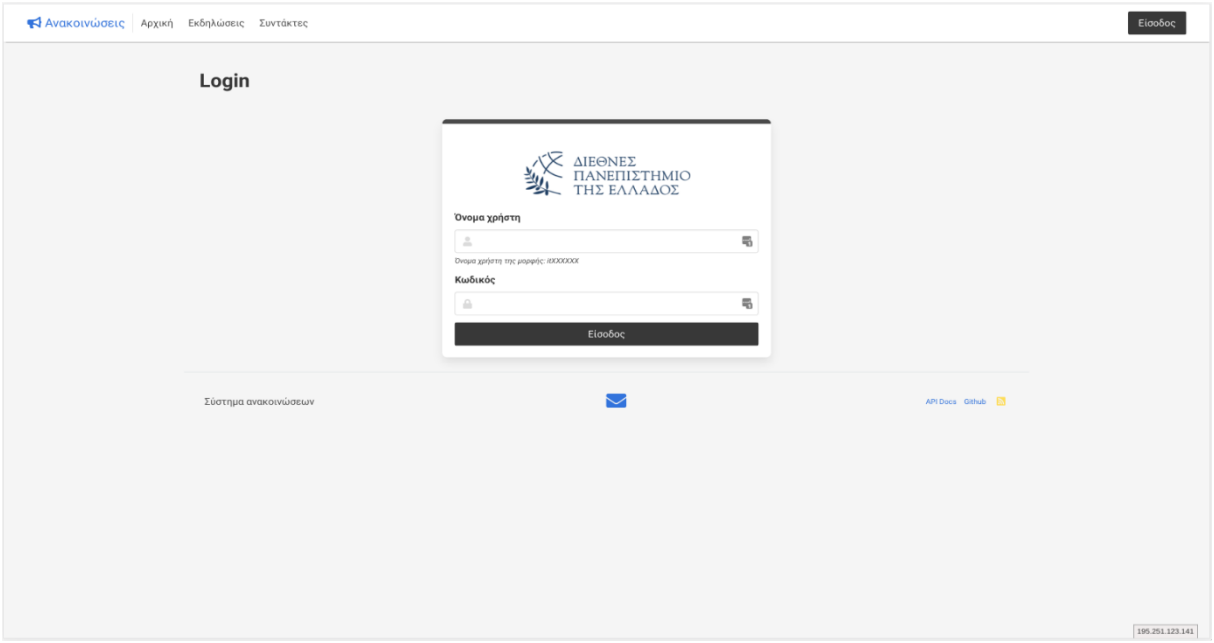


Εικόνα 24 Αρχική οθόνη αυθεντικοποιημένου χρήστη

Τα δύο πάνω κουμπιά με όνομα «ADMIN AREA» και «Προσθήκη Ανακοίνωσης» εμφανίζονται στους χρήστες που αποτελούν διαχειριστές και συντάκτες αντίστοιχα. Στην προκειμένη περίπτωση, ο λογαριασμός αποτελεί τόσο συντάκτη όσο και διαχειριστή.

Πάνω δεξιά, επίσης, βλέπουμε τα κουμπιά για την εμφάνιση των δέκα τελευταίων ειδοποιήσεων του χρήστη, καθώς και για τη διαχείριση των προτιμήσεών του.

Οθόνη εισόδου



Εικόνα 25 Οθόνη εισόδου

Η οθόνη εισόδου αποτελεί μία τυπική φόρμα με δύο πεδία: το όνομα χρήστη και κωδικό χρήστη. Είναι και τα δύο απαιτούμενα για να πραγματοποιηθεί σύνδεση. Ιδιαίτερη μνεία αξίζει να γίνει στον τρόπο που ο χρήστης αυθεντικοποιείται με το σύστημα, καθώς ξεφεύγει από τον προκαθορισμένο τρόπο που χρησιμοποιεί το Laravel ή το Laravel Passport. Και στις δύο περιπτώσεις χρησιμοποιείται το πεδίο email. Στην περίπτωση της εφαρμογής που περιγράφεται, σκοπός είναι να χρησιμοποιηθεί το πεδίο uid που επιστρέφεται από τον LDAP server του τμήματος.

Αυθεντικοποίηση χρήστη

Όπως αναφέρθηκε, σκοπός ήταν η εφαρμογή να χρησιμοποιεί προσαρμοσμένο τρόπο αυθεντικοποίησης του χρήστη. Για να επιτευχθεί αυτό, χρησιμοποιήθηκε η μέθοδος που παρέχει το Laravel Passport με όνομα findAndValidateForPassport που δέχεται δύο ορίσματα: το όνομα χρήστη και τον κωδικό. Όπως φαίνεται στις παρακάτω εικόνες, αρχικά, ορίζονται σε έναν πίνακα PHP κάποιες τιμές που αφορούν τον LDAP server, οι οποίες τιμές λαμβάνονται από τα αρχεία configuration που διαθέτει το Laravel. Στη συνέχεια, πραγματοποιείται σύνδεση στον LDAP server και ορίζονται κάποιες επιλογές σύνδεσης. Έπειτα, για κάθε DN που έχει οριστεί, πραγματοποιείται, σε επαναληπτικό βρόχο, προσπάθεια για σύνδεση του χρήστη με τα στοιχεία που έχει λάβει η εφαρμογή. Σε περίπτωση που δεν είναι εφικτή η σύνδεση, επιστρέφεται μήνυμα λάθους με κωδικό 401. Αφού έχει περάσει αυτό το στάδιο επιτυχώς, η εφαρμογή προσπαθεί να διαβάσει τα στοιχεία που έχουν οριστεί αναφορικά με τον χρήστη. Εάν ο χρήστης ανήκει στο group «staff», ορίζεται ως συντάκτης. Στη συνέχεια, γίνεται έλεγχος εάν υπάρχει ήδη ο χρήστης στη βάση της εφαρμογής. Εάν δεν υπάρχει, δημιουργείται με τις ιδιότητες που έχουμε λάβει από τον LDAP server. Εάν υπάρχει, γίνεται ενημέρωσή του. Τέλος, επιστρέφεται από τη μέθοδο η ιδιότητα $attributes, η οποία δίνει τη δυνατότητα στο Laravel να πραγματοποιήσει αναζήτηση μέσω του αναγνωριστικού id του χρήστη στη βάση.

Παρακάτω, όλος ο κώδικας της μεθόδου, με σχόλια:

```php
public function findAndValidateForPassport($username, $password)
{
    // Get values for LDAP server, try to connect and set some options
    $ldapconfig['host'] = config('services.ldap.host');
    $ldapconfig['port'] = config('services.ldap.port');
    $ldapconfig['basedn'] = config('services.ldap.base_dn');
    $ldapconfig['usersdn'] = explode(',', config('services.ldap.users_dn'));

    $ds = ldap_connect($ldapconfig['host'], $ldapconfig['port']);
    ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);
    ldap_set_option($ds, LDAP_OPT_REFERRALS, 0);
    ldap_set_option($ds, LDAP_OPT_NETWORK_TIMEOUT, 10);

    if(isset($username)) {
        $bind = 0;
        foreach ($ldapconfig['usersdn'] as $value) {
            // Build user to bind
            $dn = "uid=" . $username . "," . $value . "," .$ldapconfig['basedn'];
            if (!$bind) {
                // Try to bind to LDAP server
                $bind = @ldap_bind($ds, $dn, $password);
            }
        }

        if (!$bind) {
            throw new HttpException(401, 'Invalid credentials');
        }

        try {
            // Search based on this attribute
            $filter = "(". config('services.ldap.filter_attribute') ."=$username)";

            // Get only these attributes
            $attr = explode(',', config('services.ldap.search_attributes'));

            // Perform the search and get results
            $sr = ldap_search($ds, $ldapconfig['basedn'], $filter, $attr);
            $info = ldap_get_entries($ds, $sr);

            // Get the results we need
            $cn = Str::title($info['0']['cn']['0']);
            $group = $info['0']['edupersonaffiliation']['0'];
            $is_author = false;
            if ($group === "staff") {
                $is_author = true;
            }
            $email = $info['0']['edupersonnickname']['0'];
        } catch (Exception $e) {
            return false;
        }
    }
```

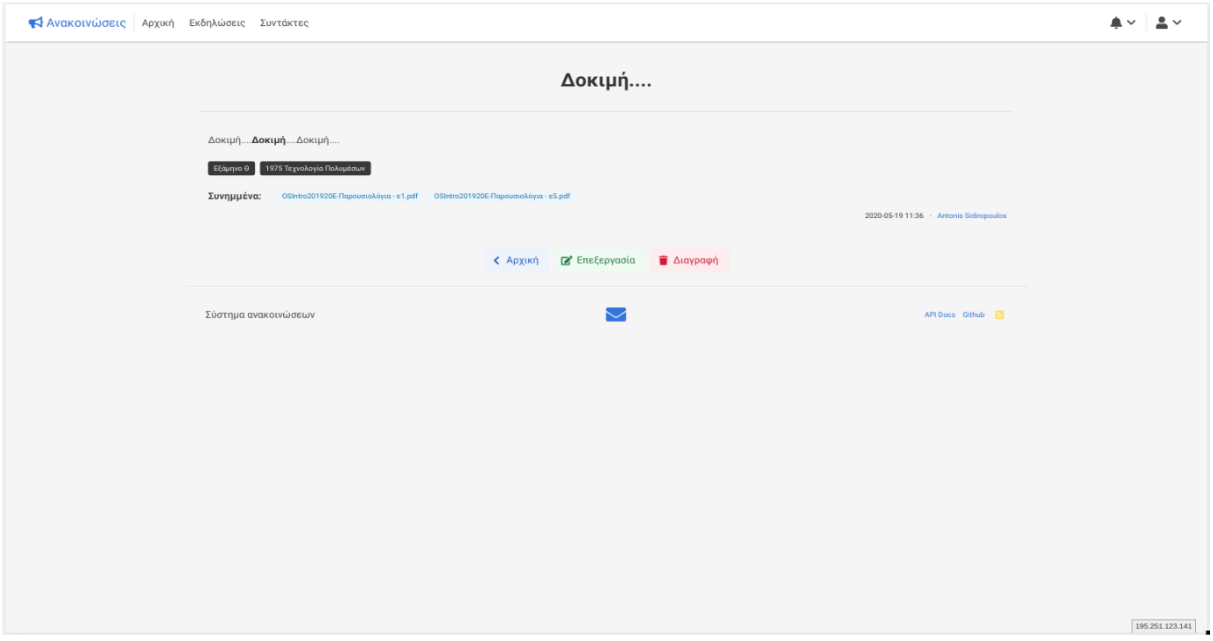
Εικόνα 26 Κώδικα για αυθεντικοποίηση χρήστη 1/2

```php
        // Create or update user based on our results
        $user = User::where('uid', $username)->first();
        if ($user === null) {
            $user = User::create(
                [
                    'name' => $cn,
                    'email' => $email,
                    'uid' => $username,
                    'is_author' => $is_author
                ]
            );
        } else {
            $user = User::where('uid', $username)->
            update([
                    'name' => $cn,
                    'email' => $email,
                    'uid' => $username,
                    'is_author' => $is_author
                ]
            );
        }
        $user = User::where('uid', $username)->first();

        // Set attributes for Laravel
        $attributes = [
            'id' => $user->id
        ];

        return new static ($attributes);
    }
```

Εικόνα 27 Κώδικα για αυθεντικοποίηση χρήστη 2/2

Προσπέλαση ανακοίνωσης

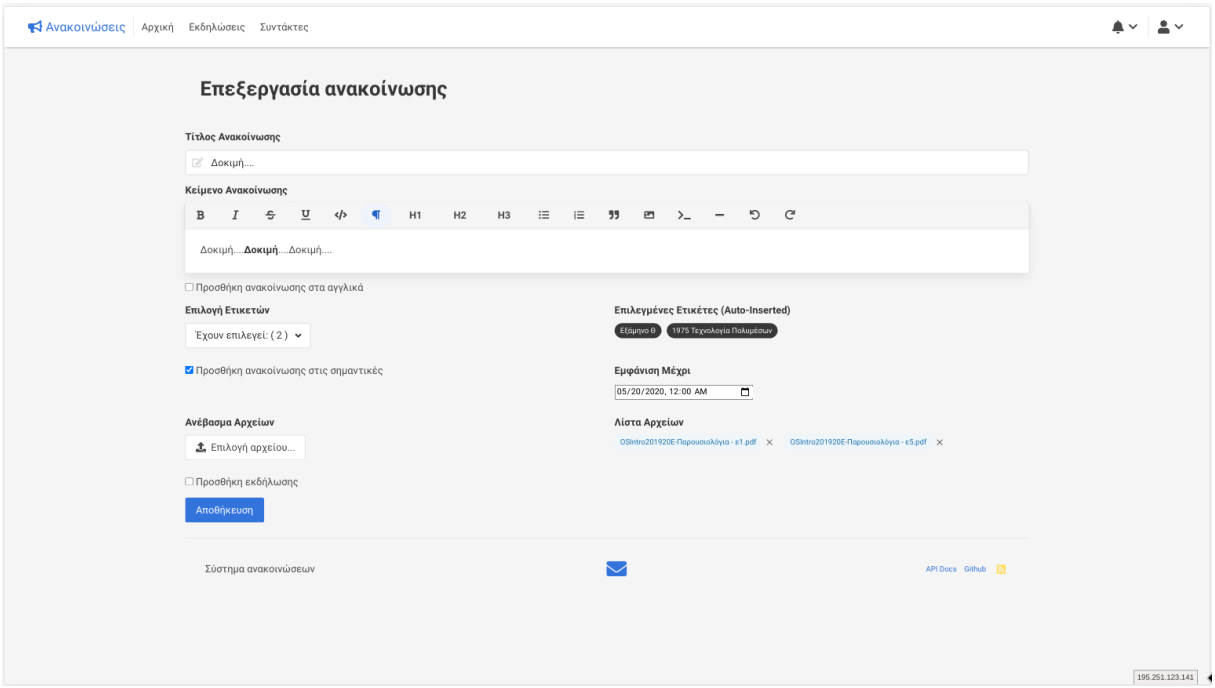
Παρακάτω, θα παρατεθεί μία εικόνα από το εσωτερικό μίας ανακοίνωσης:



Εικόνα 28 Προσπέλαση ανακοίνωσης

Τα κουμπιά «Επεξεργασία» και «Διαγραφή» εμφανίζονται, επειδή ο χρήστης αποτελεί διαχειριστή. Θα εμφανίζονται, επίσης, σε περίπτωση που ο χρήστης που βλέπει την ανακοίνωση είναι αυτός που την έχει προσθέσει. Τρίτοι χρήστες που αποτελούν συντάκτες δε θα έχουν δικαιώματα στην ανακοίνωση πέραν της ανάγνωσής της.

Επεξεργασία ανακοίνωσης

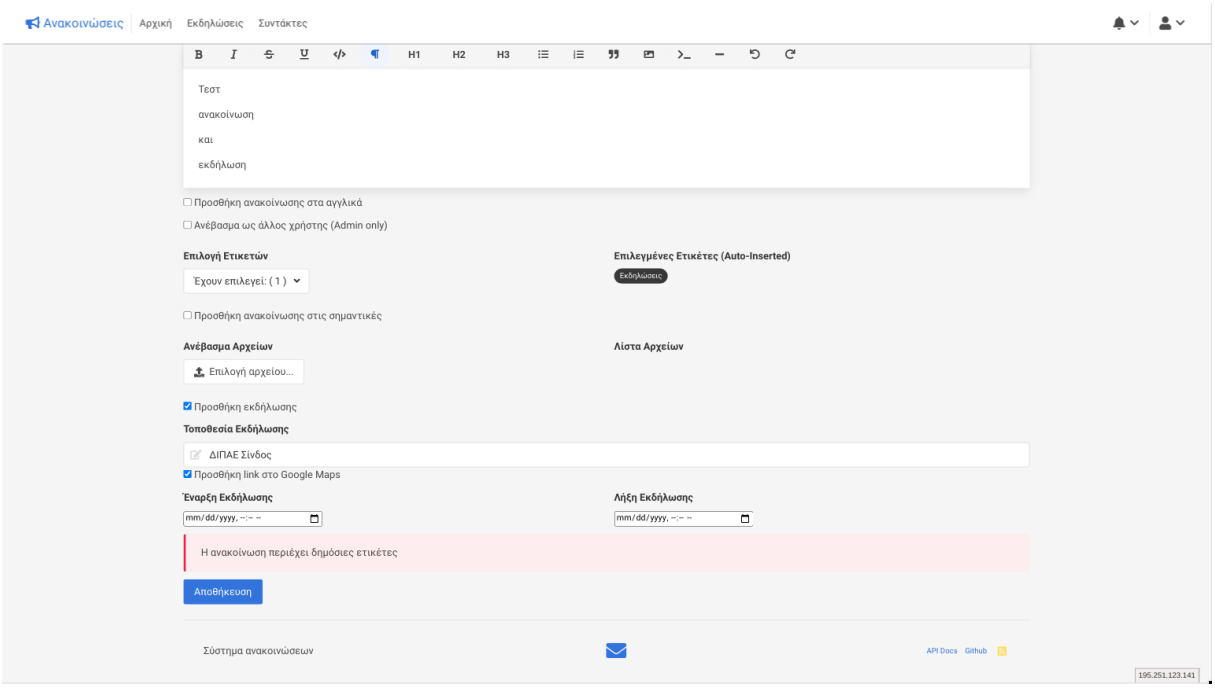


Εικόνα 29 Επεξεργασία ανακοίνωσης

Όπως αναφέρθηκε παραπάνω, καθώς χρησιμοποιείται η δυναμική δέσμευση εισαγωγής του Vue.js, θα γίνει λήψη της ανακοίνωσης και στη συνέχεια θα αντιστοιχηθούν όλα τα πεδία που επιστρέφουν από το API με τις αντίστοιχες μεθόδους εισαγωγής τους.

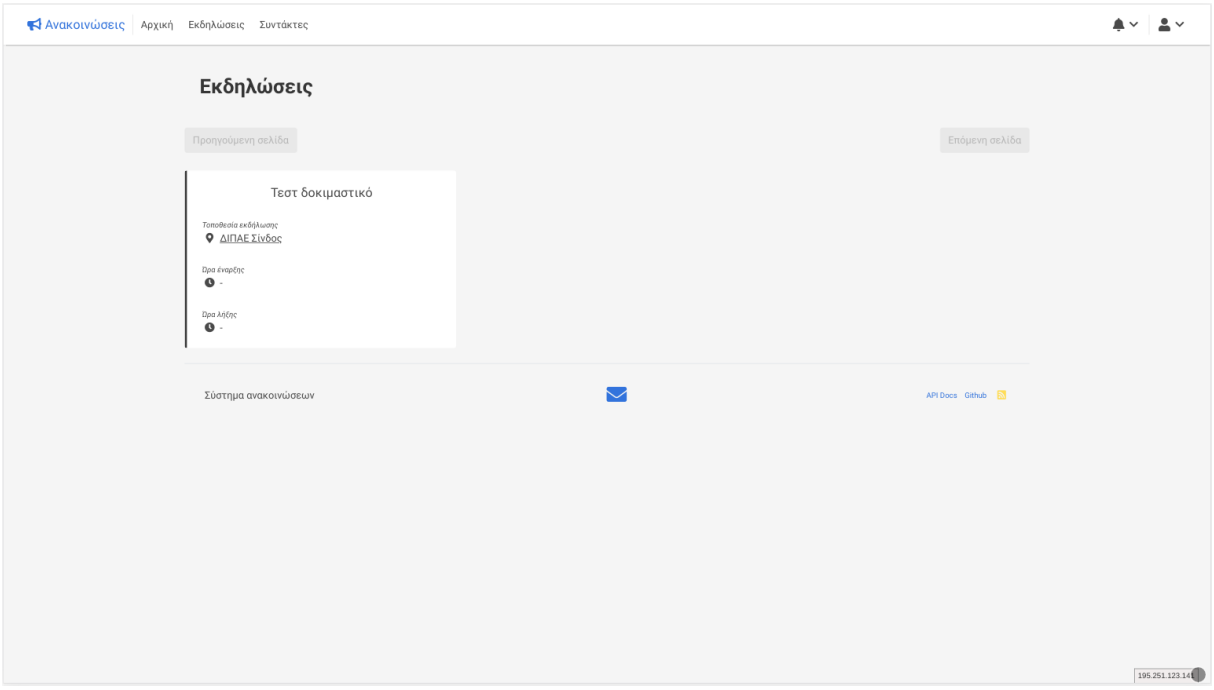
Προσθήκη ανακοίνωσης

Σε μία, όμοια με την παραπάνω, φόρμα γίνεται και η προσθήκη νέας ανακοίνωσης. Σε αυτήν την περίπτωση, όμως, καθώς δεν υπάρχει κάποιο αναγνωριστικό ανακοίνωσης που περνιέται στη σελίδα, όλα τα πεδία θα είναι άδεια. Ακόμη, εάν η ανακοίνωση, νέα ή παλιά που επεξεργάζεται, περιέχει μία δημόσια ετικέτα, εμφανίζεται ενημερωτικό μήνυμα στον χρήστη, όπως παρακάτω:



Εικόνα 30 Προσθήκη ανακοίνωσης

Εκδηλώσεις

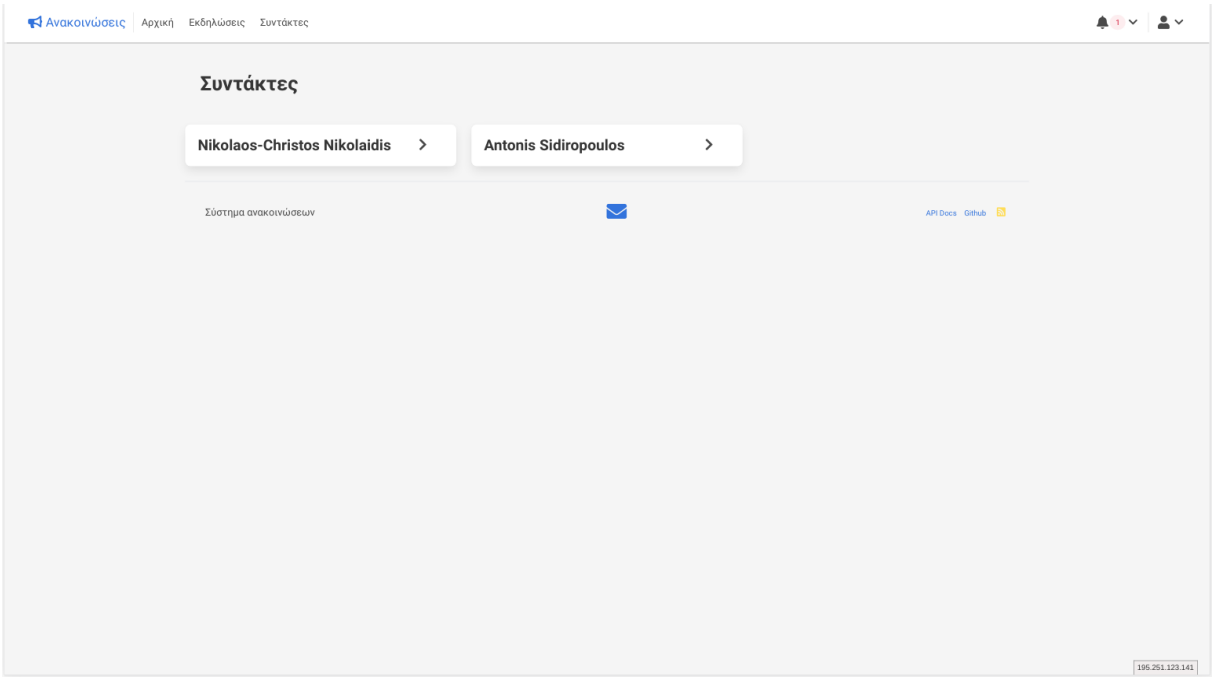


Εικόνα 31 Εκδηλώσεις

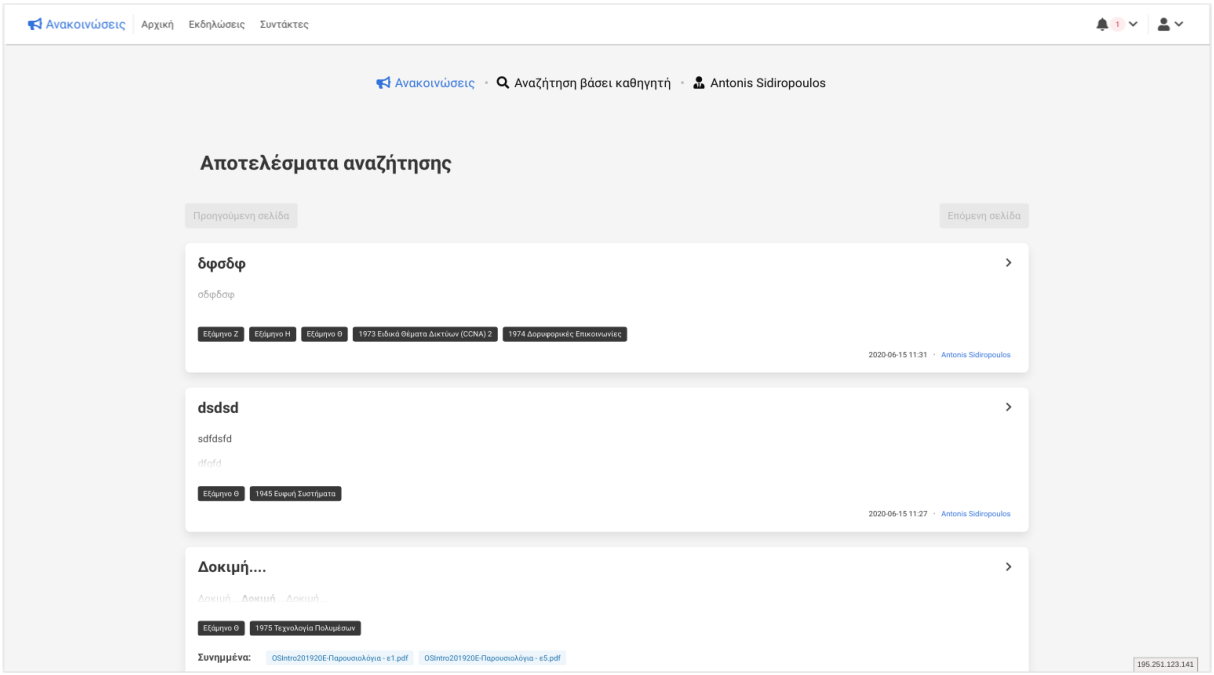
Οι εκδηλώσεις φαίνονται σε όλους τους χρήστες, συνδεδεμένους και μη. Για θέαση του περιεχομένου της ανακοίνωσης που αφορά την εκδήλωση ίσως απαιτείται σύνδεση αναλόγως με τον τύπο των ετικετών. Σημειώνεται πως μία ανακοίνωση για να οριστεί ως εκδήλωση δεν αρκεί να έχει την αντίστοιχη ετικέτα, αλλά πρέπει να έχει και την επιλογή «Προσθήκη ανακοίνωσης».

Συντάκτες

Για τη θέαση της σελίδας με τους συντάκτες απαιτείται είσοδος στην εφαρμογή. Εδώ εμφανίζονται όλοι οι συντάκτες της εφαρμογής και με κλικ του χρήστη πάνω σε κάποια καρτέλα εμφανίζονται όλες οι ανακοινώσεις του καθενός όπως φαίνεται στις παρακάτω εικόνες:



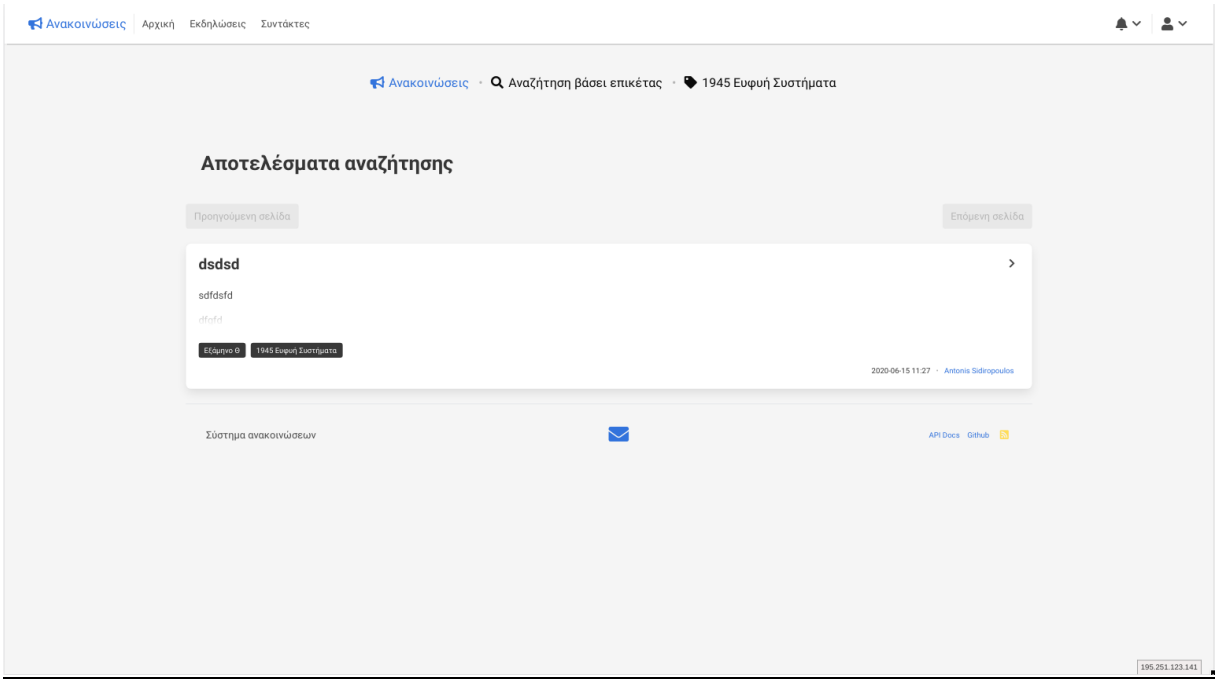
Εικόνα 32 Συντάκτες



Εικόνα 33 Αναζήτηση βάσης συντάκτη

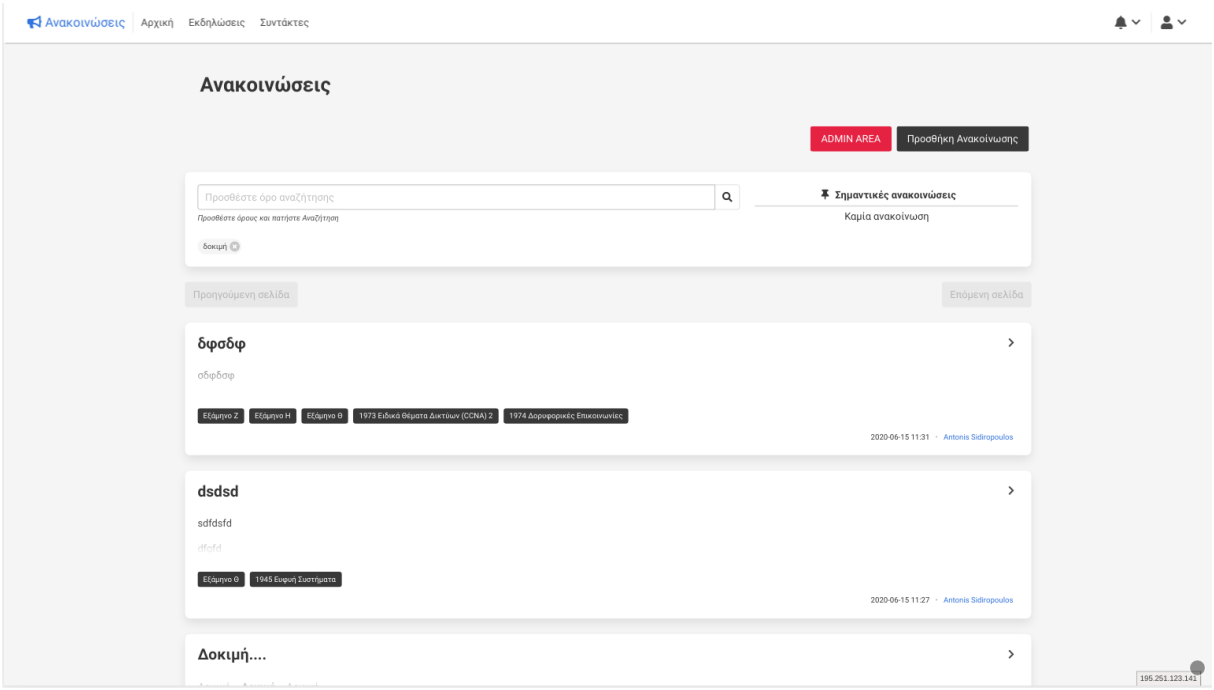
Αναζήτηση βάσει ετικέτας

Ο χρήστης κάνοντας κλικ σε οποιαδήποτε ετικέτα σε οποιοδήποτε σημείο της εφαρμογής μπορεί να κάνει αναζήτηση για όλες τις ανακοινώσεις που περιέχουν αυτήν την ετικέτα. Αν ο χρήστης είναι συνδεδεμένος, μπορεί να δει όλες τις ανακοινώσεις, ενώ εάν δεν είναι, μόνο αυτές που περιέχουν ετικέτα ως που έχει ορισθεί ως δημόσια.

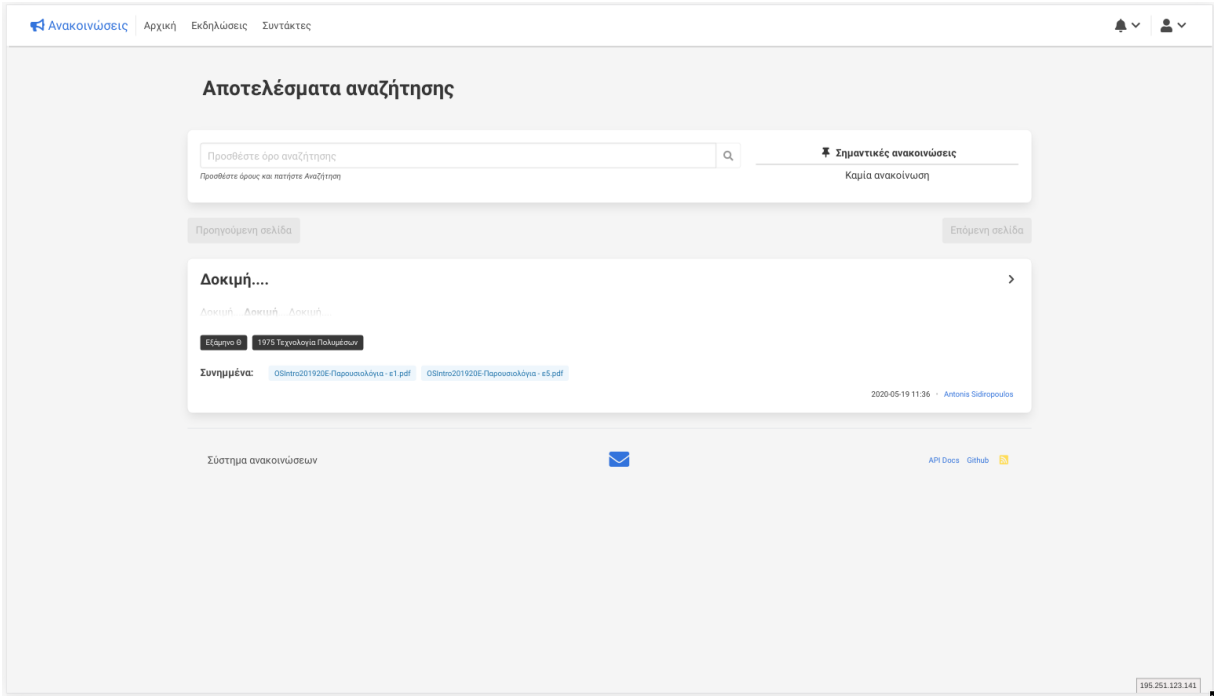


Εικόνα 34 Αναζήτηση βάσει ετικέτας

Αναζήτηση βάσει κειμένου



Εικόνα 35 Αναζήτηση βάσει κειμένου

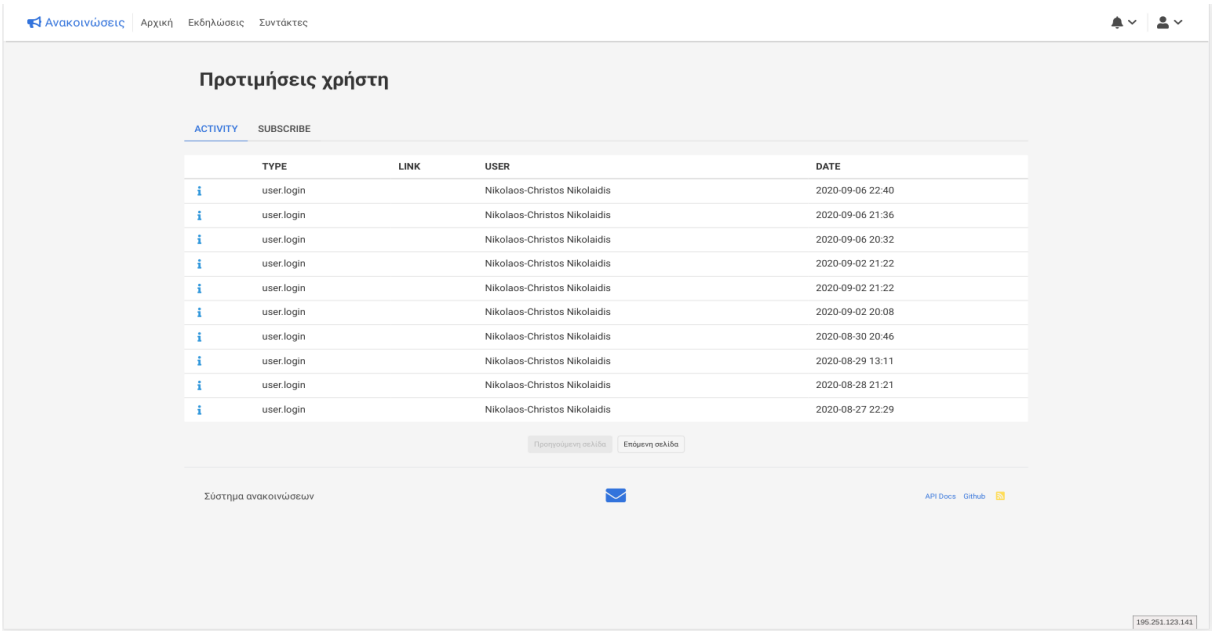


Εικόνα 36 Αποτελέσματα αναζήτησης βάσει κειμένου

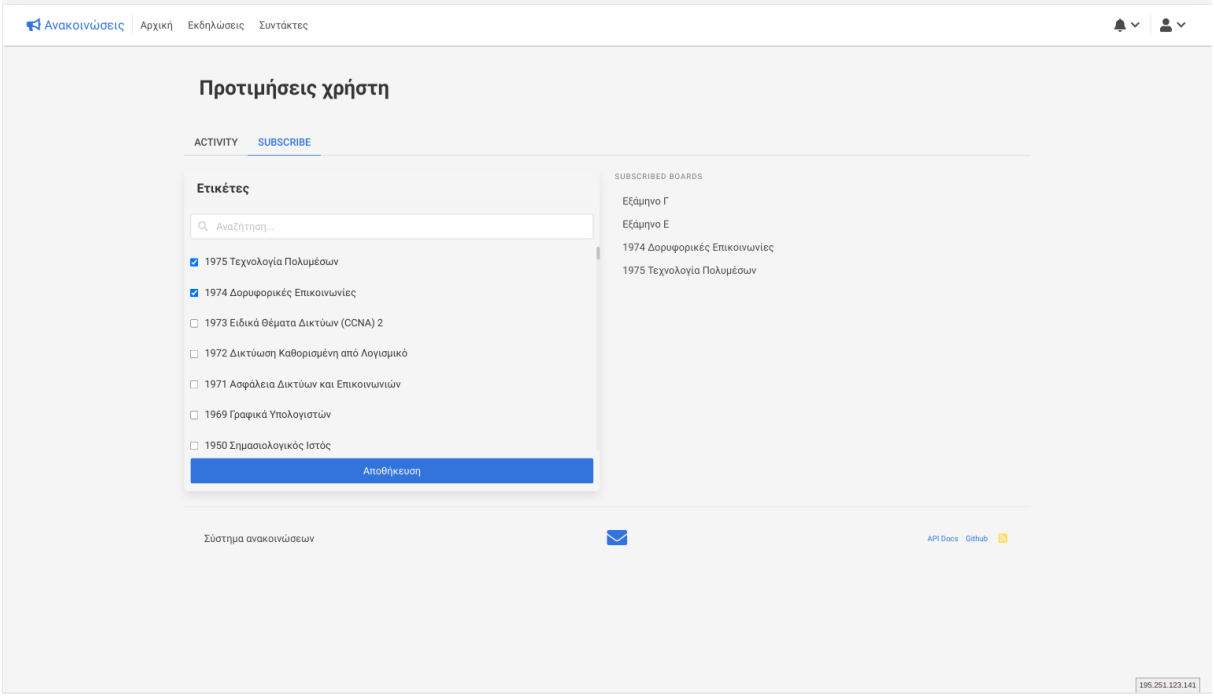
Στις παραπάνω εικόνες φαίνεται πώς πραγματοποιείται μία αναζήτηση στην εφαρμογή. Υπενθυμίζεται πως η αναζήτηση πραγματοποιείται μόνο στους τίτλους των ανακοινώσεων και όχι στο κείμενο.

Προτιμήσεις του χρήστη

Η σελίδα με τις προτιμήσεις του χρήστη χωρίζεται σε δύο ενότητες: η πρώτη αφορά αναλυτική παρουσίαση της δραστηριότητας του χρήστη και η δεύτερη τις ετικέτες στις οποίες έχει πραγματοποιήσει εγγραφή. Ακολουθούν παραδείγματα:



Εικόνα 37 Δραστηριότητα χρήστη

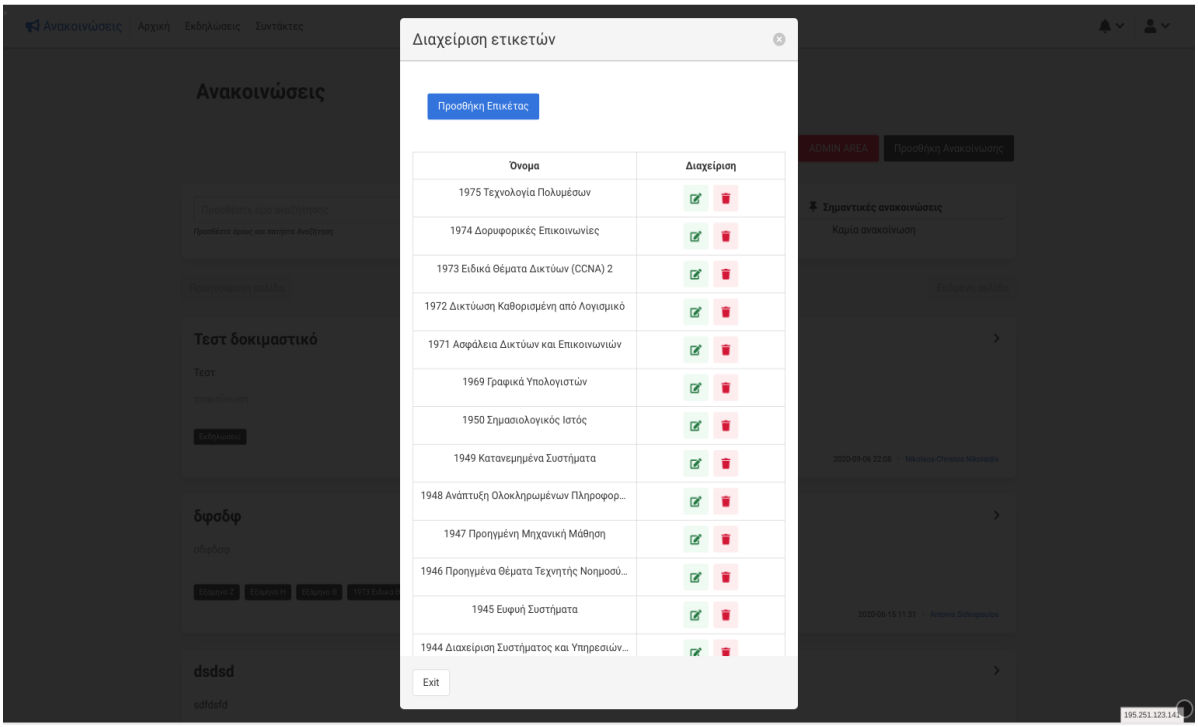


Εικόνα 38 Εγγραφή χρήστη σε ετικέτες

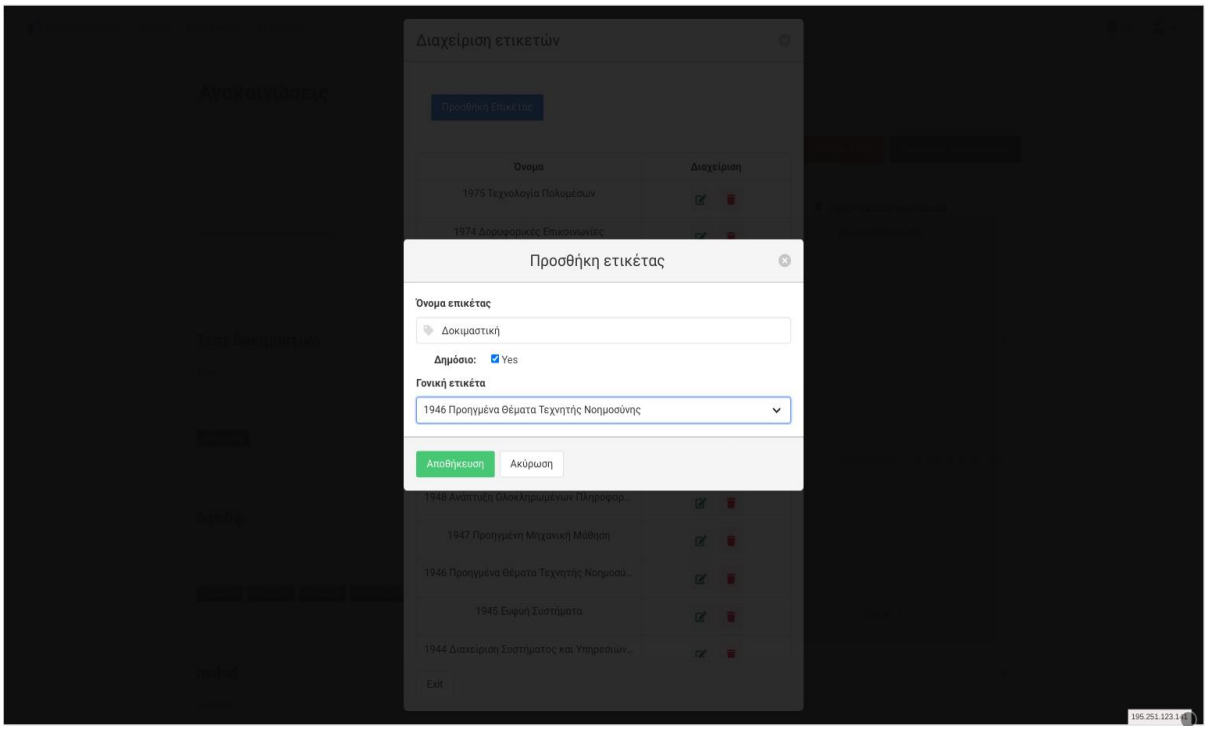
Με αλλαγή των επιλεγμένων στο αριστερό κομμάτι και κλικ στο κουμπί της αποθήκευσης ενημερώνονται οι εγγραφές του χρήστη στις αντίστοιχες ετικέτες. Κάθε χρήστης επιτρέπεται να έχει από καμία έως όλες τις εγγραφές στις ετικέτες. Ο κάθε χρήστης θα λαμβάνει, επίσης, ειδοποίηση κάθε φορά που προστίθεται μία ανακοίνωση που περιέχει κάποια ετικέτα στην οποία έχει εγγραφεί.

Διαχείριση ετικετών

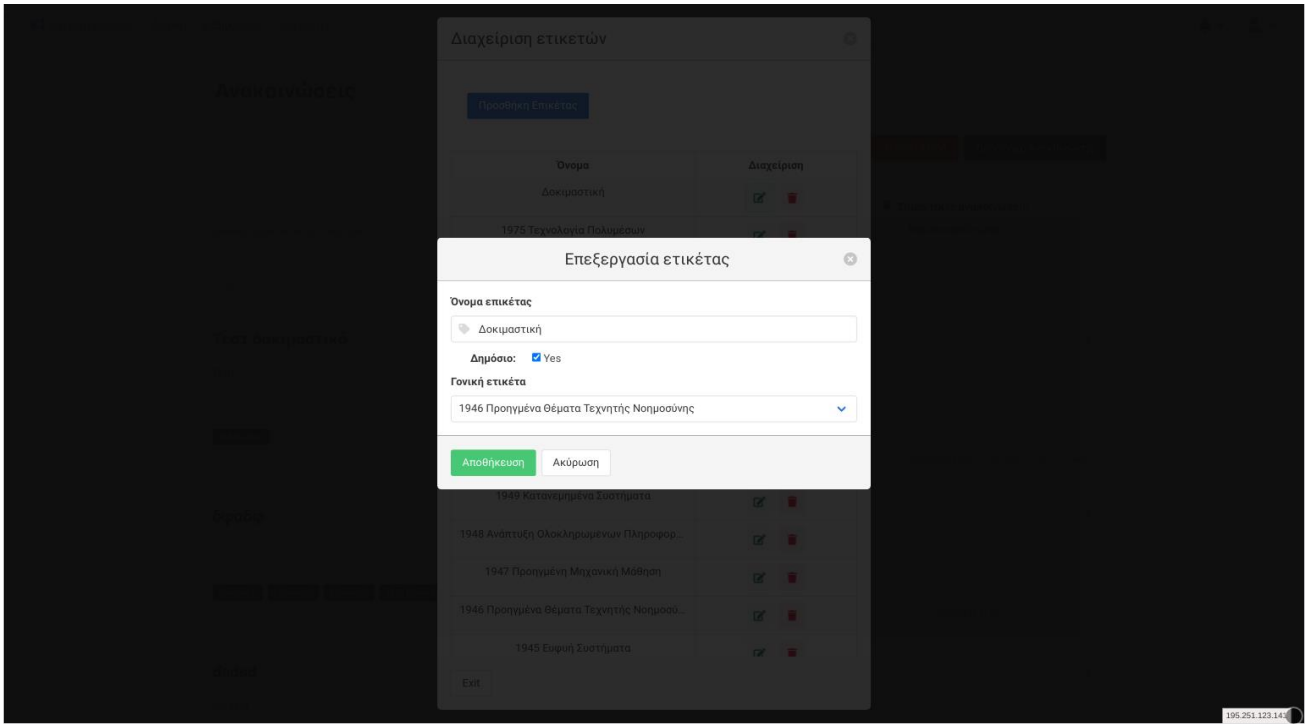
Η διαχείριση ετικετών είναι διαθέσιμη μόνο στους χρήστες διαχειριστές. Δεν υπάρχει τρόπος ελέγχου αναφορικά με το ποιος χρήστης θα ορισθεί ως διαχειριστής από το back-end. Μετά από προτροπή του επιτηρητή καθηγητή, αυτός ο διαχωρισμός θα γίνεται απευθείας στη βάση δεδομένων, ορίζοντας την τιμή του πεδίου is_admin στο κάθε χρήστη σε αληθή. Σε περίπτωση που υπάρξει κάποια συνθήκη αναφορικά με τον ορισμό ενός χρήστη ως διαχειριστή, μπορεί εύκολα να προστεθεί στη μέθοδο εισόδου που αναλύθηκε παραπάνω. Ακολουθεί παράδειγμα προσθήκης, επεξεργασίας και διαγραφής ετικέτας:



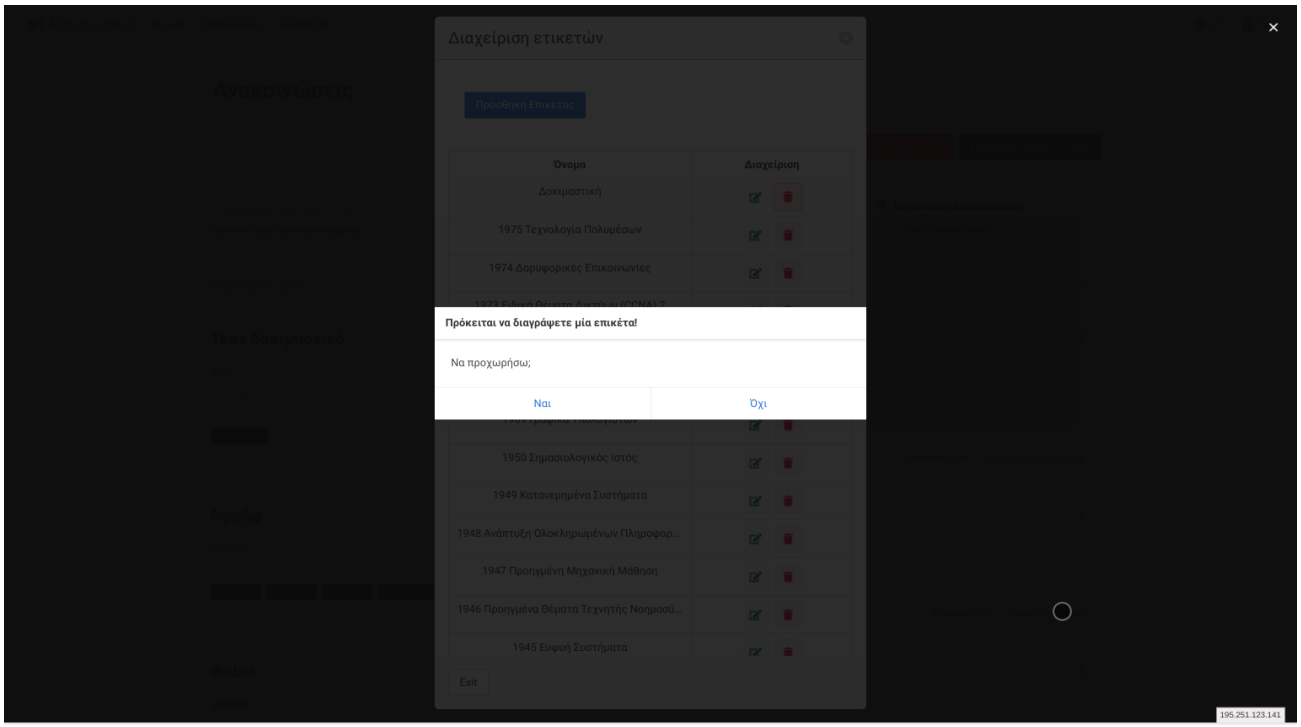
Εικόνα 39 Όλες οι ετικέτες



Εικόνα 40 Προσθήκη ετικέτας

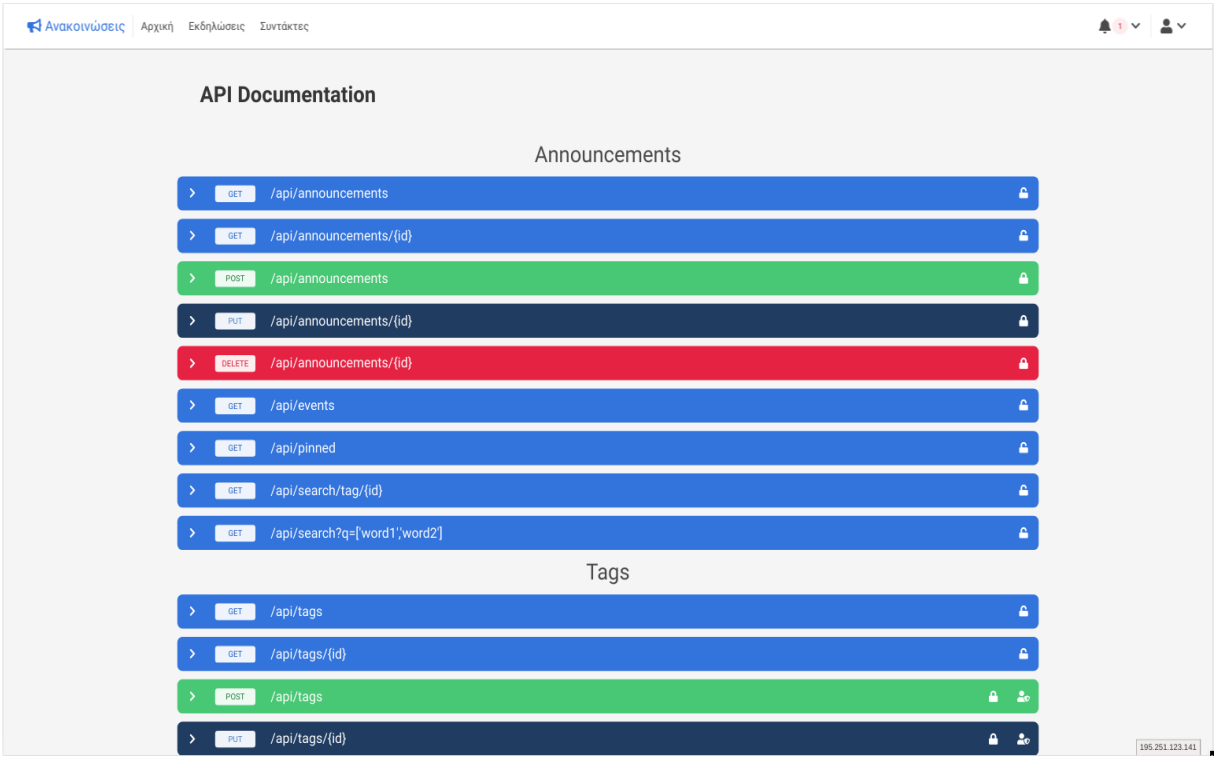


Εικόνα 41 Επεξεργασία ετικέτας



Εικόνα 42 Διαγραφή ετικέτας

Η σελίδα της τεκμηρίωσης



Εικόνα 43 Τεκμηρίωση API

Η σελίδα αυτή απαιτεί αυθεντικοποίηση. Εδώ υπάρχουν όλα τα σημεία του API με ενδεικτικές απαντήσεις που αυτά επιστρέφουν. Υπάρχουν, επίσης, τα μοντέλα που χρησιμοποιούνται με τα πεδία που αυτά έχουν στη βάση δεδομένων της εφαρμογής.

Επικοινωνία μεταξύ Vue components

Στην εφαρμογή πολλές φορές συναντάται η ανάγκη δύο ή περισσότερα components να επικοινωνούν μεταξύ τους. Για παράδειγμα, θα πρέπει το component που ελέγχει για την κατάσταση εισόδου του χρήστη να είναι σε θέση να ενημερώσει το component με τις ανακοινώσεις αναφορικά με το εάν ο χρήστης είναι συνδεδεμένος ή όχι. Ή το component του loader να γνωρίζει πότε έχουν ληφθεί όλες οι ανακοινώσεις ώστε να «κρυφτεί» έτσι ώστε να εμφανιστούν οι ανακοινώσεις. Γι' αυτόν τον λόγο, η εφαρμογή δημιουργεί ένα δεύτερο instance του Vue ώστε αυτό να χρησιμοποιηθεί για τη μεταφορά δεδομένων. Μία από τις διάφορες περιπτώσεις που χρησιμοποιείται στην παρούσα εφαρμογή είναι η παρακάτω που ελέγχει την κατάσταση σύνδεσης του χρήστη:

```
import { bus } from "../../app";

export default {
    data: function () {
        return {
            userAuthenticated: false,
        };
    },
    created: function () {
        this.checkAuth();
    },
    methods: {
        checkAuth: async function () {
            let vm = this;
            if (localStorage.getItem("token")) {
                const response = await axios.get("/api/auth/user");
                if (response.status == 200 && response.statusText == "OK") {
                    localStorage.setItem(
                        "user_info",
                        JSON.stringify(response.data.data)
                    );
                    vm.userAuthenticated = true;
                } else {
                    localStorage.removeItem("token");
                    localStorage.removeItem("user_info");
                    delete axios.defaults.headers.common["Authorization"];
                }
            }
            bus.$emit("authCheckFinished");
        },
    },
};
```

Εικόνα 44 Έλεγχος κατάστασης αυθεντικοποίησης χρήστη

Όπως παρατηρείται, υπάρχει ένα αντικείμενο με μία ιδιότητα με όνομα userAuthenticated που εκ ορισμού είναι ψευδές. Αφού ελεγχθεί ο τοπικός αποθηκευτικός χώρος του φυλλομετρητή για το αντικείμενο με όνομα token, πραγματοποιείται ένα αίτημα στο σημείο του API που επιστρέφει την κατάσταση σύνδεσης του χρήστη. Εάν η απάντηση έχει κωδικό 200 και κείμενο «OK», προχωρούμε σε ορισμό του αντικειμένου user_info, εισάγοντας τα δεδομένα που επιστρέφονται από το API και ορίζουμε την ιδιότητα userAuthenticated σε αληθή. Εάν, όμως, δεν ισχύουν οι δύο παραπάνω συνθήκες, αφαιρούμε από τον αποθηκευτικό χώρο τα αντικείμενα και μηδενίζουμε την ορισμένη ιδιότητα Authorization. Τέλος, επιστρέφουμε ένα event με όνομα authCheckFinished. Γι' αυτό το event περιμένουν άλλα components ώστε να πραγματοποιήσουν τις δραστηριότητες που πρέπει.

Κεφαλίδα Authorization

Όπως αναφέρθηκε στην αμέσως προηγούμενη παράγραφο, η ιδιότητα Authorization χρησιμοποιείται για τον ορισμό του τεκμηρίου που στέλνεται από το front-end στο back-end για τη λήψη των διαφόρων δεδομένων. Ο ορισμός του γίνεται στο αρχείο bootstrap.js, ενός από τα αρχεία js που χρησιμοποιούνται στην εφαρμογή. Το axios είναι το πακέτο εκείνο που επιτρέπει στο front-end της εφαρμογής να πραγματοποιεί αιτήματα. Αυτό παρέχει μία λειτουργία που ονομάζεται interceptor και η οποία φροντίζει ώστε κάθε αίτημα να περιλαμβάνει τις κεφαλίδες που έχουν οριστεί από τον προγραμματιστή. Υπάρχουν δύο ειδών interceptors: για τα αιτήματα και για τις απαντήσεις. Παρακάτω βλέπουμε τη χρήση τους:

```
axios.interceptors.request.use(function (config) {
    if (window.localStorage.getItem("token")) {
        token = window.localStorage.getItem("token");
        config.headers.Authorization = `Bearer ${token}`;
    } else {
        delete config.headers.Authorization;
    }
    window.axios.defaults.headers.common["Accept"] = "application/json";
    return config;
});

window.axios.interceptors.response.use(
    function (response) {
        return response;
    },
    function (error) {
        return error.response;
    }
);
```

Εικόνα 45 Κώδικας προσθήκης κεφαλίδας Authorization

Αναφορικά με τα αιτήματα, εφόσον, λοιπόν, έχει οριστεί στον τοπικό αποθηκευτικό χώρο το αντικείμενο token, αυτό χρησιμοποιείται σε κάθε αίτημα που στέλνετε στο API. Επίσης, ορίζεται πως ο τύπος απάντησης που περιμένει η εφαρμογή είναι JSON. Αναφορικά με τις απαντήσεις, ελέγχουμε την περίπτωση σφάλματος και επιστρέφουμε συγκεκριμένο κομμάτι από αυτήν, έτσι ώστε να εμφανιστούν τα μηνύματα λάθους στον χρήστη.

5.2 Επίλογος

Σε αυτό το κεφάλαιο έγινε μία πλήρης ανάλυση της εφαρμογής. Παρατέθηκαν εικόνες από όλες τις λειτουργίες που αυτή περιλαμβάνει, καθώς και εικόνες από κομμάτια κώδικα όπου κρίθηκε πως χρειάζεται. Η επιλογή του κώδικα που παρατέθηκε έγινε βάσει της δυσκολίας που ίσως συναντούσε κάποιος που θα προσπαθούσε να αναλάβει να συντηρήσει την εφαρμογή, γι’ αυτό και επιλέχθηκαν τα κομμάτια που παρεκκλίνουν από τον συνήθη τρόπο που χρησιμοποιεί το Laravel ή το Vue.js.

Κεφάλαιο 6ο: Μελλοντικές προτάσεις

6.1 Μελλοντικές προτάσεις

Όπως είναι προφανές, καθώς η παρούσα εφαρμογή αποτελεί μία πτυχιακή εργασία, δε θα μπορούσαν να λείπουν οι προτάσεις ώστε να γίνει πιο ποιοτική. Θα πρέπει να ληφθεί υπόψη πως οι απαιτήσεις ήταν συγκεκριμένες και καλά ορισμένες και, από τη στιγμή που υλοποιήθηκαν, η πτυχιακή θα πρέπει να θεωρείται ολοκληρωμένη. Στις επόμενες παραγράφους θα διατυπωθούν ορισμένες σκέψεις του γράφοντος αναφορικά με μελλοντικές επεκτάσεις και προτάσεις που θα μπορούσαν να κάνουν την εφαρμογή καλύτερη.

Εξωτερικός δρομολογητής κίνησης

Όπως έχει αναφερθεί σε ορισμένα σημεία της εργασίας, επί του παρόντος στην εφαρμογή γίνεται χρήση του εγγενούς δρομολογητή κίνησης του Laravel. Αυτό έχει ως αποτέλεσμα, παρότι τα αιτήματα προς το API να είναι stateless, η κατάσταση αυθεντικοποίησης να ορίζεται από την κατάσταση συνεδρίας μέσω του Laravel. Αυτό, με τη σειρά του, σημαίνει πως στην περίπτωση που, για παράδειγμα, έχει λήξει ένα τεκμήριο, ο χρήστης θα μπορεί να προσπελάσει μία διαδρομή επειδή δε θα έχει λήξει η συνεδρία στο Laravel. Βέβαια, δε θα έχει καμία πρόσβαση στα δεδομένα του API. Με τη χρήση ενός εξωτερικού δρομολογητή κίνησης, όπως για παράδειγμα του επίσημο πακέτο του Vue για αυτήν τη λειτουργία ονόματι vue-router, αυτή η συμπεριφορά θα μπορούσε να αποφευχθεί. Στα θετικά αυτού του τρόπου υλοποίησης είναι πως ο προγραμματιστής δεν ασχολείται με παραμετροποίηση των ρυθμίσεων για την κοινή χρήση πόρων από διαφορετικές προελεύσεις (Cross-Origin Resource Sharing, CORS).

Κεντρικός αποθηκευτικός χώρος

Όπως αναφέρθηκε και παραπάνω, η επικοινωνία μεταξύ των διαφόρων components του Vue γίνεται με τη χρήση ενός δεύτερο Vue instance (γνωστό και ως message bus). Αυτή η υλοποίηση, παρότι δουλεύει, δημιουργεί προβλήματα στη συντήρηση του προγράμματος και στην προσθήκη καινούργιων λειτουργιών. Για τη λύση αυτού του προβλήματος, τα περισσότερα JavaScript frameworks χρησιμοποιούν κεντρικό αποθηκευτικό χώρο για τα δεδομένα γνωστό και ως store. Αυτό κάνει τη διαχείρισή τους πιο εύκολη και την εφαρμογή πιο εύκολα συντηρήσιμη.

Διαχωρισμός back-end και front-end

Όπως αναφέρθηκε και παραπάνω, η κατάσταση εισόδου του χρήστη είναι συνδεδεμένη με την κατάσταση συνεδρίας στο Laravel. Μία ακόμη ενδιαφέρουσα πρόταση είναι ο πλήρης διαχωρισμός του back-end από το front-end ακόμη και σε επίπεδο domain names. Το API, για παράδειγμα, να βρίσκεται στο όνομα api.aboard.iee.ihu.gr και το front-end στο aboard.iee.ihu.gr. Σε αυτήν την περίπτωση, βέβαια, θα έπρεπε να γίνει ρύθμιση για CORS. Το ενδιαφέρον με αυτήν την προσέγγιση είναι πως θα μπορούσε πιο εύκολα να επεκταθεί η εφαρμογή υποστηρίζοντας διαφορετικούς τύπους εφαρμογών (π.χ. Android, iOS) που θα χρειάζονταν διαφορετικούς τύπους αυθεντικοποίησης.

Χρήση τεκμηρίου ανανέωσης (refresh token)

Στην παρούσα υλοποίηση, δε γίνεται χρήση τεκμηρίων ανανέωσης, συνεπώς, όταν λήξει ένα τεκμήριο πρόσβασης, ο χρήστης γίνεται αυτόματα αποσυνδέεται από την εφαρμογή με την λήξη του χρονικού

ορίου του τεκμηρίου. Για να αποφευχθεί αυτή η συμπεριφορά, θα μπορούσε να γίνεται χρήση τεκμηρίων ανανέωσης.

Γλωσσική τοπικοποίηση (Localization)

Αυτήν τη στιγμή, στην εφαρμογή όλα τα μηνύματα επιτυχίας ή σφάλματος υπάρχουν μέσα στον κώδικα των Vue components γραμμένα μόνο στα ελληνικά. Σε μία ορθότερη υλοποίηση, το API θα έπρεπε να παρέχει ένα σημείο (endpoint) στο οποίο θα γίνεται αντιπαραβολή των αντιστοίχων κωδικών με μηνύματα σε διάφορες γλώσσες (π.χ. ελληνικά και αγγλικά) και να εμφανίζονται αντιστοίχως στο front-end. Το ίδιο θα έπρεπε να συμβαίνει και στα διάφορα στοιχεία HTML (π.χ. τίτλους σελίδων, κείμενα στα κουμπιά κ.ο.κ.). Αυτό θα έδινε τη δυνατότητα στον χρήστη να επιλέγει τη γλώσσα που αυτός επιθυμεί και η εφαρμογή θα μπορούσε να χρησιμοποιηθεί επιτυχώς από περισσότερους χρήστες. Σημειώνεται πως η δυνατότητα του localization παρέχεται εγγενώς και αυτή από το Laravel.

Περισσότερες μέθοδοι ταυτοποίησης

Ορισμένη από τις απαιτήσεις, η ταυτοποίηση ενός χρήστη γίνεται μέσω του LDAP server του τμήματος. Μία προσθήκη θα μπορούσε να είναι η επιλεκτική αλλαγή της σε κάποια άλλη μέθοδο (π.χ. εγγενής βάση) και η αντίστοιχη αυτόματη αλλαγή στις μεθόδους και τα στοιχεία που χρησιμοποιούνται.

Καλύτερο admin page

Η διαχείριση των ετικετών γίνεται μέσω ενός αναδυόμενου παραθύρου (modal) στην αρχική οθόνη. Αυτή η συμπεριφορά θα μπορούσε να αλλάξει και να γίνει ένα πλήρες διαχειριστικό μενού για τις ετικέτες και για άλλες λειτουργίες που πιθανώς να κριθούν χρήσιμες από τους χρήστες.

Περισσότερες επιλογές για ειδοποιήσεις του χρήστη

Τώρα οι χρήστες ενημερώνονται για τις ανακοινώσεις που περιέχουν ετικέτες στις οποίες έχουν κάνει εγγραφή μόνο μέσω του μενού στην αρχική οθόνη. Αυτό θα μπορούσε να αλλάξει και να υπάρχουν και κάποιες άλλες επιλογές αναφορικά με τις ενημερώσεις (π.χ. e-mail, push notifications κ.λπ). Άλλωστε, το σύστημα ειδοποιήσεων του Laravel υποστηρίζει εγγενώς διάφορα κανάλια ειδοποιήσεων και μπορεί, ακόμη, να επεκταθεί ώστε να υποστηρίξει ό,τι άλλο μπορεί να σκεφτεί κανείς. Ακόμη, η ενεργοποίηση ή όχι συγκεκριμένου καναλιού μπορεί να γίνει από τον χρήστη ώστε να είναι ακόμη πιο παραμετροποιήσιμο το σύστημα.

Αυτόματο API Testing

Τα τελευταία χρόνια υπάρχει μία τάση στο προγραμματισμό για συστήματα back-end που ονομάζεται Ανάπτυξη οδηγούμενη από τον έλεγχο (Test Driven Developemnt - TDD). Σε αυτήν γράφονται tests για το API που ελέγχουν πως το API είναι up (δηλαδή ενεργό) και πως επιστρέφει τα αναμενόμενα αποτελέσματα. Αυτή είναι μία πρακτική που έχει αρχίσει να συνηθίζεται, καθώς βοηθάει τους διαχειριστές συστημάτων και τους προγραμματιστές στην επίλυση προβλημάτων όταν αυτά εμφανιστούν. Αυτή η διαδικασία περιλαμβάνει ελέγχους των αποτελεσμάτων που επιστρέφει το API σε συγκεκριμένα και καλά ορισμένα tests. Το Laravel υποστηρίζει API και για λόγους καλών πρακτικών (best practises) σε κάθε back-end σύστημα καλό θα ήταν να υπάρχει αυτόματο API testing.

6.2 Επίλογος

Στο τελευταίο αυτό κεφάλαιο έγινε προσπάθεια να συγκεντρωθούν όλα εκείνα τα στοιχεία που θα έκαναν την εφαρμογή πιο ποιοτική και σωστά γραμμένη. Όλες οι προτάσεις αφορούν πράγματα που ο γράφον δε γνώριζε όταν ξεκίνησε τη συγγραφή της εφαρμογής, έλαβε γνώση ύπαρξής τους στην πορεία της υλοποίησης, αλλά, για διάφορους λόγους δεν κατέστη δυνατό να υλοποιηθούν.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] «Apache Vs NGINX – Which Is The Best Web Server for You?», [Ηλεκτρονικό] Available: https://serverguy.com/comparison/apache-vs-nginx

[2] «Basic LDAP Concepts», [Ηλεκτρονικό] Available: https://ldap.com/basic-ldap-concepts

[3] «CSS: Cascading Style Sheets», [Ηλεκτρονικό] Available: https://developer.mozilla.org/en-US/docs/Web/CSS

[4] «How Laravel implements MVC and how to use it effectively», [Ηλεκτρονικό] Available: https://blog.pusher.com/laravel-mvc-use

[5] «HTML: Hypertext Markup Language», [Ηλεκτρονικό] Available: https://developer.mozilla.org/en-US/docs/Web/HTML

[6] «Composer Documentation», [Ηλεκτρονικό] Available: https://getcomposer.org/doc/00-intro.md

[7] «JavaScript», [Ηλεκτρονικό] Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[8] «JSON API Latest Specification», [Ηλεκτρονικό] Available: https://jsonapi.org/format

[9] «LDAP Overview», [Ηλεκτρονικό] Available: https://ldapwiki.com/wiki/LDAP%20Overview

[10] «NGINX Vs Apache – Which Is The Best Web Server In 2020?», [Ηλεκτρονικό] Available: https://www.plesk.com/blog/various/nginx-vs-apache-which-is-the-best-web-server

[11] «OAuth 2.0», [Ηλεκτρονικό] Available: https://oauth.net/2

[12] «Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content», [Ηλεκτρονικό] Available: https://tools.ietf.org/html/rfc7231

[13] «PATCH Method for HTTP», [Ηλεκτρονικό] Available: https://tools.ietf.org/html/rfc5789

[14] «SOAP Vs. REST: Difference between Web API Services», [Ηλεκτρονικό] Available: https://www.guru99.com/comparison-between-web-services.html

[15] «OAuth 2.0 Server, Terminology», [Ηλεκτρονικό] Available: https://oauth2.thephpleague.com/terminology/

[16] «Types of APIs», [Ηλεκτρονικό] Available: https://rapidapi.com/blog/types-of-apis

[17] «VueJS - Overview», [Ηλεκτρονικό] Available: https://www.tutorialspoint.com/vuejs/vuejs_overview.htm

[18] «SOAP vs REST (differences)», [Ηλεκτρονικό] Available: https://stackoverflow.com/questions/19884295/soap-vs-rest-differences

[19] «What is JSON and why would I use it?», [Ηλεκτρονικό] Available: https://stackoverflow.com/questions/383692/what-is-json-and-why-would-i-use-it

[20] «What is REST», [Ηλεκτρονικό] Available: https://restfulapi.net

[21] «What Is RSS? RSS Explained», [Ηλεκτρονικό] Available: http://www.whatisrss.com

[22] «Laravel Documentation», [Ηλεκτρονικό] Available: https://laravel.com/docs/6.x

[23] «Vue.js Documentation», [Ηλεκτρονικό] Available: https://vuejs.org/v2/guide

[24] Doumas O. (2016). «Σχεδίαση μιας Εξειδικευμένης Πλατφόρμας Συστήματος διαδικτυακών υπηρεσιών REST και αυτόματη παραγωγή εκτελέσιμου κώδικα» [Ηλεκτρονικό] Available: http://ikee.lib.auth.gr/record/292040/files/doumas_odysseas_mde.pdf

[25] Liapis C. (2018). «Πλατφόρμα διαχείρισης προσωπικού σε ASP.NET με αρχιτεκτονική MVC» [Ηλεκτρονικό] Available: https://apothesis.lib.teicrete.gr/bitstream/handle/11713/9059/LiapisChristos2018.pdf?sequence=1&isAllowed=y

[26] Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, 3rd Edition. McGraw-Hill, 2003.

[27] Thomson Laura and Welling Luke, PHP and MySQL Web Development, 5th Edition. Ergodebooks, 2013.