Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Τμήμα Μηχανικών Πληροφορικής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη διαδικτυακής εφαρμογής και επέκτασης περιηγητή, για τη συλλογή και οργάνωση διαδικτυακών ψηφιακών πόρων σε προσωπικούς και κοινόχρηστους χώρους

Γαργανουράκης Μιχαήλ
Αρ. Μητρώου: 3575

Επιβλέπων Καθηγητής: **Παπαδάκης Νικόλαος**

Ηράκλειο Κρήτης
2019

Ευχαριστίες

Ολοκληρώνοντας την πτυχιακή μου εργασία, θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Παπαδάκη Νικόλαο τόσο για την υποστήριξη και καθοδήγηση του σε όλα τα βήματα της διπλωματικής μου εργασίας, αλλά και για την εμπιστοσύνη που μου έδειξε με την επιλογή μου να ακολουθήσω ένα τόσο ιδιαίτερο θέμα γεμάτο προκλήσεις. Ένα μεγάλο ευχαριστώ οφείλω επίσης, στην οικογένεια μου για την στήριξη τους σε κάθε επίπεδο τα χρόνια αυτά, δίχως της οποίας δεν θα ήταν δυνατό το ταξίδι αυτό.

Abstract

The aim of this thesis is to develop and build a web application along with a browser extension that is able to aggregate, manage and organize web resources collected around the web by the user, using the given extension. The extension itself, will also provide the ability to bind those resources with tags meaningful to the collector, but also suggest any tags already made in each webpage, crawling through its the content or even by searching on the application's database for possible resource matches. In that way, users will be able to group their interests with words relative with that they want to correlate it and make collections either personal or even shared.

Σύνοψη

Η εν λόγω πτυχιακή εργασία έχει ως στόχο τη δημιουργία διαδικτυακής εφαρμογής/πλατφόρμας καθώς και την ανάπτυξη επέκτασης περιηγητή (browser extension) που θα καθιστούν δυνατή τη συλλογή και οργάνωση διαδικτυακού υλικού όπως και τη δημιουργία ψηφιακών συλλογών.

Ειδικότερα, η επέκταση περιηγητή θα δίνει στο χρήστη τη δυνατότητα να συλλέγει από το διαδίκτυο ψηφιακούς πόρους (ιστοσελίδες, έγγραφα, κλπ.) και να τους κατατάσσει σε προσωπικό ή κοινόχρηστο χώρο για μελλοντική χρήση. Ταυτόχρονα, μέσω μηχανισμού, θα ελέγχει τόσο τη βάση δεδομένων όσο και την σελίδα για προτεινόμενες ετικέτες/μεταδεδομένα βάσει του περιεχομένου της. Ο χρήστης θα έχει τη δυνατότητα κατά την αποθήκευση να προσθέσει δικές του λέξεις κλειδιά, καθώς και να επιλέξει από τις προτεινόμενες ετικέτες για τη καλύτερη οργάνωση του περιεχομένου που συλλέγει.

Τέλος, η διαδικτυακή εφαρμογή θα επιτρέπει την τροποποίηση και οργάνωση του υλικού που έχει συλλεχθεί, το σχολιασμό του, καθώς και τον διαμοιρασμό του σε τρίτους. Επίσης, θα δίνεται η δυνατότητα αναζήτησης σε όλο το αποθηκευμένο υλικό.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Λίστα Πινάκων

1 Εισαγωγή

Η διπλωματική εργασία —ένα μικρό χρονικά κομμάτι των προπτυχιακών σπουδών ενός φοιτητή— χρήζει σπουδαίας σημαντικότητας, καθώς προσφέρει στον φοιτητή μία πτυχή του εργασιακού τομέα που έχει επιλέξει να ακολουθήσει, εντάσσοντας τον σε μία πρώτη γενική κατεύθυνση αυτού. Παράλληλα, δίνει την ευκαιρία στον σπουδαστή να πειραματιστεί με τεχνολογίες και πρακτικές, που ακολουθούνται στον κλάδο του, παρέχοντας ταυτόχρονα την ευελιξία της επιλογής και του συνδυασμού, ώστε να προσκομίσει όσο το δυνατόν μία πιο σφαιρική εικόνα.

1.1 Περίληψη

Ο σκοπός της εν λόγω διπλωματικής εργασίας είναι ο σχεδιασμός και η ανάπτυξη πλατφόρμας συλλογής και οργάνωσης διαδικτυακών πόρων σε προσωπικές και διαμοιραζόμενες συλλογές. Η ολοκληρωμένη πλατφόρμα σε συνδυασμό με την επέκταση περιηγητή, θα προσφέρει στον χρήστη μία συνολική λύση για την οργάνωση διαδικτυακού υλικού, σε συλλογές ιδιωτικές αλλά και με δυνατότητα να διαμοιραστεί με τρίτους χρήστες της ίδιας εφαρμογής.

Η επέκταση περιηγητή θα επιτρέπει την συγκέντρωση υλικού κάθε τύπου (π.χ. βίντεο, εικόνων, άρθρων, ιστοσελίδων εγγράφων κλπ.), επιτρέποντας την χρήση ετικετών (tags) που θα «δένουν» με τον συγκεκριμένο πόρο, κατηγοριοποιώντας τον μετέπειτα στην διαδικτυακή εφαρμογή ανάλογα. Ταυτόχρονα, θα σαρώνει/ελέγχει την σελίδα για λέξεις-κλειδιά, προτείνοντας στο χρήστη τη χρήση τους για καλύτερη ταξινόμηση του επιλεγμένου περιεχομένου. Στο παρασκήνιο, θα γίνεται σχετικός έλεγχος στην βάση δεδομένων της εφαρμογής για τυχόν συλλογή του ίδιου πόρου από διαφορετικούς χρήστες/συνεργάτες, ειδοποιώντας τον συλλέκτη ότι ο συγκεκριμένος πόρος έχει συλλεχθεί στο παρελθόν και έχει συνδυαστεί με συγκεκριμένες λέξεις-κλειδιά.

Η συγκεκριμένη σουίτα αναπτύχθηκε, με γνώμονα τόσο την ατομική οργάνωση περιεχομένου και υλικού, όσο και την συνεργασία μεταξύ ατόμων μίας ομάδας και την ανάγκη για διαμοιρασμό κατηγοριοποιημένου περιεχομένου, για την σωστότερη και ομαλότερη διεξαγωγή εργασιών. Για τις ανάγκες αυτές, καθώς και άλλα ζητούμενα που θα αναπτυχθούν παρακάτω, με την απαραίτητη μελέτη αναπτύχθηκε ένας κεντρικός server με τον οποίο επικοινωνούν και οι δύο αυτές μεγάλες οντότητες του συγκεκριμένου συστήματος (διαδικτυακή εφαρμογή & επέκταση περιηγητή).

Πιο συγκεκριμένα με την χρήση της πλατφόρμας Node.js, η οποία βασίζεται στο V8 engine[1] του περιηγητή Chrome και έχει αναπτυχθεί σε περιβάλλον JavaScript, δημιουργήθηκε ο κεντρικός server που είναι υπεύθυνος τόσο για την ένωση της διαδικτυακής εφαρμογής με την επέκταση περιηγητή, όσο και για την επικοινωνία κάθε οντότητας με την PostgreSQL βάση δεδομένων του συγκεκριμένου συστήματος. Για την ανάπτυξη του συγκεκριμένου σχήματος, προέκυψε επίσης η ανάγκη δημιουργίας ενός API που έχει τον ρόλο ενός διαύλου επικοινωνίας, ενώ ταυτόχρονα κρίθηκε αναγκαία η χρήση των τεχνολογιών JavaScript, HTML, JSON, jQuery, EJS και της βιβλιοθήκης γραφικών Materialize —που είναι βασισμένη στην αρχιτεκτονική της Google γνωστή και ως Material Design.

1.2 **Κίνητρο για την Διεξαγωγή της Εργασίας – Στόχοι**

Όπως είναι γνωστό, ανέκαθεν η επικοινωνία και η οργάνωση μεταξύ μελών μιας ομάδας ήταν μια διαδικασία χρονοβόρα και δύσκολη, ειδικότερα όταν τα άτομα αυτά για πρώτη φορά θα συνυπήρχαν μέλη της ίδιας ομάδας. Επίσης, το γεγονός ότι το κάθε μέλος θα μπορεί να αφιερώσει χρόνο ανεξάρτητα και δρώντας μεμονωμένα στη συλλογή δεδομένων, σε μεγάλους όγκους πληροφοριών προκαλεί σύγχυση, καθώς ίδια δεδομένα συλλέγονται πολλαπλές φορές και άσκοπα ξοδεύεται χρόνος από διαφορετικά μέλη για την ανάλυση δεδομένων που έχουν ήδη αναλυθεί.

Ακόμα, η προσωπική οργάνωση και κατηγοριοποίηση πόρων πολλές φορές οδηγεί σε ανάλογη δυσκολία, καθώς έχουμε τόσο την ανάγκη να ενισχύσουμε το περιεχόμενο του ψηφιακού πόρου που μόλις συλλέξαμε για προσωπικό λόγο, όσο και την πολλαπλή κατηγοριοποίηση του για μετέπειτα χρήση σε διαφορετικές εφαρμογές.

Τα προβλήματα αυτά έχει ως επίκεντρο επίλυσης η σουίτα που μου δίνεται η ευκαιρία να αναπτύξω μέσω της διπλωματικής μου εργασίας και στοχεύει στη διευκόλυνση τόσο των ομάδων καθιστώντας ευκολότερη την ομαδική συλλογή και διαμοιρασμό υλικού, αλλά και την απλοποίηση και ενίσχυση της εμπειρίας στη συλλογή προσωπικών ενδιαφερόντων.

1.3 **Δομή Εργασίας**

Η εν λόγω εργασία είναι δομημένη σε πέντε (5) κεφάλαια τα οποία έχουν ως στόχο τη ανάλυση των βασικών σκελών της, τόσο δηλαδή το σχεδιαστικό όσο και το κομμάτι της ανάπτυξης, αλλά και να μεταφέρει στον αναγνώστη, τα προβλήματα τα οποία επιλύει το συγκεκριμένο σύστημα. Στην προσπάθεια αυτή, θα υπάρξουν παραπομπές σε κείμενα και έρευνες τρίτων, τα οποία θα αναφέρονται πάντα δύο φορές. Μία στο εσωτερικό του κειμένου της διπλωματικής εργασίας με τη μορφή IEEE[1] —ώστε να μην χάνει τη ροή του ό αναγνώστης— αλλά ταυτόχρονα και στο τέλος της διπλωματικής εργασία, με πλήρεις λεπτομέρειες της αναφοράς, ώστε ο εκάστοτε αναγνώστης να μπορέσει να ανατρέξει στην αναφορά αυτή για εκτενέστερη πληροφόρηση.

Επί των κεφαλαίων, πιο συγκεκριμένα, το **Κεφάλαιο 1** αποτελεί μία πρώτη επαφή, καθώς στοχεύει στην εισαγωγή της ιδέας —παρουσιάζοντας βασικούς όρους και έννοιες συνοπτικά— και πως η συγκεκριμένη σουίτα μπορεί να βοηθήσει σε ζητήματα συνεργατικότητας που δημιουργούνται μεταξύ μελών μίας ομάδας, είτε ακόμα και σε προσωπικό επίπεδο οργάνωσης.

Στο **Κεφάλαιο 2**, θα υπάρξει αναφορά στους τρόπους αντιμετώπισης των προβλημάτων που θέτει για επίλυση η εν λόγω εργασίας, καθώς και την μεθοδολογία που ακολουθείται για να έρθουν εις πέρας. Επίσης, θα γίνει αναφορά σε σχετικούς αλγόριθμου, θεωρίες και μοντέλα που ακολουθήθηκαν ή εφαρμόστηκαν για να γίνει δυνατή η επίλυση θεμάτων που προέκυψαν στις διαδικασίες.

Στο αμέσως επόμενο **Κεφάλαιο 3**, γίνεται μια σύντομη παρουσίαση της διαδικτυακής εφαρμογής αλλά και της επέκτασης περιηγητή, με επίκεντρο τις βασικές λειτουργίες τους, ενώ στο **Κεφάλαιο 4** γίνεται λεπτομερείς αναφορά και ανάλυση και στους δύο (2) αυτούς μηχανισμούς. Επίσης, αναλύονται τα βασικά μέρη κάθε υλοποίηση, με την βοήθεια ενός γενικού σεναρίου, που έχει ως στόχο την εξοικείωση του αναγνώστη με όλα τα κομμάτια και τις δυνατότητες του συστήματος. Στην προσπάθεια αυτή, θα γίνει χρήση εικόνων καθώς και αποσπασμάτων πηγαίου κώδικα, ώστε να υπάρξει διαφάνεια μεταξύ των εκάστοτε λειτουργιών και τον συστημάτων που τα απαρτίζουν.

Τέλος, η παρούσα αναφορά ολοκληρώνεται με το **Κεφάλαιο 5**, με μία σύνοψη στα οφέλη της συγκεκριμένης πτυχιακής εργασίας, καθώς και στα επιτεύγματα μέσα από την

διαδικασίας ολοκλήρωσης αυτής, τόσο σε προσωπικό επίπεδο όσο και στο συνολικό αντίκτυπο.

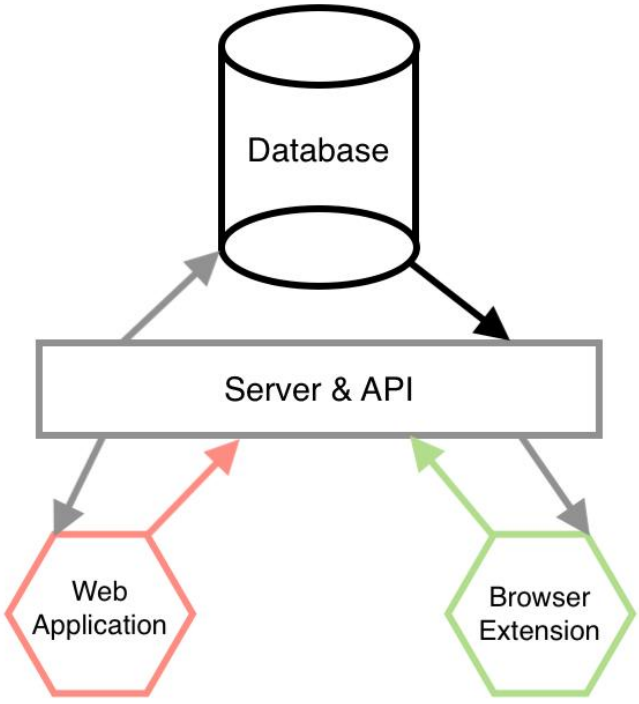
2 Μεθοδολογία Υλοποίησης

Στην ενότητα αυτή επιχειρούμε μία συνοπτική περιγραφή της μεθοδολογίας που ακολουθείται και στην αντιμετώπιση ζητημάτων που προέκυψαν κατά τον σχεδιασμό και την ανάπτυξη του συγκεκριμένου οικοσυστήματος. Επίσης, θα γίνει αναφορά στις βασικές θεωρητικές και τεχνολογικές έννοιες που σχετίζονται με την εν λόγω εργασία, με αρχή το γενικότερο αντικείμενο της οργάνωσης μελών μίας ομάδας και παρακάτω εξειδικεύοντας στα επιμέρους ζητήματα άμεσα συνδεδεμένα με την οργάνωση πόρων και εύκολη ανάκτηση τους.

2.1 Δομή Εργασίας & Μέθοδος Ανάλυσης

Η βασική δομή της εργασίας αποτελείται από τέσσερα (4) κύρια μέρη, το καθένα με τη δική του ανάγκη για ξεχωριστή μελέτη και έρευνα. Η δομή αυτή προέκυψε ύστερα από μελέτη των απαιτήσεων, στόχων αλλά και ζητουμένων βάσει των προβλημάτων που είχαν τεθεί στο στόχαστρο επίλυσης. Τα μέρη αυτά αναφέρονται επιγραμματικά παρακάτω, ενώ θα αναλυθούν σε ξεχωριστές υπό-ενότητες, αναπτύσσοντας τον τρόπο σκέψης και προσέγγισης που ακολουθήθηκε για κάθε ένα ξεχωριστά, ώστε να γίνει η ανάλογη επιλογή τεχνολογιών για κάθε ένα εξ ’αυτών:

- Βάση δεδομένων
- Εξυπηρετητής & Διεπαφή Προγραμματισμού Εφαρμογών (Server & API)
- Διαδικτυακή εφαρμογή
- Επέκταση περιηγητή



Εικόνα 1: Βασική δομή εργασίας

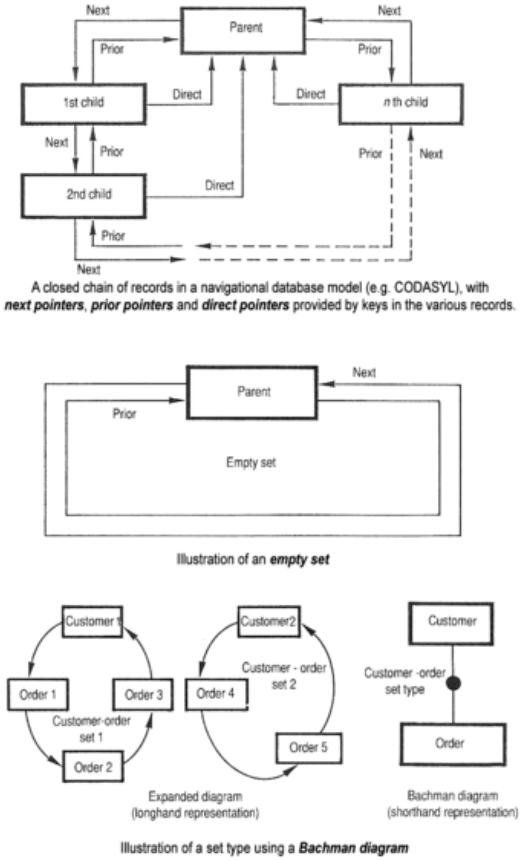
2.1.1 Βάση δεδομένων

Η ανάγκη για αποθήκευση μεγάλων όγκων δεδομένων υπήρξε από τα πρώτα βήματα του διαδικτύου ενώ οι τρόποι με τους οποίους αρχικά γινόταν αυτό, δεν είχε κάποια συγκεκριμένη δομή ή μορφή, αλλά προσαρμοζόταν ανάλογα της περίπτωσης και της εκάστοτε ανάγκης, κάνοντας δύσκολη πολλές φορές την περαιτέρω ανάπτυξη τους. Η ταχύτητα αλλά και η πολυπλοκότητα ήταν οι δύο (2) κυριότεροι λόγοι που οδήγησαν στην περαιτέρω μελέτη τους ενώ η τεχνολογική ανάπτυξη τους, μπορεί να χωριστεί σε τρεις (3) περιόδους – εποχές:

- Πλοηγούμενες (Navigational)
- Σχεσιακές (SQL/relational)
- Μετά-σχεσιακές (NoSQL/Document-oriented)

2.1.1.1 Πλοηγούμενες βάσεις δεδομένων

Η αρχαιότερη χρονολογικά μορφή βάσης δεδομένων με την δυνατότητα άμεσης προσπέλασης του αποθηκευτικού χώρου, επιτρέποντας διαμοιραζόμενα διαδραστική προσπέλαση του, σε αντίθεση με την ακόμα αρχαιότερη προσέγγιση της διαδικασίας επαναυπολογισμού σταθερών για τον χρήστη δεδομένων/τιμών. Η πρώτη τέτοιου τύπου «για εμπορική χρήση» προσέγγιση έγινε από τον Charles Bachman [2], Αμερικάνος επιστήμονας υπολογιστών, ο οποίος ανέπτυξε το IDS (Integrated Data Store)[3] για τις ανάγκες της CODASYL (Conference/Committee on Data Systems Languages)[4].



Εικόνα 2: Βασική δομή μοντέλου της πλουηγούμενης βάσης της CODASYL

Η συγκεκριμένη υλοποίηση, επέτρεπε την προσπέλαση ενός συνδεδεμένου συνόλου δεδομένων που σχημάτιζε μία μορφή δικτύου με δεδομένα. Οι εφαρμογές θα μπορούσαν να προσπελάσουν/ανακτήσουν δεδομένα ή πληροφορίες με μία από τις παρακάτω τρεις (3) μεθόδους:

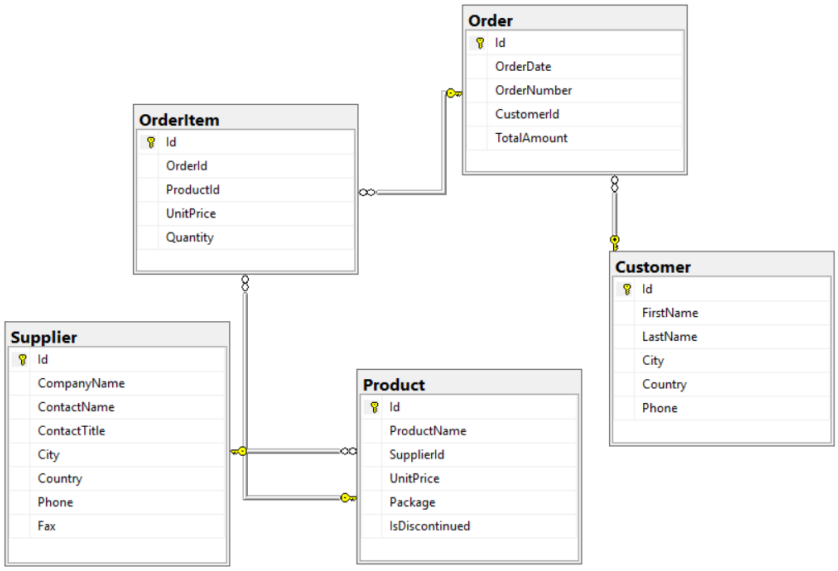
1. Χρήση κύριου κλειδιού —τυπικά σχεδιασμένο με τη χρήση hash[5]
2. Πλοήγηση μεταξύ συσχετίσεων (ονομαζόμενα sets) πηγαίνοντας από τη μία καταγραφή στην επόμενη
3. «Σκανάροντας» όλες τις καταγραφές σειριακά

Η συγκεκριμένη μορφή γρήγορα άρχισε να γίνεται ολοένα και πιο περίπλοκη λόγω των νέων αναγκών που προκύπταν σταδιακά και τον μεγάλων μοντέλων που αναπτύχθηκαν μέσω της νέας αυτής προσέγγισης.

2.1.1.2 Σχεσιακές (SQL/relational)

Οι σχεσιακές βάσεις ήρθαν να λύσουν το πρόβλημα αυτό με τη χρήση συστημάτων με πολλαπλούς πίνακες όπου έδινε την δυνατότητα ακόμα και σε καταγραφές που σχετιζόταν μεταξύ τους να «σπάσουν» σε διαφορετικούς πίνακες και να οργανωθούν με διαφορετικό κριτήριο, απομυθοποιώντας την διαδικασίας της εποχής. Ένα πρότυπο αυτής της λογικής εμφανίστηκε κοντά στο 1974, βασισμένο στη μελέτη του Edgar F. Codd[6] τότε υπάλληλο της IBM[7].

Η συγκεκριμένη υλοποίηση, τότε γνωστή ως System R[8], έδινε την ευελιξία της αποκοπή δεδομένων ίδιας σημασίας και σημαντικότητας, σε μικρούς οργανωμένου πίνακες. Σχέσεις που αναπτύσσονταν μεταξύ των πινάκων πρόσφεραν την δυνατότητα με απλά ερωτήματα/query προς το σύστημα/βάση να εξάγονται συμπεράσματα για τα δεδομένα αυτά σα να ήταν ένας ενιαίος άρρηκτα συνδεδεμένος πίνακας.

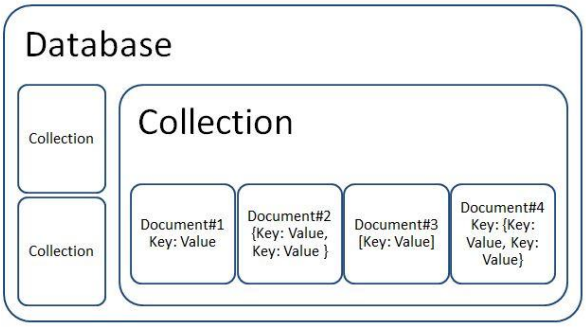


Εικόνα 3: Βασική δομή μοντέλου σχεσιακής βάσης SQL

Παρόλο που η System R, ξεκίνησε σαν ένα πείραμα στην IBM πολύ γρήγορα αναπτύχθηκε και έγινε το νέο στάνταρ σαν μία query language[9] και ονομάστηκε SQL. Η συγκεκριμένη λύση, θεωρήθηκε ανώτερη της CODASYL, εκτοξεύοντας την IBM. Επίσης υπήρχαν άτομα όπως τον Larry Ellison[10] και τον Michael Stonebraker[11] οι οποίοι δανείστηκαν την βάση —System R— και προχώρησαν σε δικές τους SQL προσεγγίσεις την Oracle V1 και την PostgreSQL αντίστοιχα.

2.1.1.3 Μετά-σχεσιακές (NoSQL/Document-oriented)

Με την ανάπτυξη του προγραμματισμού, κυρίως εντός του 20ου αιώνα, ήρθε στην επιφάνεια η ανάγκη αποθήκευσης δεδομένων σε μορφή συλλογών αποτελούμενες από έγγραφα, με δομές που μπορεί να ποικίλουν και με πολλή μεγάλη ευελιξία. Μία πρώτη μορφή τέτοιου είδους προσέγγισης, ήταν η XML[12] που με τη χρήση γνωρισμάτων (document attributes) μπορούσες να οργανώσεις σε ένα έγγραφο πολλαπλές καταγραφές με πλήρη ευελιξία. Σε δεύτερο χρόνο, λύσεις όπως τη MongoDB, Cassandra, Redis και Neo4j έκαναν την εμφάνιση τους, καλύπτοντας όλες τις απαιτήσεις της συγκεκριμένης ανάγκης για μία αποκεντρωμένη και ευέλικτη αποθήκευση δεδομένων.



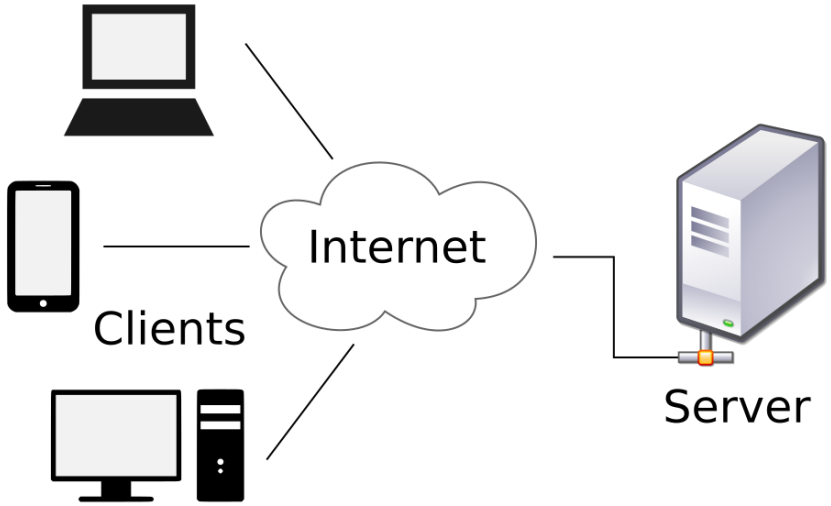
Εικόνα 4: Βασική δομή NoSQL/Document-oriented βάσης

2.1.2 Εξυπηρετητής & Διεπαφή Προγραμματισμού Εφαρμογών (Server & API)

Τόσο ο εξυπηρετητής/server όσο και η διεπαφή προγραμματισμού εφαρμογών —εφεξής και για συντομία API(Application Programming Interface) ή Web API— είναι δύο ξεχωριστές οντότητες που αναπτύχθηκαν ανεξάρτητα και σε διαφορετικές χρονικές περιόδους.

2.1.2.1 Εξυπηρετητής/Server

Η έννοια του «εξυπηρετητή» —στο κόσμο των υπολογιστών— χρονολογείται πίσω στο 1965, σε έγγραφο[13] σχετικό με το ARPANET[14], τον πρόγονο του σημερινού διαδικτύου. Η σημερινή του έννοια, μία συσκευή ή ένα πρόγραμμα ή ακόμη και τα 2 που παρέχουν λειτουργικότητα/εξυπηρέτηση/διευκόλυνση σε άλλα προγράμματα ή συσκευές γνωστά ως «clients»[15]. Γενικότερα, ένας server/εξυπηρετητής έχει την ιδιότητα να παρέχει διάφορες λειτουργίες/υπηρεσίες, για την διευκόλυνση του/των εκάστοτε client/s, όπως τον διαμοιρασμό δεδομένων μεταξύ τους ή να κάνει υπολογισμούς που θα ήταν κόστος να γίνουν από αυτούς, αποτρέποντας επίσης το να γίνουν παραπάνω από μία φορά, καθώς το πραγματοποιεί ο ίδιος για όλους και μοιράζεται το αποτέλεσμα με τους ενδιαφερόμενους.



Εικόνα 5: Βασική δομή επικοινωνίας server-client

Η έννοια του server χωρίζεται σε δύο (2) κομμάτια. Το φυσικό κομμάτι —τη συσκευή αυτή καθαυτή ή αλλιώς hardware— το οποίο και δεν θα αναλυθεί στη συγκεκριμένη εργασία καθώς η επιλογή της συσκευή ήταν καθαρά παθητική ενέργεια στη συγκεκριμένη περίπτωση, και το κομμάτι του περιβάλλοντος που είναι και αυτό που μας απασχολεί. Η ενότητα του περιβάλλοντος, εφεξής για διευκόλυνση web server environment ή web server, μπορεί να χωριστεί τόσο βάσει αναγκών όσο και με γνώμονα τη γλώσσα προγραμματισμού που θα καλείται να χρησιμοποιηθεί για την ανάπτυξή του. Οι πιο διαδεδομένες επιλογές είναι οι δύο (2) παρακάτω:

- Apache
- Node.js

Και οι δύο αυτές επιλογές είναι υπηρεσίες ανοιχτού κώδικα (open-source[16]) και έχουν τεράστιες κοινότητες που στηρίζουν τη κάθε μία προσέγγιση χωριστά και σε βάθος. Ο Apache web server, γνωστό και σαν Apache HTTP Server Project[17] υποστηρίζει μία ευρύς γκάμα προγραμματιστικών γλωσσών όπως Perl, Python, PHP ή Tcl, ενώ υπάρχουν άπειρα modules με ήδη υλοποιημένες λογικές που διευκολύνουν κατά πολύ την ανάπτυξη περίπλοκων server δίνοντας σε όσους αναπτύσσουν web servers τη δυνατότητα να εστιάσουν στα πραγματικά ζητούμενα/ανάγκες της υλοποίησης τους και όχι σε ζητήματα όπως ασφάλειας και caching[18].

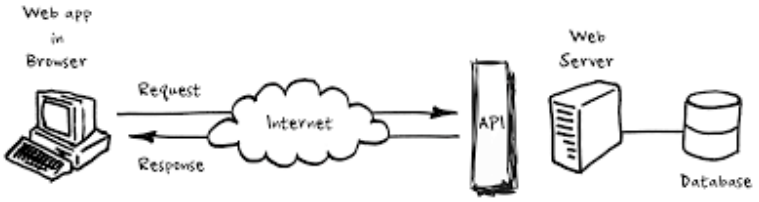
Το Node.js εμφανίστηκε πολύ αργότερα, στα τέλη του 2009, μία περίοδος που η γλώσσα προγραμματισμού JavaScript[19]—η οποία θα αναλυθεί περαιτέρω παρακάτω σε ξεχωριστή υπό-ενότητα—ήταν ήδη πολύ δημοφιλής, και «έσπασε» τα δεδομένα της εποχής, βγάζοντας τη γλώσσα αυτή εκτός του browser που έως τότε ήταν αδύνατον να συμβεί. Η JavaScript συγκεκριμένα, ήταν γλώσσα προγραμματισμού και μία εκ των τεχνολογιών «πυρήνες» του παγκόσμιου ιστού που αρχικά χρησιμοποιούνταν για την ανάπτυξη διαδραστικών σελίδων στο διαδίκτυο.

Με την εμφάνιση του Node.js, αυτό έπαψε να ισχύει, καθώς κάνοντας χρήση του V8 JavaScript engine[20], κατάφερε να τρέξει και εκτός περιβάλλον περιηγητή, και έδωσε τη δυνατότητα στη γλώσσα να αναπτυχθεί περαιτέρω, αλλά και σε χρήστες της, την ευκαιρία να γνωρίσουν το κόσμου του backend — γνωστό και ως ανάπτυξη σε επίπεδο προσπέλασης δεδομένων— χωρίς να χρειάζεται να εξοικειωθούν με άλλες γλώσσες προγραμματισμού.

Σύντομα μετά την κυκλοφορία του, δημιουργήθηκε το npm[21] ένας διαχειριστής πακέτων ο οποίος ξεκλείδωσε τη δυνατότητα σε όλο το κόσμο του Node.js να συντάξει τα δικά του πακέτα/κομμάτια κώδικα, και μέσω του μηχανισμού αυτού να τα κάνει διαθέσιμα σε όσους χρήστες που αντίστοιχα το χρησιμοποιούσαν. Έτσι, πολύ γρήγορα υπήρξε μία τεράστια σουίτα από έτοιμα κομμάτια κώδικα και λογικής γνωστά ως node modules, που έκανα εύκολη τη διαδικασία ανάπτυξης περίπλοκων backend (και όχι μόνο) εφαρμογών.

2.1.2.2 Web API

Η ανάγκη της επικοινωνία ενός web server που διατηρεί μία βάση δεδομένων και ένα πηγαίο κώδικα προγραμματισμένο να εκτελεί συγκεκριμένες διεργασίες, με έναν χρήστη/client ήταν αυτή που έφερε στην επιφάνεια τη έννοια του Web API. Ουσιαστικά, ο ρόλος ενός Web API, είναι να εκθέτει στο frontend κομμάτι μίας εφαρμογής, μόνο όσα είναι απαραίτητα για την σωστή λειτουργία της εφαρμογής και την σωστή εξυπηρέτηση του χρήστη, ενώ παράλληλα «κρύβει» όσα δεν πρέπει να είναι διαθέσιμα σε κανένα άλλο πέραν αυτού που αναπτύσσει τη συγκεκριμένη εφαρμογή.



Εικόνα 6: Βασική δομή Server – API – Client

Η επικοινωνία αυτή επιτυγχάνεται με την δημιουργία και χρήση των endpoints που έχουν το ρόλο ενός διαύλου επικοινωνίας μεταξύ των δύο κόσμων. Τα endpoints[22], ορίζουν μία συγκεκριμένη γλώσσα επικοινωνίας μεταξύ της ζήτησης – απάντησης συνήθως χρησιμοποιώντας τη μορφή JSON[23] ή XML.

2.1.3 Διαδικτυακή εφαρμογή

Με τον όρο διαδικτυακή εφαρμογή, αναφερόμαστε στην διεπαφή αλλά και λογική ενός προγράμματος, το οποίο διανέμεται εντός ενός περιηγητή/browser, χωρίς να απαιτείται κάποια εγκατάσταση από τον χρήστη της. Η έννοια αύτη διαφοροποιείται από αυτή μίας απλά δυναμικής σελίδας, καθώς ο στόχος τους διαφέρει. Η δυναμική σελίδα έχει ο στόχο απλά την καλύτερη εμπειρία σε μία σελίδα για ένα χρήστη, ενώ μία διαδικτυακή εφαρμογή, εξυπηρετεί και είναι χτισμένη με επίκεντρο μία λογική και προσφέρει στο χρήστη ένα σύνολο υπηρεσιών. Μία διαδικτυακή εφαρμογή, είναι ταυτόχρονα και δυναμική σελίδα, ενώ το ανάποδο δεν ισχύει πάντοτε.

Η ανάπτυξη διαδικτυακών εφαρμογών γίνεται πλέον κυρίως με τη χρήση HTML5, JavaScript και CSS3. Αυτές οι τρεις (3) τεχνολογίες είναι ο πυρήνας, όμως για την διευκόλυνση της ανάπτυξης τους, ανοιχτές βιβλιοθήκες τρίτων βασισμένες στις τρεις (3) αυτές αρχές χρησιμοποιούνται κατά κόρον. Αυτό γίνεται, τόσο για την πιο ασφαλείς και γρήγορη ανάπτυξη τους, αλλά και για την πιο ομοιόμορφη διάπλαση μία μεγάλης και περίπλοκης εφαρμογής.

2.1.3.1 HTML5

Με τον όρο HTML[24] ορίζεται ένα σετ από συγκεκριμένα πρότυπα σήμανσης και συμπεριφορών που με τη σειρά τους ορίζουν το περιεχόμενο μιας διαδικτυακής σελίδας. HTML5 είναι η πέμπτη ($5^{η}$) έκδοση της, όπου και θεωρείται σημαντική ενημέρωση όπου είχε ως σκοπό να υποστηρίξει της περαιτέρω ανάγκες των πολυμέσων στο διαδίκτυο αλλά και να εισάγει νέες δυνατότητες. Ταυτόχρονα, να κρατήσει τη μορφή της κατανοητή από τους υπολογιστές και τις συσκευές, χωρίς να περιπλέξει τα πράγματα για τους προγραμματιστές και να είναι συμβατή και με τις παλιότερες εκδόσεις της.

2.1.3.2 JavaScript

Η JavaScript συχνά αναφερόμενη και ως JS είναι μία scripting[25] γλώσσα προγραμματισμού που μαζί με την HTML και τη CSS αποτελούν τις τεχνολογίες «πυρήνες» του παγκόσμιου ιστού. Η συγκεκριμένη γλώσσα δίνει τη δυνατότητα προσθήκης διαδραστικότητας σε μία σελίδα, ενώ βιβλιοθήκες για την πιο απλή και κατανοητή σύνταξη της, έκαναν πολύ γρήγορα την εμφάνιση τους. Μία εξ ‘αυτών, η jQuery η οποία είχε ως στόχο την απλοποίηση κανόνων ή ακόμα και τον συνδυασμό πολλαπλών κανόνων, δημιουργώντας ένα νέο σετ κανόνων πιο κατανοητό για τον τελικό χρήστη. Η επανάσταση που έφεραν βιβλιοθήκες σαν αυτή, ώθησαν όλο και περισσότερο κόσμο να ασχοληθεί με τη χρήση τους, όπου ήταν και ένας από τους κύριους παράγοντες εμφάνισης τον Web Applications.

Πολύ γρήγορα η ανάγκη της JavaScript έγινε αισθητή και στην πλευρά του σερβερ, όπου έγινε μία πρώτη προσπάθεια με την JScript σε ASP και .NET σελίδες από την Microsoft, χωρίς μεγάλη επιτυχία, ενώ την καλύτερη υλοποίηση ήρθε να δώσει το Node.js στα τέλη του 2009.

2.1.3.3 CSS3

CSS3 είναι η τρίτη ($3^{η}$) έκδοση της γλώσσας μορφοποίησης CSS[26]. Χρησιμοποιείται για την μορφοποίηση εγγράφου γραμμένου σε γλώσσες σήμανσης, όπως HTML. Είναι μία από τις γλώσσες «πυρήνας» του διαδικτύου και είναι πλέον το στάνταρ σε όλους τους browsers. Για την ακρίβεια, το CSS2.1 είναι αυτό που προτείνεται σαν χρήση και είναι ασφαλές για όλους τους browsers, ενώ το CSS3 τεμαχίζεται σταδιακά σε μικρότερα κομμάτια και περνάει την διαδικασία της ένταξης σαν το νέο στάνταρ. Κάθε browser ξεχωριστά, θα πρέπει να φροντίσει να μπορεί να τρέξει τα νέα δεδομένα που θέτει αυτή η τρίτη ($3^{η}$) έκδοση της γλώσσας, πριν μπορέσει να θεωρηθεί το νέο standard.

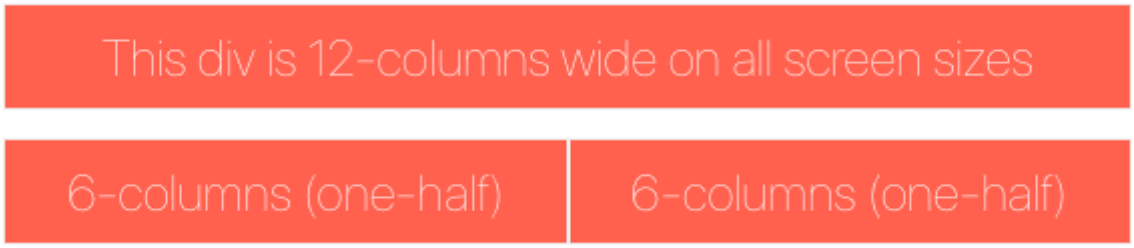
Καθώς η ανάπτυξη τόσο διαδικτυακών εφαρμογών, όσο και δυναμικών ιστοσελίδων έχει γίνει αρκετά περίπλοκη και απαιτητική, βιβλιοθήκες από την πλευρά της μορφοποίησης (CSS) αλλά ακόμα και από την πλευρά της διαδραστικότητας (JavaScript) έχουν κάνει την εμφάνιση του, με σκοπό την διευκόλυνση του developer αλλά και την ταχύτερη ανάπτυξη μία ιδέας χωρίς «τριβές» σε βασικά ζητήματα που έρχονται να προσφέρουν λύση σχεδόν χωρίς κόπο.

Στο στρατόπεδο της μορφοποίησης, βιβλιοθήκες όπως το **Bootstrap** ή το **materialize** είναι αυτές που έρχονται να δώσουν τη λύση σε τέτοιου είδους προβλήματα. Και οι δύο αυτές βιβλιοθήκες είναι ανοιχτού κώδικα και είναι σχεδιασμένες να προσφέρουν «responsive» σελίδες, γνωστές ως mobile-first. Με τον όρο αυτό νοείται η δυνατότητα—πλέον—του προγραμματιστή σχεδιάζοντας μία φορά την εφαρμογή να μπορεί να τρέξει ακόμα και σε κινητές συσκευές χωρίς να πρέπει να προβλέψει ξεχωριστά για κανόνες και σχεδιασμό.

Και οι δύο παραπάνω λύσεις προσφέρουν χαρακτηριστικά όπως Typography[27], φόρμες, κουμπιά, μενού – πλοηγούς ενώ από τη μία το Bootstrap είναι βασισμένο σε δικές του αρχές σχεδίασης, αντίθετα με την materialize βιβλιοθήκη η οποία είναι βασισμένη στους κανόνες που έχει θεσπίσει η google για το Material Design pattern. Επίσης, είναι άξιο αναφοράς ότι η βιβλιοθήκη αυτή χτίστηκε από τέσσερις (4) συμφοιτητές του πανεπιστημίου Carnegie Mellon[28]. Συγκεκριμένα, η materialize είναι από τις πρώτες βιβλιοθήκες που ακολούθησαν την αρχή αυτή και δημιούργησαν μία τόσο μεγάλη σουίτα με χαρακτηριστικά τηρώντας πιστά τις αρχές του Material Design.

Ένα από τα δυνατά χαρτιά και των δύο (2) αυτών βιβλιοθηκών, είναι τα Grid patterns που έχουν σχεδιάσει, που στην πραγματικότητα σου προσφέρουν μια γραμμή με δώδεκα (12) **πάντα** ίσες στήλες, ανεξάρτητα του διαθέσιμου χώρου. Το μαγικό του συγκεκριμένου συντακτικού, είναι ότι μπορείς για τον ίδιο χώρο να ορίσεις διαφορετικό αριθμό στηλών ανά συσκευή. Επομένως, ανάλογα τη συσκευή να διαμορφώνεται και ο χώρος στο επιθυμητό μέγεθος. Έτσι, για παράδειγμα, ένα χώρος που στην οθόνη στου υπολογιστή θέλεις να χωριστεί σε δώδεκα (12) ίσες στήλες, μπορείς ταυτόχρονα να δηλώσεις ότι θέλεις σε tablet συσκευή αυτές να γίνουν δύο (2) ενώ σε κινητή συσκευή να υπάρχει μόνο μία (1). Όπως φαίνεται και παρακάτω, αυτό χάρης τις βιβλιοθήκες αυτές, επιτυγχάνεται χωρίς πολύ κόπο.

Εικόνα 7: Βασική δομή Grid – Row – Column design για Desktop



Εικόνα 8: Βασική δομή Grid – Row – Column design για mobile (πάνω) και για table device (κάτω)

Αντίστοιχα ο κώδικας για τα παραπάνω:

```
<div class="row">
   <div class="col s12 m6 l1">1</div>
</div>
```

Στο συγκεκριμένο παράδειγμα κώδικα, δείχνουμε μόνο πως αντιμετωπίζεται ένα μόνο column, ενώ αντίστοιχα (με τη γραμμή δύο του κώδικα) θα πρέπει να γίνει για όλες τις στήλες που θέλουμε να διαμορφώσουμε. Με τον κανόνα s12 δηλώνουμε ότι το συγκεκριμένο αντικείμενο θέλουμε να πιάσει και τους δώδεκα (12) διαθέσιμους χώρους σε μικρές συσκευές (κινητά) και να θεωρηθεί σαν ένας χώρος, ενώ με το m6 δηλώνουμε ότι θέλουμε σε medium devices να καταληφθούν έξι (6) χώροι ως ένας (1). Τέλος, με το l1 (large 1) δηλώνουμε την χρήση μίας στήλης για τη ίδια δουλεία σε χώρους large (οθόνη υπολογιστή/λάπτοπ).

2.1.4 Επέκταση περιηγητή

Η ανάγκη για επεκτάσεις περιηγητών ήρθε πολύ αργότερα στην επιφάνεια. Είναι μικρά αυτόνομα προγράμματα, που έρχονται να ενισχύσουν την εμπειρία που προσφέρει ένας περιηγητής και στοχεύουν στην καλύτερη εμπειρία ενός χρήστη με το διαδίκτυο. Έκαναν την εμφάνιση τους το 2004, πρώτη φορά στον περιηγητή Firefox, ενώ στον Chrome εμφανίστηκαν αρκετά αργότερα, στις αρχές του 2009.

Είναι μικρές αυτόνομες σελίδες, που απαρτίζονται από HTML, CSS και JavaScript και δρουν σαν ένα μικρό διαδραστικό Web Application, που έχει τη δυνατότητα να επικοινωνεί με τα ανοιχτά παράθυρα του χρήστη, παίρνοντας και στέλνοντας πληροφορίες σχετικές με το περιεχόμενο τους.

2.2 Μέθοδος Ανάπτυξης

Με γνώμονα την παραπάνω δομή και ανάλυση κάθε ενότητας αλλά και της ανάγκες που έχει ως στόχο να καλύψει η εν λόγω πτυχιακή εργασία, έγιναν οι απαραίτητες επιλογές στις τεχνολογίες που θεώρησα ότι θα προσφέρουν την κατάλληλη ευελιξία για την πιο άμεση και σωστή επίλυση των προβλημάτων.

Ο κεντρικός server—χτισμένος σε Node.js—θα παρέχει ένα API επικοινωνίας για τις διεργασίες που απαιτούνται από την διαδικτυακή εφαρμογή και την επέκταση περιηγητή. Θα δίνει τη δυνατότητα αποθήκευση και ανάκτησης δεδομένων από μία βάση PostgreSQL, ενώ παράλληλα, θα αποτρέπει την απευθείας επικοινωνία του χρήστη με αυτή, προσθέτοντας ένα σκαλοπάτι ασφαλείας.

Τόσο η διαδικτυακή εφαρμογή όσο και η επέκταση περιηγητή, θα αναπτυχθούν με τη χρήση JavaScript, HTML5 αλλά και CSS3. Παράλληλα οι βιβλιοθήκες jQuery και materialize, θα βοηθήσουν στην σωστή προσέγγιση προβλημάτων, όπως responsiveness, ασύγχρονες επικοινωνίες, ασφαλείς μεταφορές δεδομένων, user experience και ταχύτητας.

3 Σχέδιο Δράσης

Σύμφωνα με την παραπάνω έρευνα και παράλληλα με τις ανάγκες της πτυχιακής εργασίας αλλά και πάντα με γνώμονες την ασφάλεια, ταχύτητα και εμπειρία του χρήστη διαμορφώθηκε το τεχνολογικό stack της σουίτας. Οι επιλογές αυτές δεν ήταν μονόδρομος, αντίθετα, οι συνδυασμοί για την συγκεκριμένη υλοποίηση θα μπορούσαν να είναι πολλοί, αλλά η τελική επιλογή και διαχωρισμός βασίστηκε επίσης στις τάσεις τις εποχής. Λαμβάνοντας υπόψη τη ζήτηση τεχνολογιών στην αγορά εργασίας, οι επιλογές αυτές θα βοηθήσουν την ένταξη μου σε αυτή, δίνοντας μου την ευκαιρία να δουλέψω σε ένα βαθμό τις δυνατότητες τους.

3.1 Σχεδιασμός και Υλοποίηση συστήματος

Καθώς ένα μεγάλο κομμάτι της λύσης αυτής βασίζεται στην συνεχή επικοινωνία του χρήστη με τον server, η επιλογή του σωστού εξυπηρετητή είναι πολύ σημαντική καθώς αυτός θα είναι υπεύθυνος για όλο το “heavy lifting”. Πολύ σημαντική είναι επίσης και η δομή της βάσης (database) καθώς μία σωστά δομημένη και γεμάτη εξαρτήσεις βάση θα γλυτώσει αρκετούς μπελάδες στην ανάκτηση περίπλοκων και πολλαπλά συνδεδεμένων αποτελεσμάτων.

Από την πλευρά του χρήστη η δουλεία δεν είναι πιο απλή. Αντίθετα, περιπλέκει καθώς θα πρέπει δύο διαφορετικές εφαρμογές (Web Application & Browser Extension) να δέσουν σαν μία, επιστρέφοντας στο χρήση μία συνολική εμπειρία, διαισθητική και πολλή γρήγορη σε πλοήγηση. Τα δύο (2) αυτά ξεχωριστά applications θα πρέπει να ταιριάζουν και να δίνουν στο χρήστη την ιδέα ότι βλέπει κάτι γνώριμο, κάνοντας αυτό που προσφέρουν ακόμα πιο εύκολο.

3.1.1 Backend – Υλοποίηση και σχεδιασμός Server & API

Ξεκινώντας με τον server, η επιλογή του **Node.js** μου ήταν εξαρχής δελεαστική, λόγω της μεγάλης κοινότητας που έχει αναπτυχθεί και του εδάφους που έχει κερδίσει σε πολύ μικρό χρονικό διάστημα. Δεδομένου ότι πρόκειται για μία απίστευτα ασφαλείς και σταθερή υλοποίηση, και σε συνάρτηση με το γεγονός ότι η υλοποίηση του απαιτεί τη χρήση της γλώσσας JavaScript η επιλογή του ήταν μονόδρομος. Εφόσον στο frontend πλέον—και στις δυο (2) υλοποιήσεις Web Application και browser extension—η χρήση της JavaScript είναι αναπόφευκτη, η χρήση της και στο backend κομμάτι της εφαρμογής μοιάζει ιδανική, καθώς η υλοποίηση θα δέσει σε ένα ενιαίο πολύ σωστά διαμορφωμένο και εύκολα διαχειρίσιμο source code με βάση την γλώσσα JavaScript.

Η επιλογή μίας σχεσιακής βάσης, λόγω τον ζητουμένων της εργασίας ήταν μονόδρομος, καθώς οι εξαρτήσεις μεταξύ των πολλαπλών καταγραφών είναι απαραίτητη, ενώ τα δεδομένα που θα συλλέγονται θα αναπτύσσουν ένα δεσμό που είναι χρήσιμος να καταγραφεί. Οι επιλογές στις σχεσιακές βάσεις είναι αμέτρητες, όμως η επέλεξε να ακολουθήσω την **PostgreSQL**. Σε σύγκριση με άλλες υλοποιήσεις της SQL, δεν διαφέρει ουσιαστικά στη σύνταξη ή στις επιδόσεις, ενώ προσθέτει/παρέχει κάποιες έξτρα δυνατότητες οι οποίες δεν έχουν ακόμη υλοποιηθεί σε παρόμοιες εκδόσεις. Επίσης, καθώς σε παλαιότερα προτζεκτ/εργασίες της σχολής έχει γίνει χρήση της MySQL και της SQLite, ήταν μία έξτρα ώθηση να δώσω την ευκαιρία στην συγκεκριμένη υλοποίηση και να ανακαλύψω μόνος μου τις διαφορές που υπάρχουν μεταξύ των άλλων δύο (2) υλοποιήσεων. Επίσης, η χρήση της PostgreSQL σε συνδυασμό με το Node.js είναι σύνηθες, καθώς το npm module pg[29] έχει

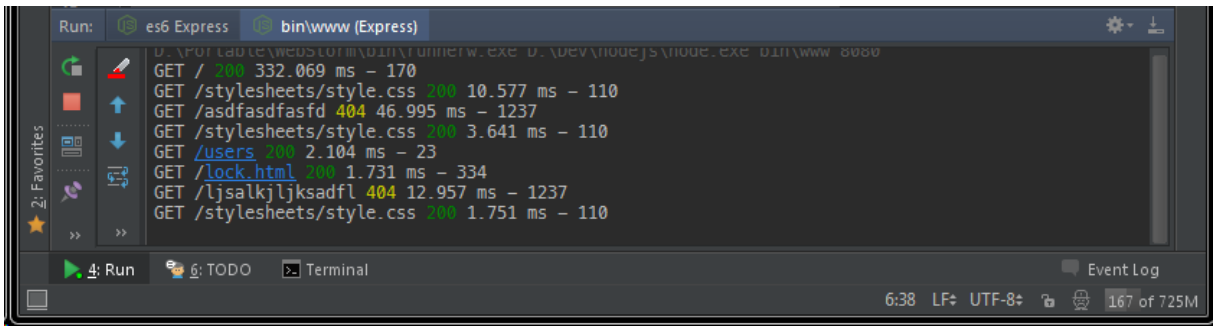
ήδη φροντίσει για πολλά όπως την ομαλή και ασφαλή χρήση/ένταξη της γλώσσας στο οικοσύστημα του Node.js, την βέλτιστη ενσωμάτωση του και την υποστήριξη του σε βάθος με αρκετά μεγάλό και εκτενές υλικό.

3.1.1.1 Node.js

Το Node.js θέτει το έδαφος πάνω στο οποίο θα χτιστεί όλο το backend, ενώ διάφορα node/npm modules έρχονται να ενισχύσουν το Node.js και να δημιουργήσουν μία ολοκληρωμένη λύση που θα παίξει τον ρόλο του εξυπηρετητή στο οικοσύστημα μας. Έχοντας στα θεμέλια το Node.js, ξεκινάμε με την ανάγκη ενός μεσάζοντα (εφεξής middleware) ο οποίος και θα αναλαμβάνει όλα τα ζητήματα (εφεξής requests) προς τον server μας και θα τα διαχειρίζεται ανάλογα. Το ρόλο αυτό θα τον αναλάβει το **Express**. Το Express είναι ένα framework ανοιχτού κώδικα που είναι σχεδιασμένο για να χτίζει και να διανέμει διαδικτυακές εφαρμογές και API.

Το Express θα είναι αυτό που θα έχει τον πρώτο λόγο όταν ένα request καταλήξει στο server μας. Έπειτα, αυτό θα καθορίζει να θα πρέπει αυτό το request να διαχειριστεί από τον ίδιο ή θα πρέπει να ειδοποιήσει κάποιον άλλο για την εκτέλεση του. Όλο το API θα χτιστεί με τη χρήση του συγκεκριμένου πακέτου, ενώ σε δεύτερο χρόνο, όταν υπάρχει ανάγκη για ανάκτηση ή αποθήκευση δεδομένων στη βάση μας, θα ειδοποιείται από το Express το πακέτο pg. Σε επίπεδο Express ξανά, θα οριστούν συγκεκριμένα routes/endpoints που θα γίνουν διαθέσιμα στο frontend, για να μπορεί να επικοινωνεί με ασφάλεια με τη βάση.

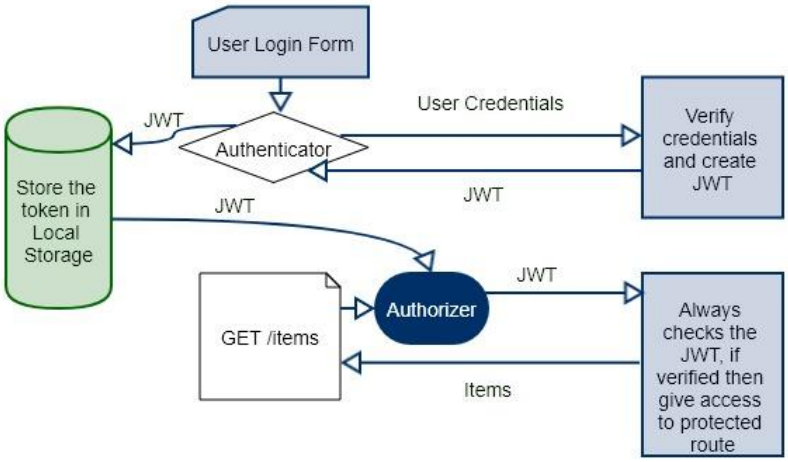
Για την σωστή λειτουργία όλων τον απαιτήσεων που θέτει η εφαρμογή πακέτα/modules όπως τα morgan[30], cookie-parser[31], body-parser[32], express-session[33] και path[34] είναι απαραίτητα. Συνοπτικά, το morgan είναι ένα logger, ο οποίος παρακολουθεί και ενημερώνει τον developer για οτιδήποτε «ύποπτο» παρατηρηθεί στο κομμάτι του σχεδιασμού, ώστε να προλάβει τα χειρότερα. Επομένως, αν κάτι πάει στραβά, αυτό θα είναι που πρώτο θα βοηθήσει και θα κατευθύνει τον developer με λεπτομέρειες για την επίλυση του προβλήματος.



Εικόνα 9: Βασική ένδειξη morgan

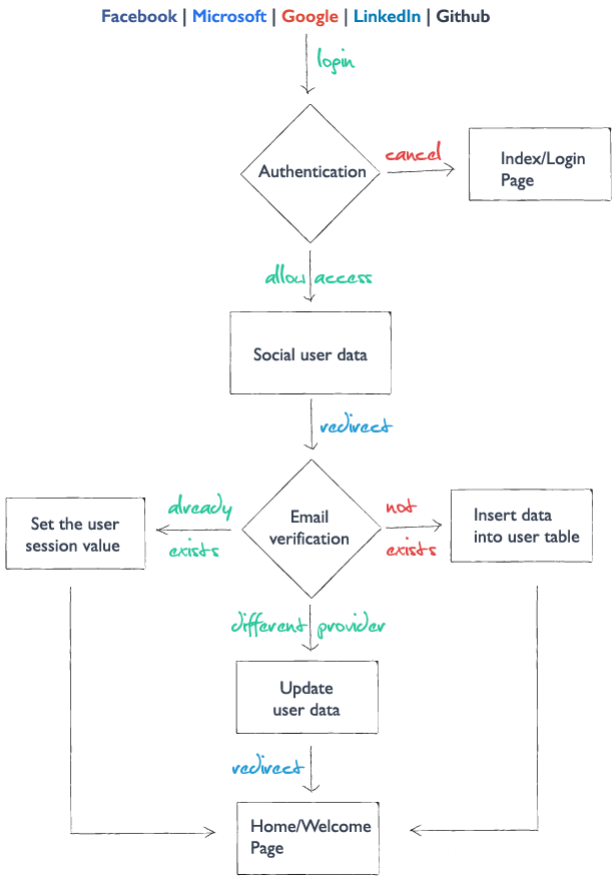
Τα πακέτα cookie-parser και body-parser βοηθάνε στη σωστή κωδικοποίηση και αποκωδικοποίηση συγκεκριμένης πληροφορίας που έρχεται και στέλνετε από και προς τον server. Το πρώτο (cookie-parser) διαχειρίζεται πληροφορίες σχετικά με την κατάσταση και τα δικαιώματα του εκάστοτε χρήστη (αν δηλαδή είναι εξουσιοδοτημένος χρήστης της εφαρμογής ή όχι) αλλά ακόμα και λεπτομέρειες όπως αν ο χρήστης είναι νέος ή επιστρέφει ξανά. Το δεύτερο (body-parser) είναι αυτό που ευθύνεται για τη σωστή μετατροπή της πληροφορίας που θα στείλει ο χρήστης, σε κάτι κατανοητό για τον server περιεχόμενο. Το πακέτο express-session σε συνδυασμό με το framework **passport**[35] είναι αυτά που μαζί θα συντάξουν τη μηχανή εξουσιοδότησης ενός χρήστη. Επομένως, αυτά τα δύο μαζί θα συντάξουν ένα σύνολο κανόνων και δοκιμών και στο τέλος θα βγάζουν το συμπέρασμα για

το αν ο χρήστης είναι απλός guest ή αν έχει δικαιώματα να πάρει και να δώσει πληροφορίες στο σύστημα μας.



Εικόνα 10: Βασική μορφή διαγράμματος εξουσιοδότησης

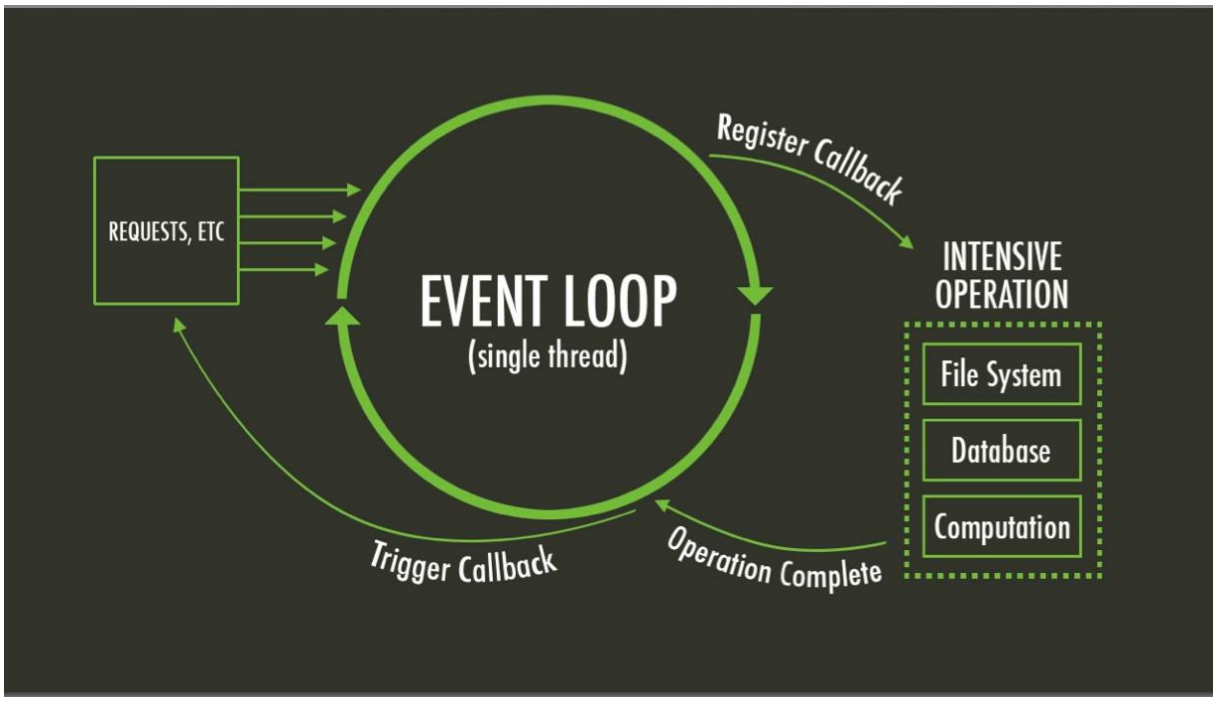
Συγκεκριμένα από το passport θα χρησιμοποιήσουμε μόνο την εξουσιοδότηση λογαριασμού google, καθώς για τις ανάγκες μας μοιάζει ιδανικό. Το passport βέβαια, μας δίνει τη δυνατότητα να επιλέξουμε ένα (1) ή παραπάνω strategies (όπως τα ονομάζει) για την ταυτοποίηση ενός χρήστη. Μερικές από τις επιλογές είναι το Facebook, Twitter, Google ή ακόμα και custom ταυτοποίηση χτισμένη από τον εκάστοτε διαχειριστή. Αυτό δίνει τη ευελιξία σε ένα χρήστη που έχει λογαριασμό σε οποιαδήποτε από τις παραπάνω πλατφόρμες να μπορεί να χρησιμοποιήσει την εφαρμογή μας, χωρίς να απαιτείται κάποια εγγραφή.



Εικόνα 11: Βασική ιδέα σύνδεσης με social media

Το Node.js σαν πυρήνα πίσω από όλα αυτά, φροντίζει όλα να μπαίνουν σε μία σειρά και να εκτελούνται όπως πρέπει χωρίς να μπλοκάρουν διαδικασίες. Συγκεκριμένα, το Node.js δεν λειτουργεί με threads αλλά με ασύγχρονο τρόπο και τη μέθοδο του event loop όπου δεν σταματάει ποτέ να ορίζει διεργασίες χωρίς να περιμένει για απάντηση, αφήνοντας οδηγίες για κάθε μία από αυτές τις διεργασίες για το τί πρέπει να γίνει αφού ολοκληρωθούν. Έτσι, οτιδήποτε και αν έρθει σαν request, αν έχει οριστεί ο τρόπος αντιμετώπισης του, το node θα του δείξει το δρόμο και έπειτα θα συνεχίσει ο εκάστοτε υπεύθυνος να συνεχίσει την διαδικασία.

Ο τρόπος αυτός στην αρχή αμφισβητήθηκε καθώς η έννοια των threads είχε ριζώσει σε αρκετές διαδικασίες και έμοιαζε ο ιδανικός καθώς κάθε thread ήταν υπεύθυνο ξεχωριστά για την διεργασία που του ανατιθόταν. Το Node.js ήρθε να αλλάξει τη λογική αυτή, και να εκμεταλλευτεί το 100% τον πόρων με την συγκεκριμένη προσέγγιση.



Εικόνα 12: Βασική δομή Node.js Server

3.1.1.2 PostgreSQL

Επιλέγοντας την PostgreSQL, μία σχεσιακή μορφή βάσης, μας δίνεται η δυνατότητα να απομονώσουμε τις οντότητες μας για ξεκάθαρη οργάνωση και παράλληλα να αναπτύξουμε δεσμούς μεταξύ των πινάκων για εύκολη ανάκτηση όλων των δεδομένων. Οι οντότητες τις συγκεκριμένης υλοποίησης είναι οι εξής:

- users
- webpages
- collections
- discussions
- comments

Κρατώντας λίγες τις οντότητες και ξεκάθαρη τη σύνδεση μεταξύ τους, το μοντέλο μπορεί να μεγαλώσει χωρίς να δημιουργήσει προβλήματα στην απόδοση και στη εφαρμογή περίπλοκων ερωτήσεων.

Στον πίνακα users θα αποθηκευτούν όλες οι πληροφορίες που είναι σχετικές με τον χρήστη μας και μας είναι απαραίτητες για τη σωστή και αξιόλογη εμπειρία του στη εφαρμογή μας. Αυτές οι πληροφορίες είναι:

*Πίνακας 1: Ανάλυση πίνακα χρηστών **(users)***

email	*Το email του χρήστη*
google_id	*Το μοναδικό αναγνωριστικό του χρήστη στη βάση της Google*
fname	*Το πρώτο όνομα του χρήστη*
lname	*Το επώνυμο του χρήστη*
img	*Η εικόνα προφίλ του χρήστη στο συγκεκριμένο λογαριασμό του στη Google*
u_id	*Μοναδικό αναγνωριστικό αριθμό για τον συγκεκριμένο χρήστη στη δική μας βάση*

Αντίστοιχα, για τους πόρους που θα συλλέγει ο εκάστοτε χρήστης (ή αλλιώς σελίδες – webpages) ένα αντίστοιχος πίνακας θα πρέπει να δημιουργηθεί. Ο πίνακας αυτός θα περιέχει:

*Πίνακας 2: Ανάλυση πίνακα ψηφιακών πόρων **(websites)***

url	*Ο σύνδεσμος του πόρου που μόλις συλλέχθηκε*
usermail	*Το email του χρήστη που σύλλεξε τον πόρο αυτόν. (**Ξένο κλειδί** στον πίνακα των χρηστών για τη τιμή **email**)*
tags	*Λέξεις-κλειδιά που επέλεξε ο χρήστης από πιθανές προτάσεις που βρέθηκαν πάνω στη σελίδα*
labels	*Λέξεις-κλειδιά ορισμένες από τον ίδιο το χρήστη που θέλει να συνδεθούν με τον πόρο αυτό*
site	*Το απόλυτο URL της σελίδας που συλλέχθηκε ο συγκεκριμένος πόρος. Για παράδειγμα, αν συλλέκτικε κάποιο άρθρο από το Wikipedia, εδώ θα κρατηθεί η τιμή https://wikipedia.org/ για ευκολότερο grouping*
thumbnail	*Ένα στιγμιαίο screenshot από τον πόρο που μόλις σύλλεξε ο χρήστης. Τραβηγμένο με τη βοήθεια του extension.*
title	*Ένας τίτλος για τον ψηφιακό πόρο, ορισμένος από τον χρήστη*
favicon	*Το favicon της σελίδας*
timestamp	*Αποθηκεύεται η στιγμή που συλλέχθηκε ο συγκεκριμένος πόρος, για αναφορά στην εφαρμογή.*
url_id	*Ένα μοναδικό ID για τη συγκεκριμένη*

	καταγραφή

Πέραν αυτών των δύο (2) βασικών πινάκων, η εφαρμογή έχει σκοπό να καλύψει την ανάγκη των πολλαπλών ομάδων αλλά και ταυτόχρονα πολλαπλών προτζεκτ. Για την ανάγκη αυτή, δημιουργήθηκε πίνακας «collections» όπου ένας ψηφιακός πόρος μπορεί να ανήκει σε καμία έως πολλές συλλογές, ξεκλειδώνοντας τη δυνατότητα οργάνωσης του πόρου και διαμοιρασμού του με τρίτους, καθώς στη συλλογή μπορεί να συμμετέχουν από ένας (1) χρήστης—ο δημιουργός—έως πολλούς (συνεργάτες). Παρακάτω αναλυτικά:

*Πίνακας 3: Ανάλυση πίνακα συλλογών **(collections)***

c_id	*Μοναδικό αναγνωριστικό συλλογής*
c_name	*Όνομα συλλογή*
owner_email	*Email ιδιοκτήτη συλλογής*
people	*Πίνακας από emails με χρήστες που είναι μέλη της συγκεκριμένης συλλογής*
url_id	*Πίνακας με τα URL ID που υπάρχουν στη συλλογή αυτά (αναφορά σε url_id πίνακα websites)*

Τέλος, καθώς στην συνεργασία μεταξύ ομάδων είναι απαραίτητη η άμεση συνεννόηση, δημιουργήθηκε η ανάγκη συζητήσεων σε επίπεδο πόρων. Με την λειτουργία αυτή, ουσιαστικά δίνεται η δυνατότητα σε όσους διαμοιράζονται ένα πόρο, να μπορούν να αφήσουν σχόλια σε αυτόν. Ακόμα και αν ο πόρος δεν είναι διαμοιραζόμενος βέβαια, η λειτουργία αυτή προσφέρεται στον εκάστοτε κάτοχο για περαιτέρω ανάλυση και προσωπικές σημειώσεις.

Για την εξυπηρέτηση του σκοπού αυτού ήταν απαραίτητη η δημιουργία δύο (2) πινάκων, discussions και comments που περιγράφονται παρακάτω:

*Πίνακας 4: Ανάλυση πίνακα συζήτησης **(discussions)***

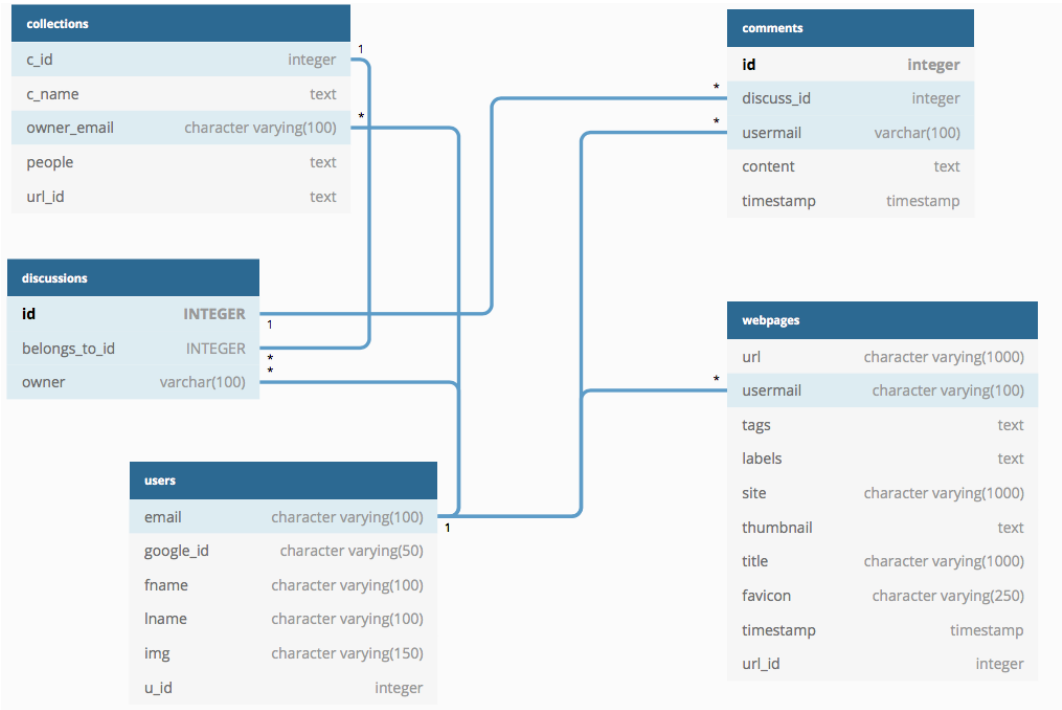
id	*Μοναδικό αναγνωριστικό συζήτησης*
belongs_to_id	*Αναγνωριστικό του πόρου στον οποίο ανήκει η συζήτηση αυτή (**ξένο κλειδί** στο url_id του πίνακα **websites**)*
owner	*Email χρήστη που ανήκει η συγκεκριμένη συζήτηση (**ξένο κλειδί** στο email του πίνακα **users**)*

*Πίνακας 5: Ανάλυση πίνακα σχολίων **(comments)***

id	*Μοναδικό αναγνωριστικό σχολίου*
discuss_id	*Αναγνωριστικό της συζήτησης που ανήκει το εκάστοτε σχόλιο (**ξένο κλειδί** στο id του πίνακα **discussions**)*
usermail	*Email χρήστη που ανήκει το εκάστοτε σχόλιο (**ξένο κλειδί** στο email του πίνακα **users**)*
content	*Περιεχόμενο σχολίου*
timestamp	*Λεπτομέρειες σχετικά με τι στιγμή – χρόνο*

	που έγινε το εκάστοτε σχόλιο

Η τελική μορφή των πινάκων καθώς και των data types που χρησιμοποιήθηκαν για την εκάστοτε τιμή φαίνονται παρακάτω, ενώ μπορείτε επίσης να δείτε τις συσχετίσεις που έχουν αναπτυχθεί μεταξύ των πινάκων που είναι απαραίτητο, για την σωστή σύνδεση/ανάκτηση δεδομένων. Για την εξαγωγή της συγκεκριμένης εικόνας διαγράμματος χρησιμοποιήθηκε η διαδικτυακή εφαρμογή dbdiagram.io[36].



Εικόνα 13: Αναλυτικό διάγραμμα βάσης

3.1.2 Frontend – Υλοποίηση και σχεδιασμός Web Application & Browser Extension

Το frontend της σουίτας, αποτελείται από δύο μεγάλες οντότητες. Το Web Application που θα σερβίρετε εν τέλη σε ένα συγκεκριμένο domain, και το browser extension το οποίο και αφού εγκαταστήσει ο χρήστης με ένα μόνο κλικ, θα βρίσκεται πάντα στον περιηγητή του έτοιμο να δράσει όποτε αυτός το χρειαστεί.

Και οι δύο (2) αυτές υλοποιήσεις έχουν αναπτυχθεί ξεχωριστά και με διαφορετική λογική επομένως και χρήζουν ξεχωριστής ενότητας έκαστος. Βέβαια, και οι τεχνολογίες στον πυρήνα τους χρησιμοποιούν JavaScript + jQuery, ενώ τη δομή την ελέγχει η βιβλιοθήκη materialize που αναφέραμε νωρίτερα, παρόλα αυτά όπως θα δείτε παρακάτω η ανάπτυξη τους δεν μοιάζει σε τίποτα.

3.1.2.1 Web Application

Η διαδικτυακή εφαρμογή αναπτύχθηκε κυρίως με τη χρήση της βιβλιοθήκης jQuery την materialize αλλά και με τη χρήση απλής JavaScript. Επίσης, για την ευκολότερη ανάπτυξη τον διαφορετικών views/pages χρησιμοποιήθηκε η template language EJS[37]. Η συγκεκριμένη template γλώσσα, δίνει τη δυνατότητα να εμπλουτίσεις της HTML σελίδες γράφοντας μέσα σε αυτές JavaScript σε οποιοδήποτε σημείο τους, και όχι μονό εντός του script element[38]. Χρησιμοποιώντας τα default delimiters <% %> ή δηλώνοντας custom, δίνεται η δυνατότητα να τρέξεις κομμάτια JavaScript που σε ενδιαφέρουν.

```
<div id="tags" class="render-word-space">
    <% if (rentags.length != 0) { %>
        <ul class="tags">
            <% for (var i = 0; i < rentags.length; i++) { %>
                <li class="tag"><a><%= rentags[i] %></a></li>
            <% } %>
        </ul>
    <% } else { %>
        <h4 class="tips blue-grey-text text-lighten-3">Your Tags will render here!</h4>
    <% } %>
</div>
```

Στα highlighted με κίτρινο χρώμα κομμάτια, φαίνεται η χρήση της EJS. Ουσιαστικά, στο συγκεκριμένο παράδειγμα, γίνεται ένας έλεγχος για το αν υπάρχει κάτι στο array rentags, και αν ναι, με τη χρήση μίας for loop εμφανίζει για κάθε ένα από τα στοιχεία, ένα li element με τη τιμή του εκάστοτε στοιχείου της loop. Σε περίπτωση που δεν υπάρχει κάτι στο array, απλά εμφανίζει ένα h4 element ενημερώνοντας ανάλογα τον χρήστη.

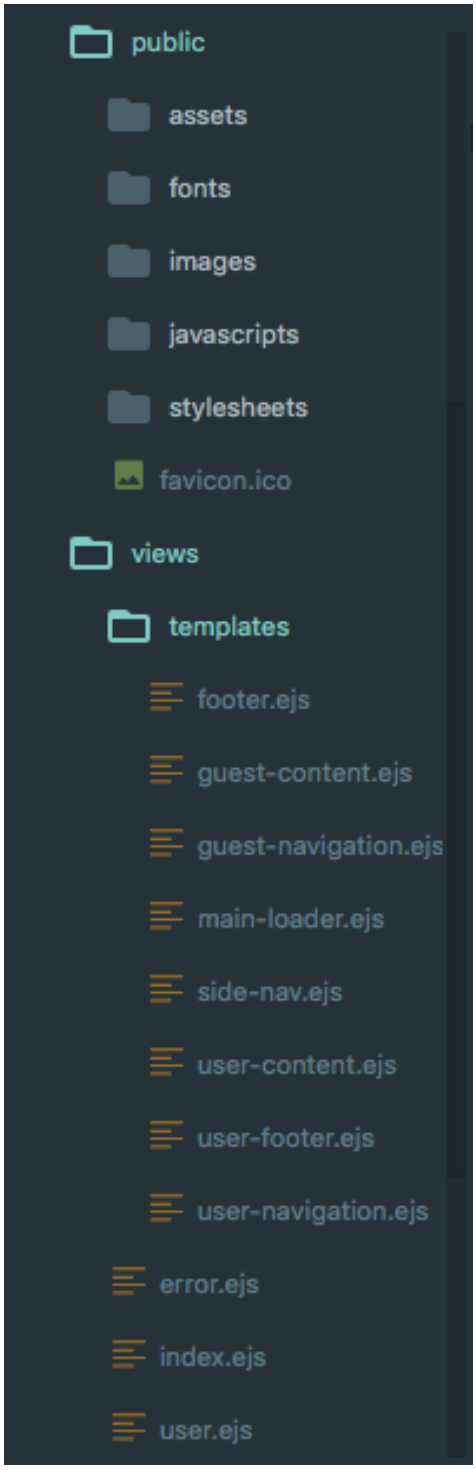
Η μόνη σύμβαση της συγκεκριμένης γλώσσα ήταν η αλλαγή της κατάληξης στα ονόματα των αρχείων που τη χρησιμοποιούσαν, από .html σε .ejs. Η χρήση της συγκεκριμένης γλώσσα, μου πρόσφερε αρκετή ευελιξία, καθώς μου έδωσε τη δυνατότητα να διαμορφώσω το περιεχόμενο κάθε view στη μορφή που ανάλογα χρειαζόταν βάσει τον εκάστοτε διαθέσιμων στοιχείων (logged in χρήστης, ή όχι) ή δεδομένων (γεμάτα ή όχι array κλπ.). Επίσης, με την χρήση της μπορούσα ακόμα να εισάγω ολόκληρα αρχεία μέσα σε άλλα.

```
<html lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
scale=1.0"/>
  <meta name="theme-color" content="#4285F4">

  <!-- CSS  -->
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  <link href="stylesheets/materialize.css" type="text/css" rel="stylesheet"
media="screen,projection"/>
  <link href="stylesheets/style.css" type="text/css" rel="stylesheet"
media="screen,projection"/>
</head>
<body>

<% include templates/main-loader.ejs %>

<% include templates/guest-navigation.ejs %>

<% include templates/guest-content.ejs %>

<% include templates/footer.ejs %>

<!--  Scripts-->
<script src="javascripts/materialize.js"></script>
</body>
</html>
```

Ξανά στα highlighted με κίτρινο κομμάτια, φαίνεται η χρήση της EJS για να γίνουν include άλλα αρχεία. Η χρήση αυτή, επίσης προσφέρει ευελιξία, καθώς δίνει τη δυνατότητα

εισαγωγής κομματιών κώδικα μόνο υπό προϋποθέσεις αυξάνοντας την απόδοση και ασφάλεια της εφαρμογής.

Πέραν των **.ejs** αρχείων, που στην πραγματικότητα είναι και τα **.html** μας αρχεία ταυτόχρονα, και όλα αυτά μαζί αποτελούν τις σελίδες της εφαρμογής ή αλλιώς γνωστά ως views, υπάρχουν ακόμα δύο μεγάλες κατηγορίες. Τα JavaScript αρχεία **.js**, που περιέχουν τη λογική κυρίως της διαδικτυακής εφαρμογής μας και ίσως ένα μικρό στιλιστικό μέρος, και τα Stylesheets αρχεία **.css**. Παρακάτω ακολουθεί η δομή των φακέλων της διαδικτυακής εφαρμογής, ενώ ακολουθεί εκτενέστερη ανάλυση τους.



Εικόνα 14: File structure - Web Application

Παρατηρούμε τον διαχωρισμό δύο μεγάλων κατηγοριών, views και public. Τον φάκελο των views με τα .ejs αρχεία τον αναλύσαμε νωρίτερα, ενώ ο φάκελος public παρατηρούμε ότι αποτελείται από 5 υπό-φακέλους και ένα αρχείο. Αναλυτικά:

Πίνακας 6: Ανάλυση public φακέλων

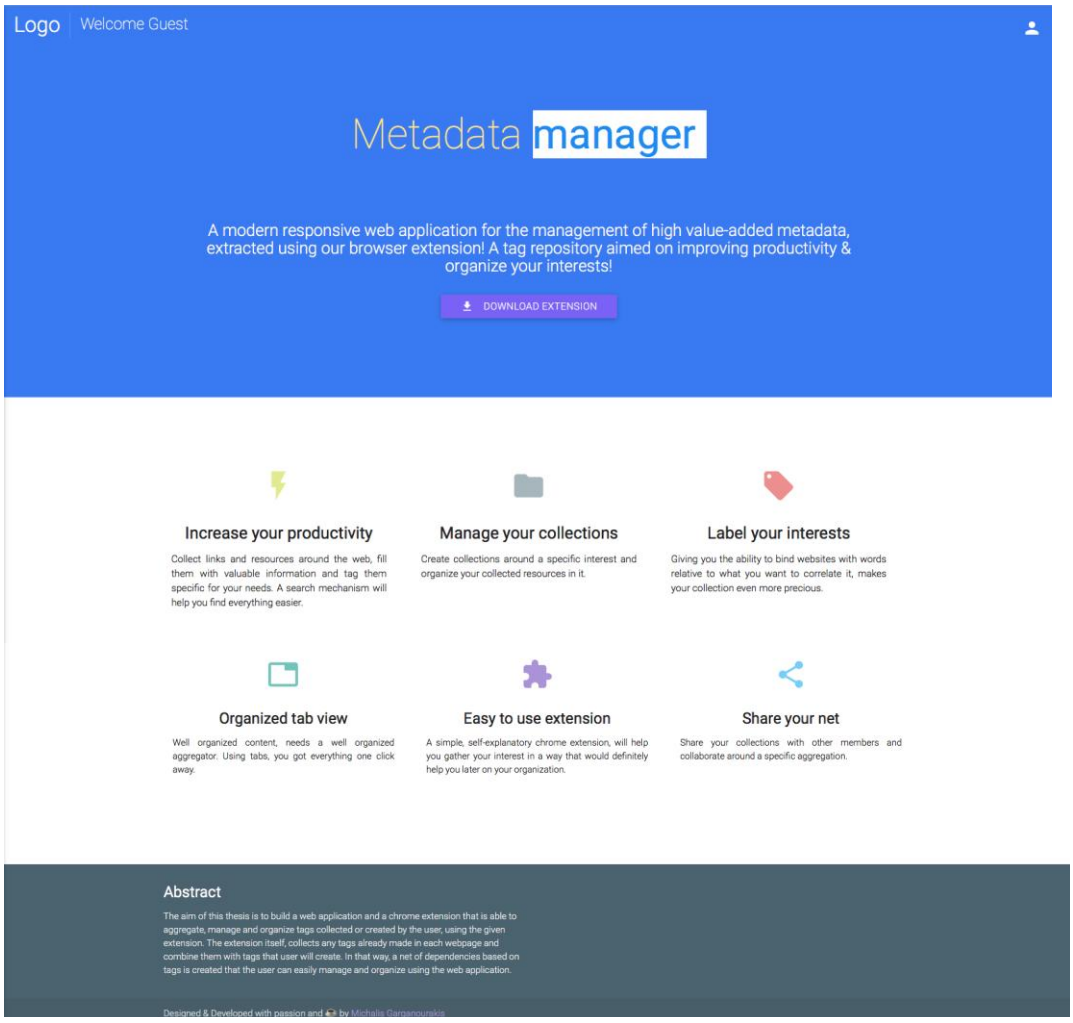
assets	*Όλα τα αρχεία που δεν ανήκουν σε καμία από τις υπόλοιπες κατηγορίες, όμως είναι αρχεία που θέλουμε να είναι διαθέσιμα στους τελικούς χρήστες.*
fonts	*Όλα τα αρχεία fonts που χρησιμοποιούνται από την εφαρμογή*
images	*Τα image holders για default περιπτώσεις, π.χ. άδεια κάρτα ή σελίδα 404*
javascripts	*Όλα τα αρχεία JavaScript απαραίτητα για να τρέξει σωστά η εφαρμογή*
stylesheets	*Όλα τα αρχεία .css για τη σωστή εμφάνιση των σελίδων*
favicon.ico	*Το favicon εικονίδιο της εφαρμογής*

4 Κύριο μέρος – Σενάρια Χρήσης

Για να μπορέσουμε να καταλάβουμε καλύτερα τις δυνατότητες της συγκεκριμένης σουίτας, θα προχωρήσουμε σε ένα (1) σενάριο χρήσης περνώντας από όλες τις διαθέσιμες επιλογές και δυνατότητες που προσφέρει η εκάστοτε υλοποίηση, καλύπτοντας πλήρως την λειτουργικότητα τους.

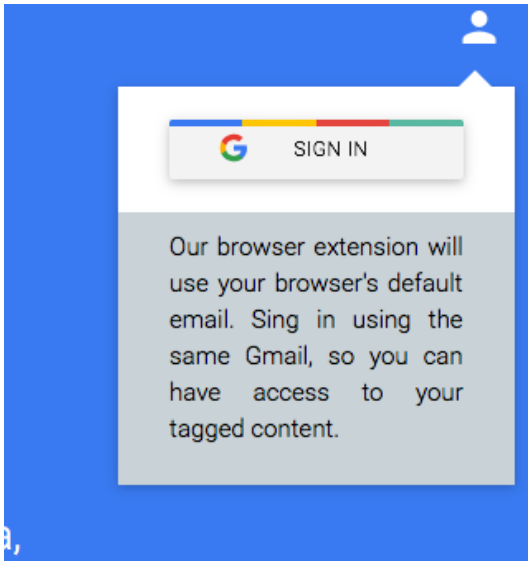
4.1 Σενάριο ατομικής/προσωπικής χρήσης

Ανοίγοντας την κεντρική σελίδα της εφαρμογής—και εφόσον δεν είμαστε εξουσιοδοτημένοι χρήστες ακόμη—above the fold[39], υπάρχει μία πολύ σύντομη animated περιγραφή σχετική με την εφαρμογή. Στην δεξιά πλευρά η φιγούρα/avatar υποδηλώνει το logged out χρήστη, ενώ στη αριστερή υπάρχει το λογότυπο, μαζί με έναν χαιρετισμό προς τον guest χρήστη. Αξίζει να σημειωθεί ότι ο χώρος αυτός έχει σκοπό να τραβήξει την προσοχή του χρήστη, καθώς κατά την περιήγηση του στην εφαρμογή θα του μεταφέρει συνεχώς χρήσιμη πληροφορία για τις διαδικασίες. Ακριβώς κάτω από το intro, παρέχεται η δυνατότητα απόκτησης της επέκτασης ενώ στο κέντρο της σελίδας, με έξι (6) μικρά grids κειμένου και εικονιδίων δίνεται στο πιθανό χρήστη μία συνοπτική επίδειξη των δυνατοτήτων της εφαρμογής. Τέλος, στο footer της σελίδας υπάρχει το abstract κείμενος της πτυχιακής.



Εικόνα 15: Αρχική σελίδα μη πιστοποιημένου χρήστη (guest)

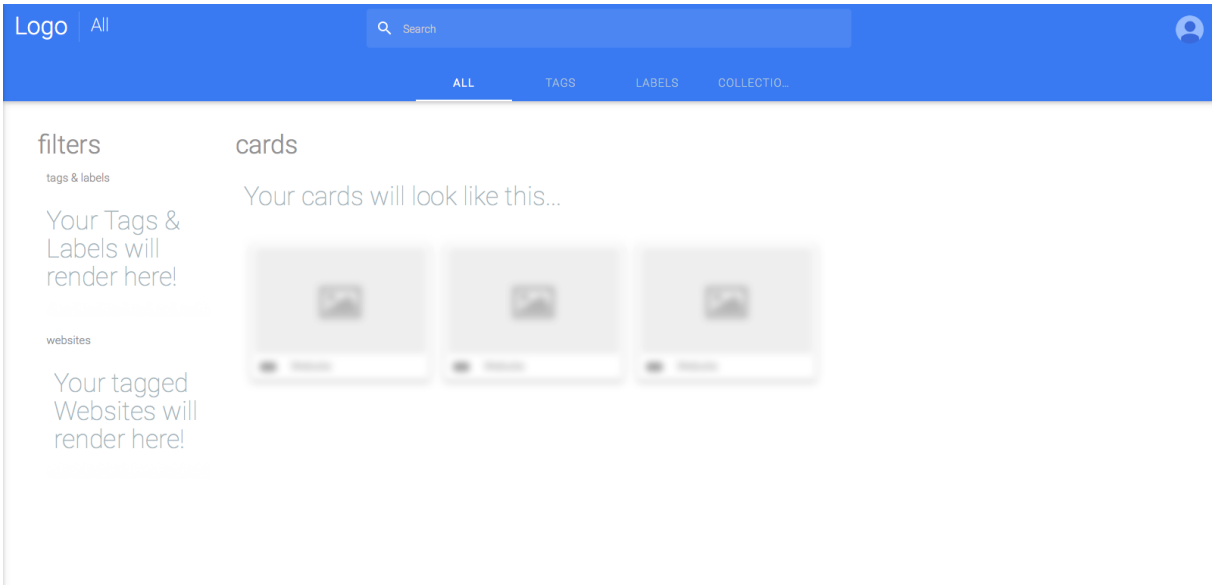
Το αμέσως επόμενο βήμα ενός ενδιαφερόμενου χρήστη, αφού κατεβάσει το browser extension, είναι η πρώτη είσοδος του στην εφαρμογή. Πατώντας στο avatar που αναφέραμε νωρίτερα, εμφανίζεται το παρακάτω popup δίνοντας του τη δυνατότητα σύνδεσης με την μόνη υποστηριζόμενη πλατφόρμα έως τώρα, αυτήν της Google.



Εικόνα 16: Pop-up εισόδου

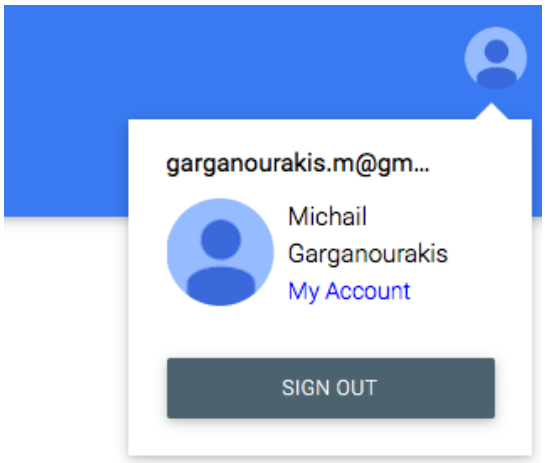
Εφόσον ο χρήστης εισέλθει στην πλατφόρμα για πρώτη φορά, θα ανακατευθυνθεί στη κεντρική σελίδα όπου μελλοντικά θα εμφανίζονται όλοι οι πόροι οι οποίοι έχει συλλέξει. Καθώς ο χρήστης δεν έχει συλλέξει ακόμη κάτι, σχετικά μηνύματα και ενδεικτικές κάρτες εμφανίζονται για να γεμίσουν τους χώρους και να δώσουν μία πρώτη αίσθηση στο χρήστη για το πως πρόκειται να διαμορφωθούν οι πόροι που θα συλλέξει. Η εφαρμογή χωρίζεται σε τέσσερις (4) βασικές καρτέλες(tabs) οι οποίες είναι αρκετές για να διαχωρίσουν όσα έχει συλλέξει ο χρήστης με βάση τις τέσσερις μεγάλες οντότητες της εφαρμογής (All, Tags, Labels και Collections).

Στην καρτέλα All αναφέραμε ότι θα εμφανίζονται όλοι οι διαδικτυακοί πόροι που έχει συλλέξει ο χρήστης ανεξάρτητα τον τρόπο. Στην περιοχή Tags θα γίνονται ορατά μόνο τα assets που έχουν συλλεχθεί και έχουν tags ενώ αντίστοιχα στην καρτέλα Labels όλα έχουν labels. Τέλος, στη περιοχή Collections ο χρήστης θα έχει τη δυνατότητα να δει όλες τις συλλογές που έχει δημιουργήσει, αλλά και όσες έχουν μοιραστεί τρίτοι μαζί του.



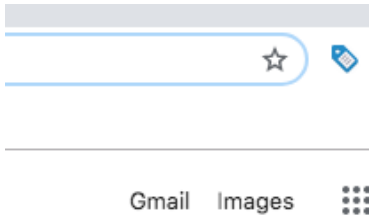
Εικόνα 17: Αρχική σέλιδα χρήστη (κενή)

Παρατηρούμε επίσης ότι πλέον το avatar του χρήστη έχει αλλάξει και έχει προσαρμοστεί με αυτό του google λογαριασμού του, και πλέον του δίνεται η δυνατότητα αποσύνδεσης. Επίσης, στα αριστερά του κεντρικού χώρου της εφαρμογής, φαίνεται ο χώρος που αργότερα θα γίνουν render τα φίλτρα, ενώ κεντρικά ο χώρος που θα εμφανιστούν οι κάρτες, κάθε μία εξ ’αυτών εκπροσωπώντας ένα ψηφιακό πόρο. Τέλος, κεντρικά στο header, παρέχεται μπάρα αναζήτησης, που μελλοντικά θα τον βοηθάει για την ευκολότερη προσπέλαση των στοιχείων που έχει συλλέξει.



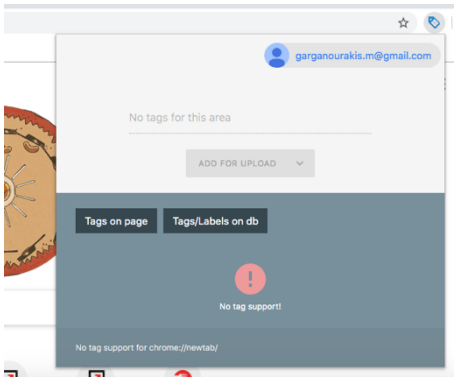
Εικόνα 18: Pop-up εξόδου

Αφού πήραμε μία πρώτη ιδέα από το πως τα πράγματα φαίνονται στην διαδικτυακή εφαρμογή, μπορούμε πλέον να δούμε και το browser extension. Η εγκατάσταση είναι ένα κλικ, ενώ αμέσως μετά εμφανίζεται δίπλα στην μπάρα αναζήτησης του περιηγητή σας.



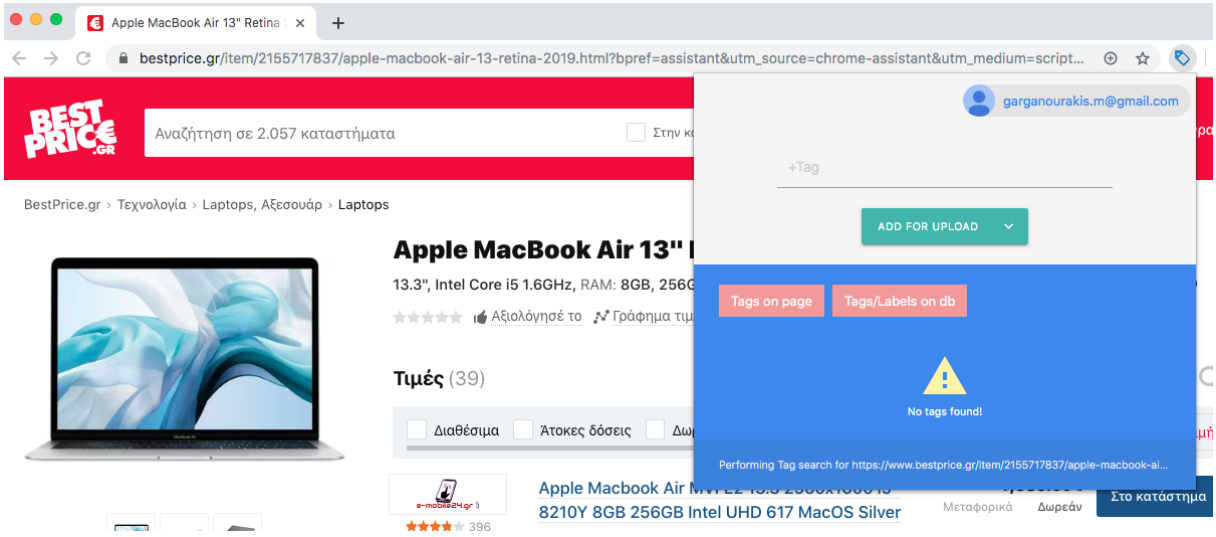
Εικόνα 19: Εικονίδιο extension στον browser

Το extension γνωρίζει που μπορεί να «λάβει δράση» και σε περιοχές όπως την αρχική σελίδα του chrome για παράδειγμα, βγάζει σχετικές ενημερώσεις προς τον χρήστη.
Εφόσον ο χρήστης βρει κάτι που τον ενδιαφέρει να συλλέξει (αρχικά θα δούμε το σενάριο προσωπικής χρήσης), και βρίσκεται στη σελίδα αυτού, αρκεί να πατήσει το κουμπί του extension για να ξεκινήσει τη διαδικασία. Αφού πατήσει το κουμπί, στην κάτω πλευρά του χώρου που θα ανοίξει, παίρνει το ανάλογο μήνυμα από το σύστημα ότι γίνεται σχετικό έλεγχος στη βάση της εφαρμογής αλλά και στη σελίδα, για πιθανές προτάσεις σε tags ή



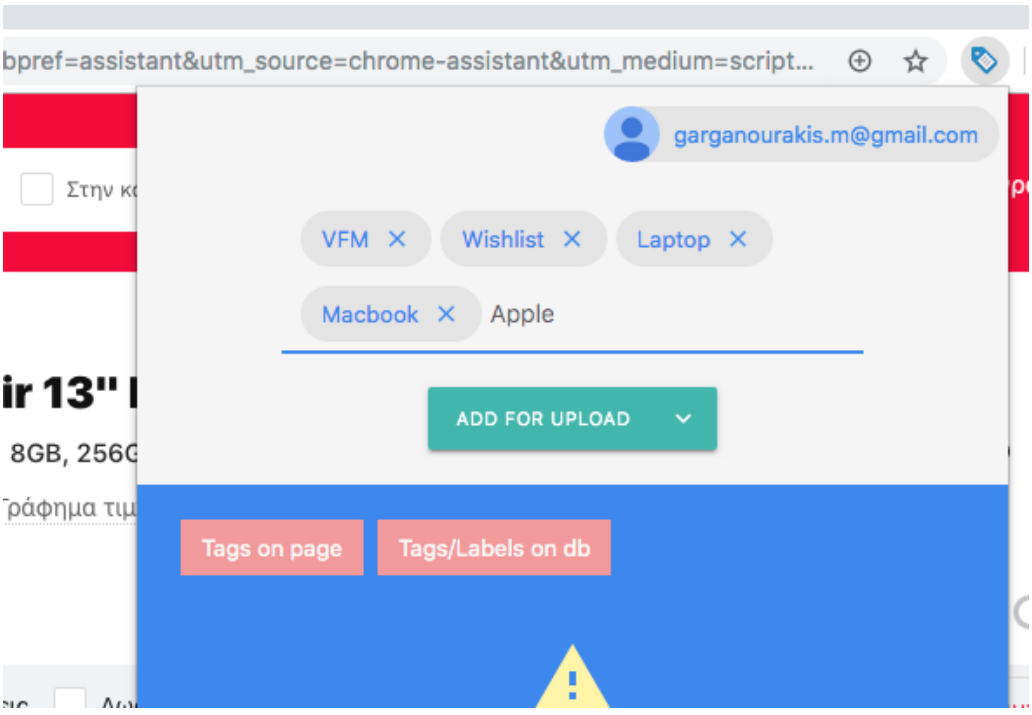
Εικόνα 20: Επίδειξη χώρου μη δυνατής χρήσης του extension

labels. Στη συγκεκριμένη περίπτωση δεν βρίσκει κάτι, επομένως και επιστρέφει το σχετικό μήνυμα.



Εικόνα 21: Επίδειξη χώρου πιθανής χρήσης extension

Ο χρήστης έχει τη δυνατότητα να «δέσει» την σελίδα αυτή με όσα labels επιθυμεί προσθέτοντας τα στο χώρο που φαίνεται **+Tag.**

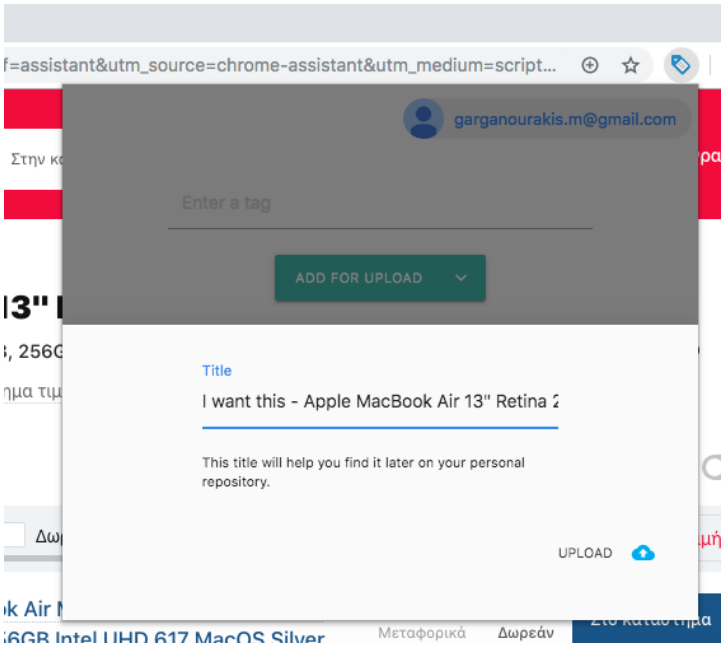


Εικόνα 22: Επίδειξη εισαγωγής tags μέσω του extension

Αφού τελειώσει με την προσθήκη τους, πατάει το **Add for upload** το οποίο προετοιμάζει το έδαφος αποστολής των δεδομένων στο server.

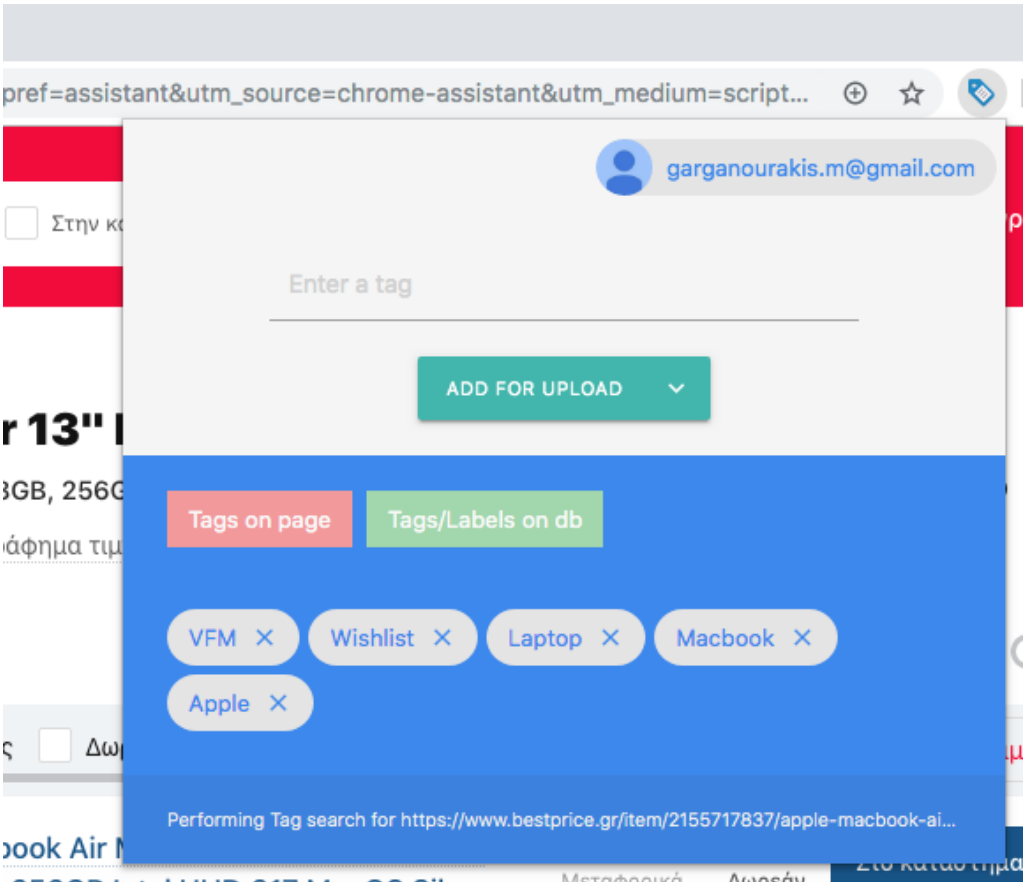
Εικόνα 23: Staging στάδιο αποστολής tags προς server

Τέλος, αν ο χρήστης πατήσει το check στην πάνω πλευρά, αφού δώσει ένα τίτλο για τη συγκεκριμένη διαλογή του (προτείνεται πάντα από το σύστημα σαν τίτλος του πόρου, για διευκόλυνση, ο τίτλος της σελίδας, όμως ο τίτλος αυτός είναι απόλυτα διαχειρίσιμος και στο χέρι του χρήστη) έπειτα αποστέλλεται στη βάση.



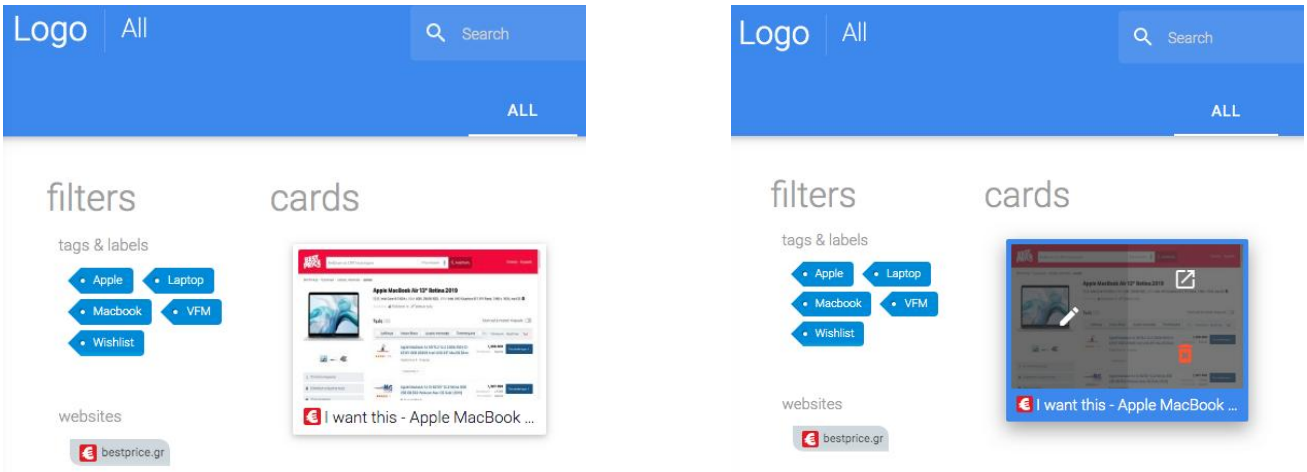
Εικόνα 24: Αποθήκευση δεδομένων ψηφιακού πόρου στη βάση

Αφού ο χρήστης πατήσει το κουμπί του **Upload**, πλέον ο σύνδεσμος/πόρος έχει αποθηκευτεί στον προσωπικό του χώρο. Επίσης, αν στο μέλλον βρεθεί ξανά στη σελίδα αυτή, το extension θα του δώσει τη παρακάτω εικόνα, ενώ θα του δίνει τη δυνατότητα να προσθέσει/αφαιρέσει tags, αλλά ακόμα και να αλλάξει τον τίτλο που έχει κρατήσει για το συγκεκριμένο ψηφιακό πόρο.



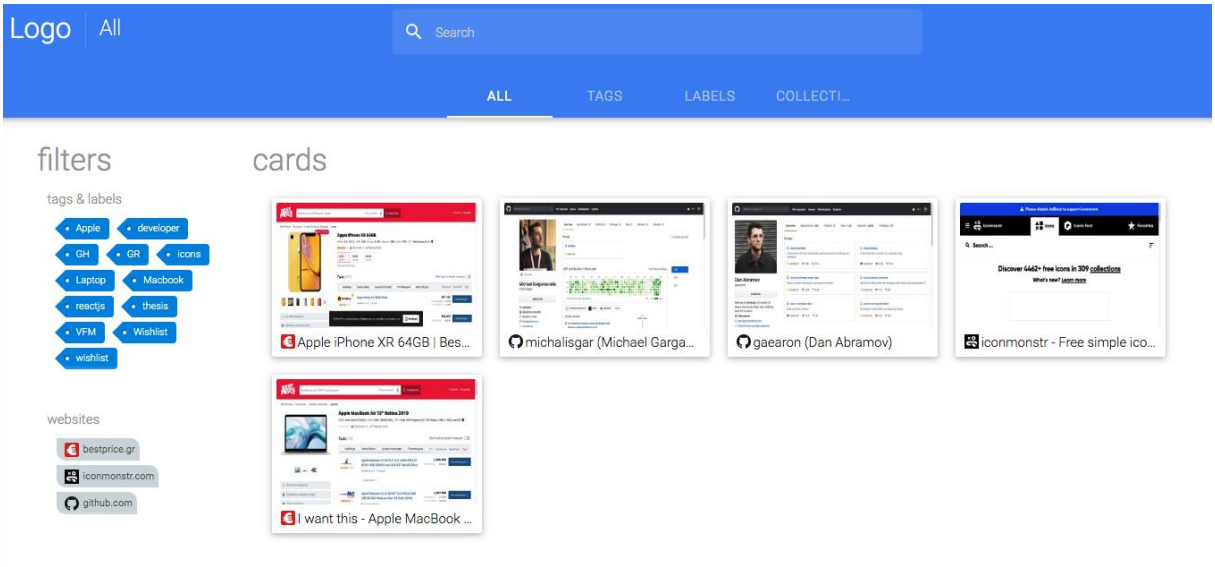
Εικόνα 25: Ενδεικτική επιστροφή χρήστη σε ήδη συλλεγμένο πόρο

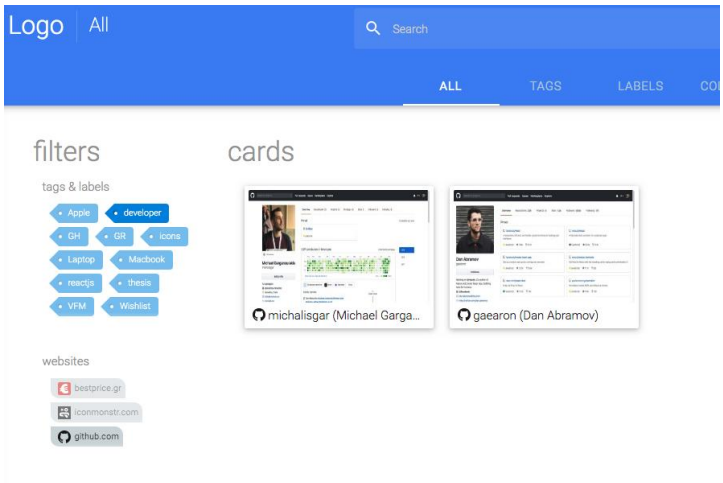
Επιστρέφοντας στην εφαρμογή, ο χρήστης πλέον μπορεί να δει και να επεξεργαστεί τη συγκεκριμένη σελίδα που μόλις σύλλεξε.



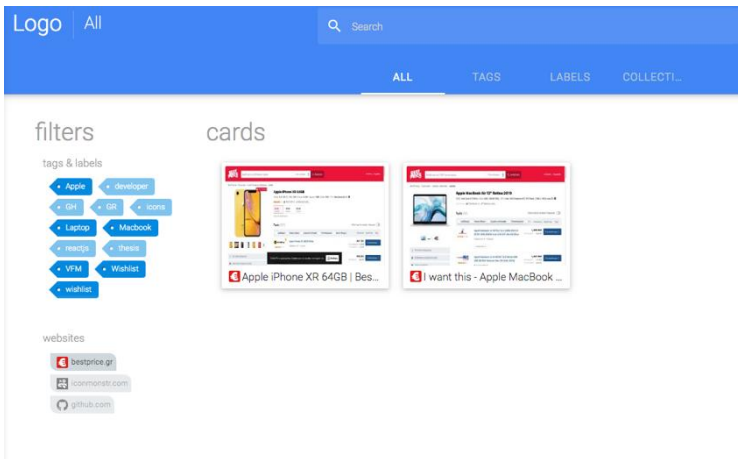
Εικόνα 26: Αριστερά η κανονική μορφή, δεξιά μετά από κλικ πάνω στον πόρο από τον χρήστη

Παρακάτω, ενδεικτικά η μορφή της εφαρμογής αφού έχουν μαζευτεί αρκετοί πόροι από τον χρήστη και πως αυτοί διαμορφώνονται με βάση της επιλογές από τα διαθέσιμα φίλτρα.





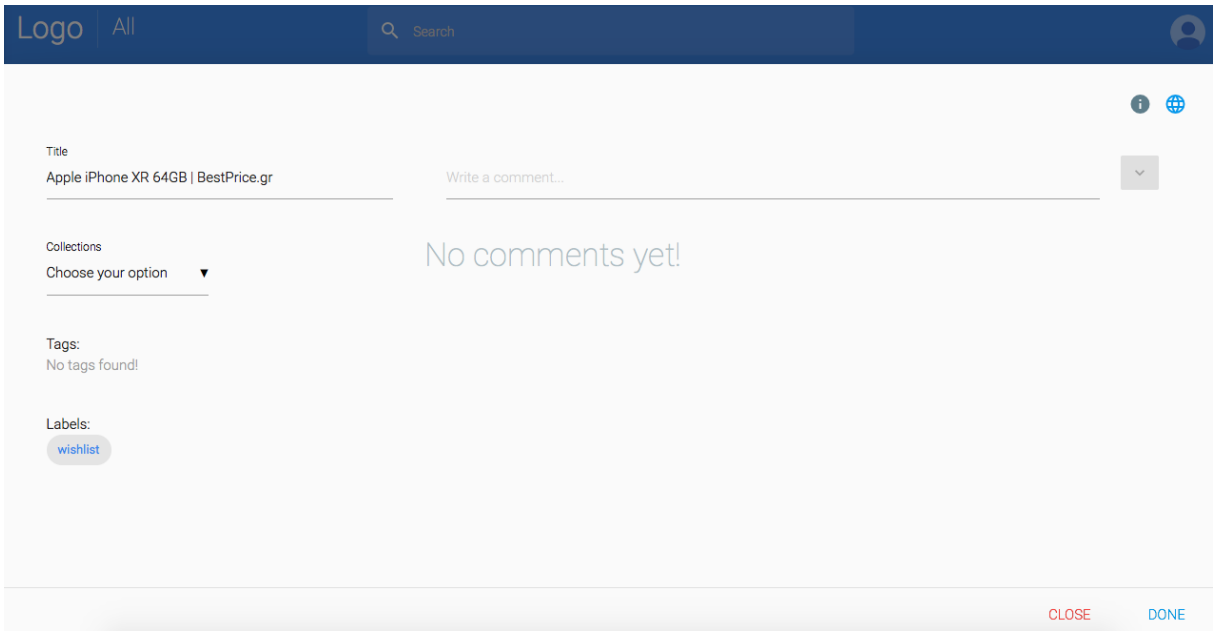
Εικόνα 28: Πόροι φιλτραρισμένοι επιλέγοντας το tag "developer"



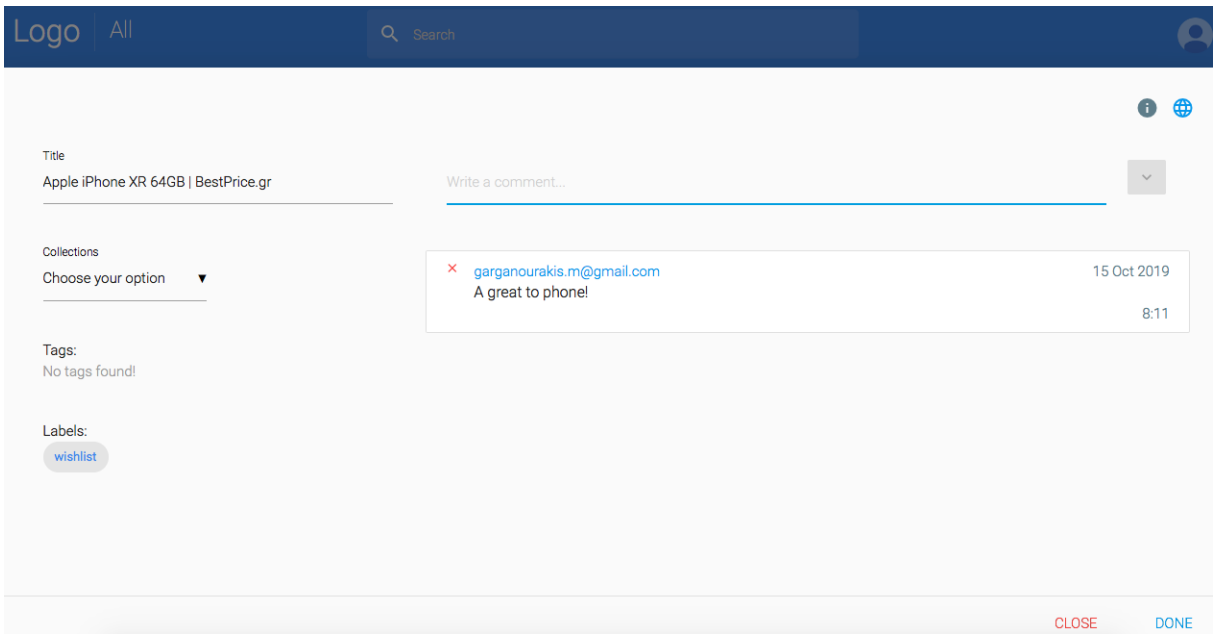
Εικόνα 27: Πόροι φιλτραρισμένοι με την επιλογή bestprice.gr στο φίλτρο websites

Εικόνα 29: Κανονική μορφή εφαρμογής με παραπάνω από ένα πόρους

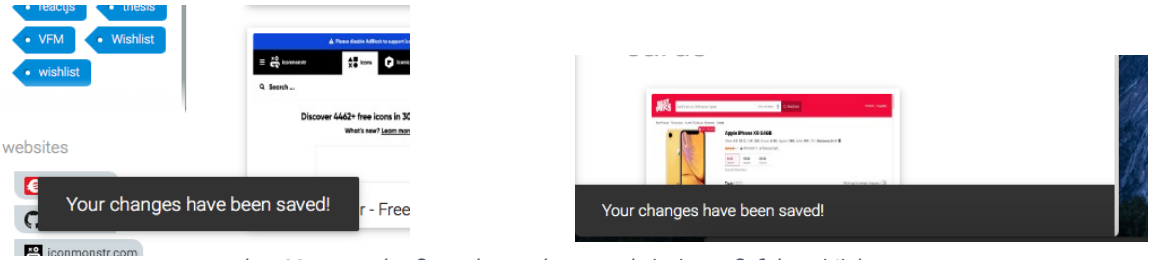
Στη συνέχεια θα δούμε πως είναι δυνατή η επεξεργασία ενός συλλεγμένου πόρου και πως αυτή φαίνεται εντός της εφαρμογής μας καθώς και τη δυνατότητα ενός χρήστη να αφήσει comments σε συλλεγμένο υλικό. Τέλος, κάθε φορά που ο χρήστης προχωράει σε σχετική αλλαγή, ενημερώνεται από το σύστημα με ειδοποίηση σε μορφή toast notification αν ήταν επιτυχείς ή όχι η «συναλλαγή» του με το σύστημα. Να σημειωθεί ότι το toast notification, ακολουθεί επίσης τις αρχές του material design και προσαρμόζεται βάσει των κανόνων στην κατάλληλη μορφή στις συσκευές mobile.



Εικόνα 30: Επίδειξη διαχείρισης και αλλαγών σε συλλεγμένο πόρο



Εικόνα 31: Επίδειξη δημιουργίας σχολίων σε συλλεγμένο πόρο



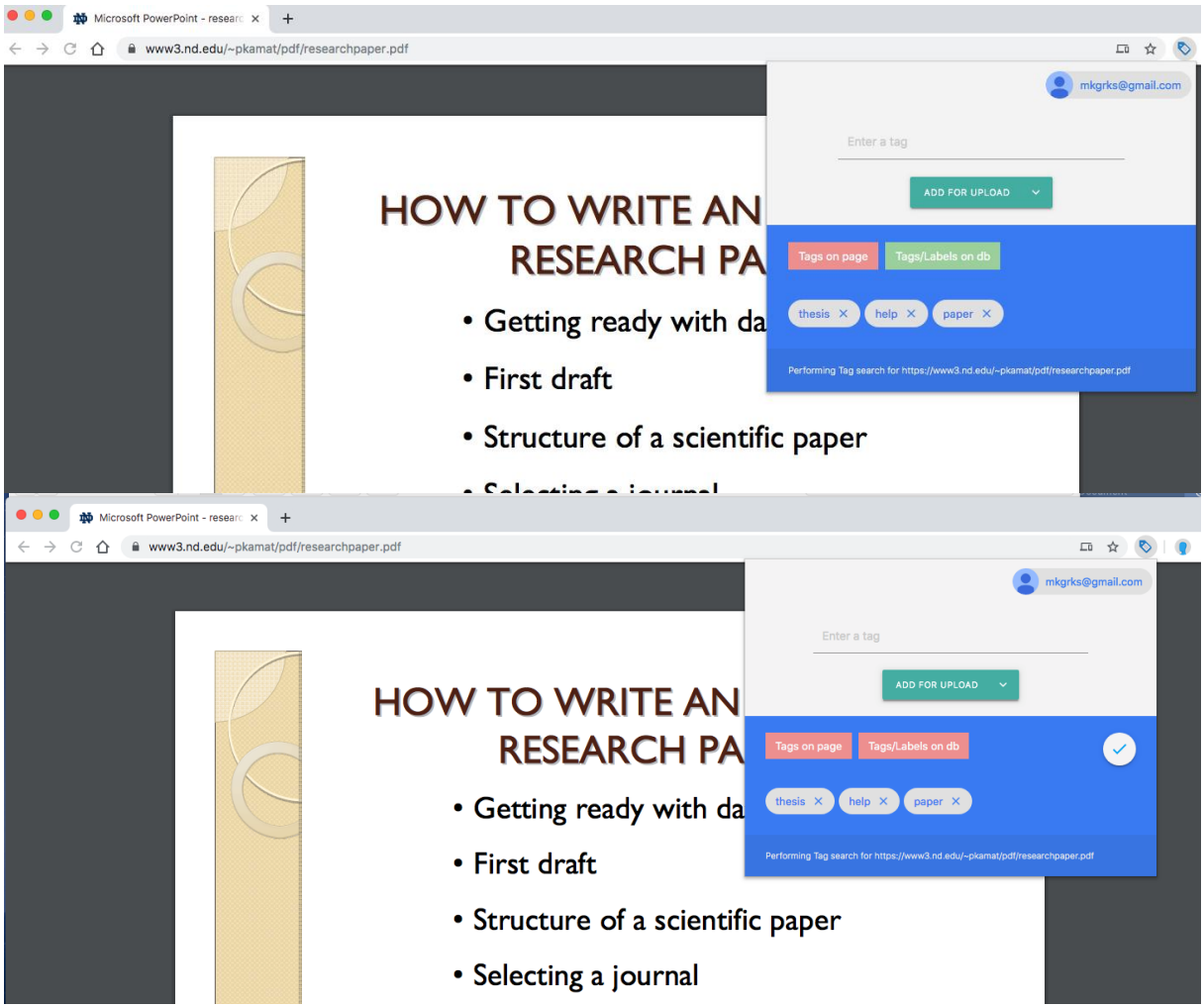
Εικόνα 32: Μορφή ειδοποιήσεων (αριστερά desktop, δεξιά mobile)

Με όλα αυτά έχουμε ολοκληρώσει ένα συνολικό σενάριο χρήσης της σουίτας σε προσωπικό επίπεδο, ενώ μένει να δούμε τις δυνατότητες που προσφέρει το συγκεκριμένο οικοσύστημα στην περίπτωση που χρησιμοποιηθεί σαν εργαλείο σε μία ομαδική εργασία.

4.2 Σενάριο χρήσης σε ομάδα

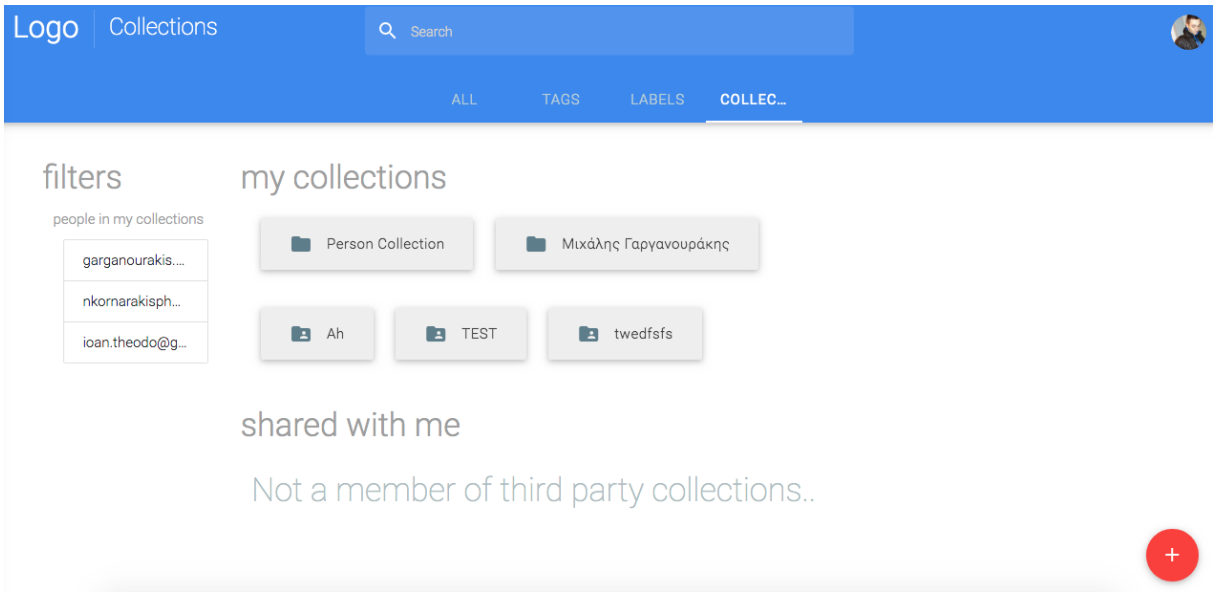
Σε αυτή την περίπτωση θα ξεκινήσουμε από το προφίλ δεύτερου χρήστη, ο οποίος και θα βρει κάποιο χρήσιμο υλικό σχετικό με την εργασία στο διαδίκτυο, το οποίο και θα συλλέξει και σε δεύτερο χρόνο θα διαμοιραστεί με όσους επιθυμεί.

Σύμφωνα με τις παρακάτω εικόνες βλέπουμε πως ο χρήστης βρήκε το υλικό που τον ενδιαφέρει, πρόσθεσε τις λέξεις κλειδιά που ταιριάζουν με το συγκεκριμένο υλικό σύμφωνα με το περιεχόμενο και προχώρησε στην αποθήκευση του.

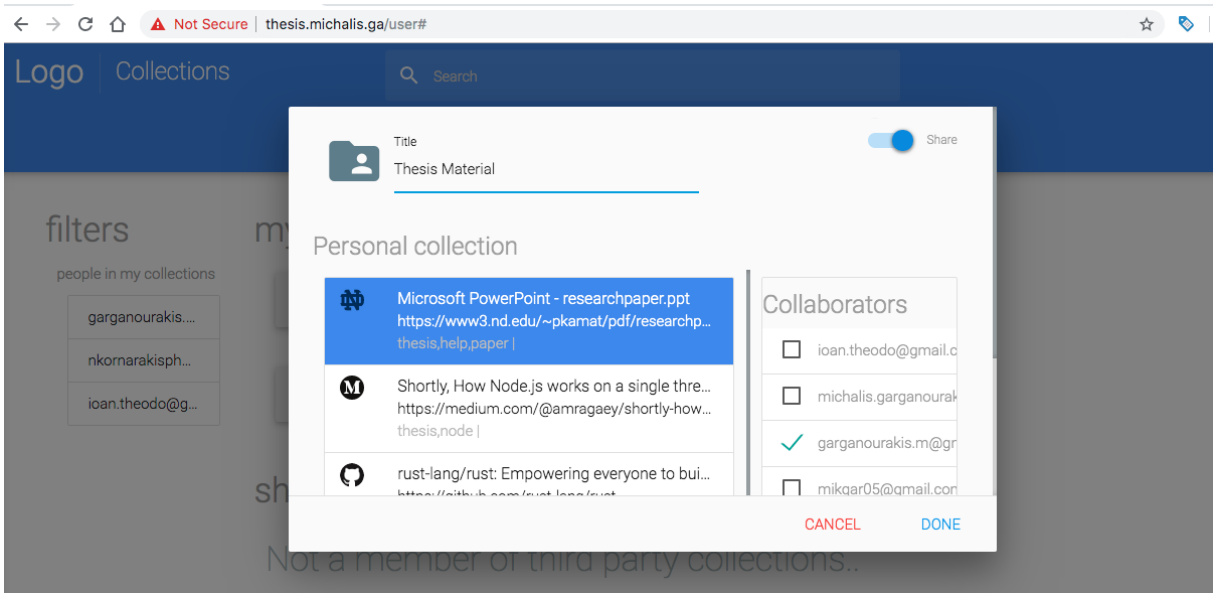


Εικόνα 33: Εύρεση υλικού για ομαδικό διαμοιρασμό

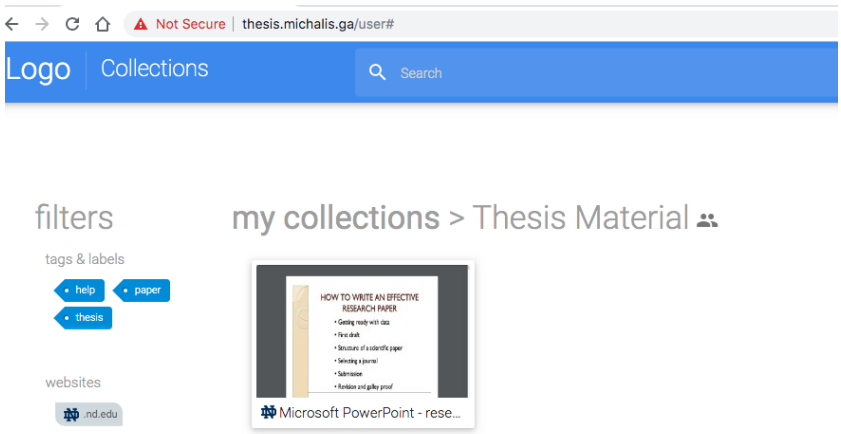
Επιστρέφοντας ξανά στην εφαρμογή και πλέον στην καρτέλα “Collections” θα δούμε πως φαίνονται οι συλλογές ενός χρήστη και πως δημιουργούμε ένα collection και προσθέτουμε μέλη σε αυτό. Στην συνέχεια, θα δούμε πως φαίνεται ο διαμοιρασμός αυτός στα μέλη που έχουμε προσθέσει και πως σε εμάς που τον έχουμε συλλέξει αλλά ταυτόχρονα πως θα γίνουν comments και από τους δύο στον πόρο αυτόν. Τέλος, θα δούμε την περίπτωση που τα μέλη θα δουν τον συλλεγμένο πόρο στο ίντερνετ και πως αυτός θα φαίνεται ενώ ήδη έχει βρεθεί κάποιο άλλο μέλος νωρίτερα εκεί.



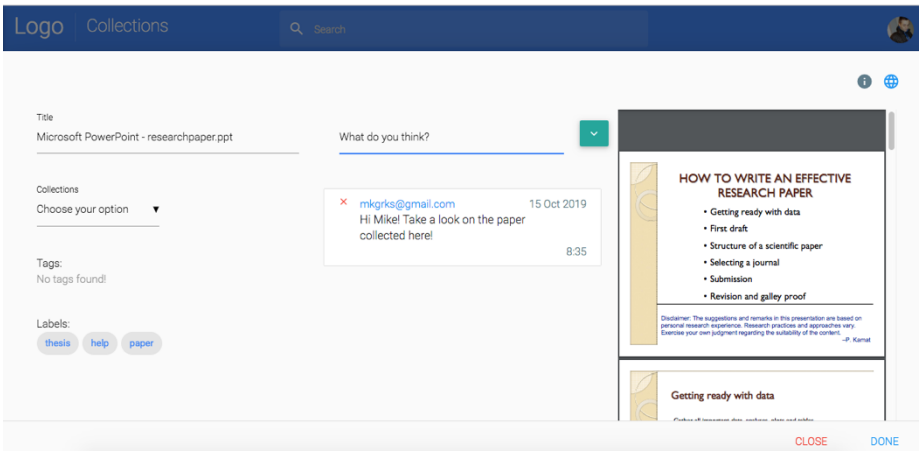
*Εικόνα 34: Προσωπικές συλλογές & διαμοιραζόμενες **σε** τρίτους*



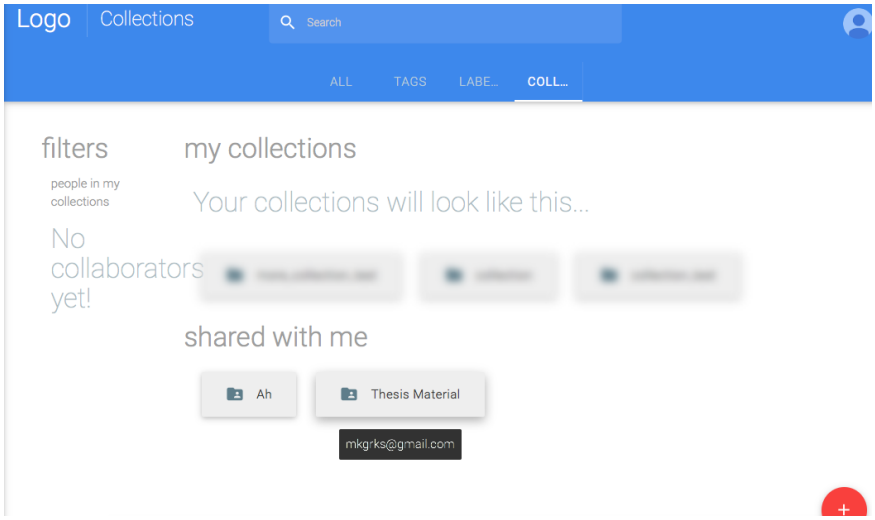
Εικόνα 35: Δημιουργία νέο collection, επιλογή υλικού για διαμοιρασμό, επιλογή ατόμων για διαμοιρασμό, ενεργοποίηση διαμοιρασμού



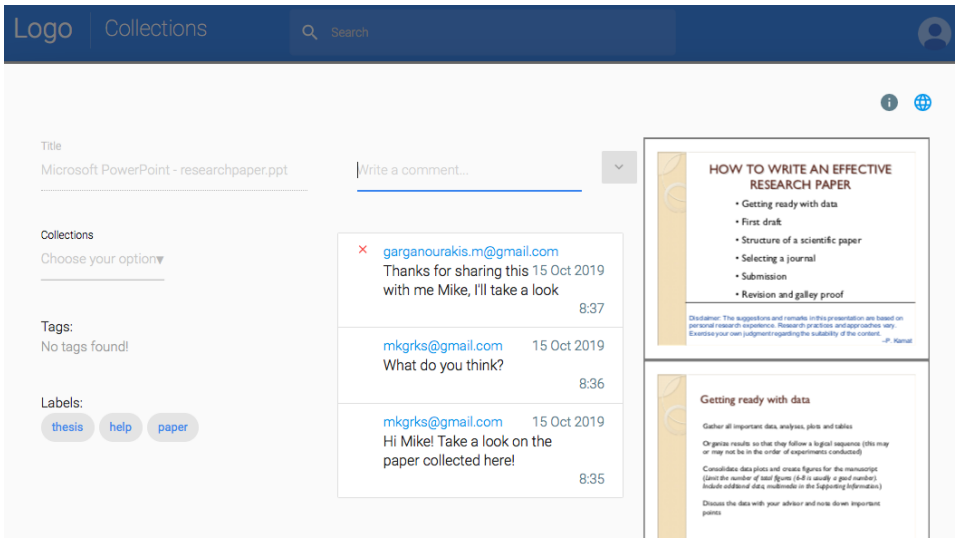
Εικόνα 36: Σελίδα διαμοιρασμού/collection view



Εικόνα 38: Edit διαμοιραζόμενου υλικού ***από χρήστη που το έχει συλλέξει***

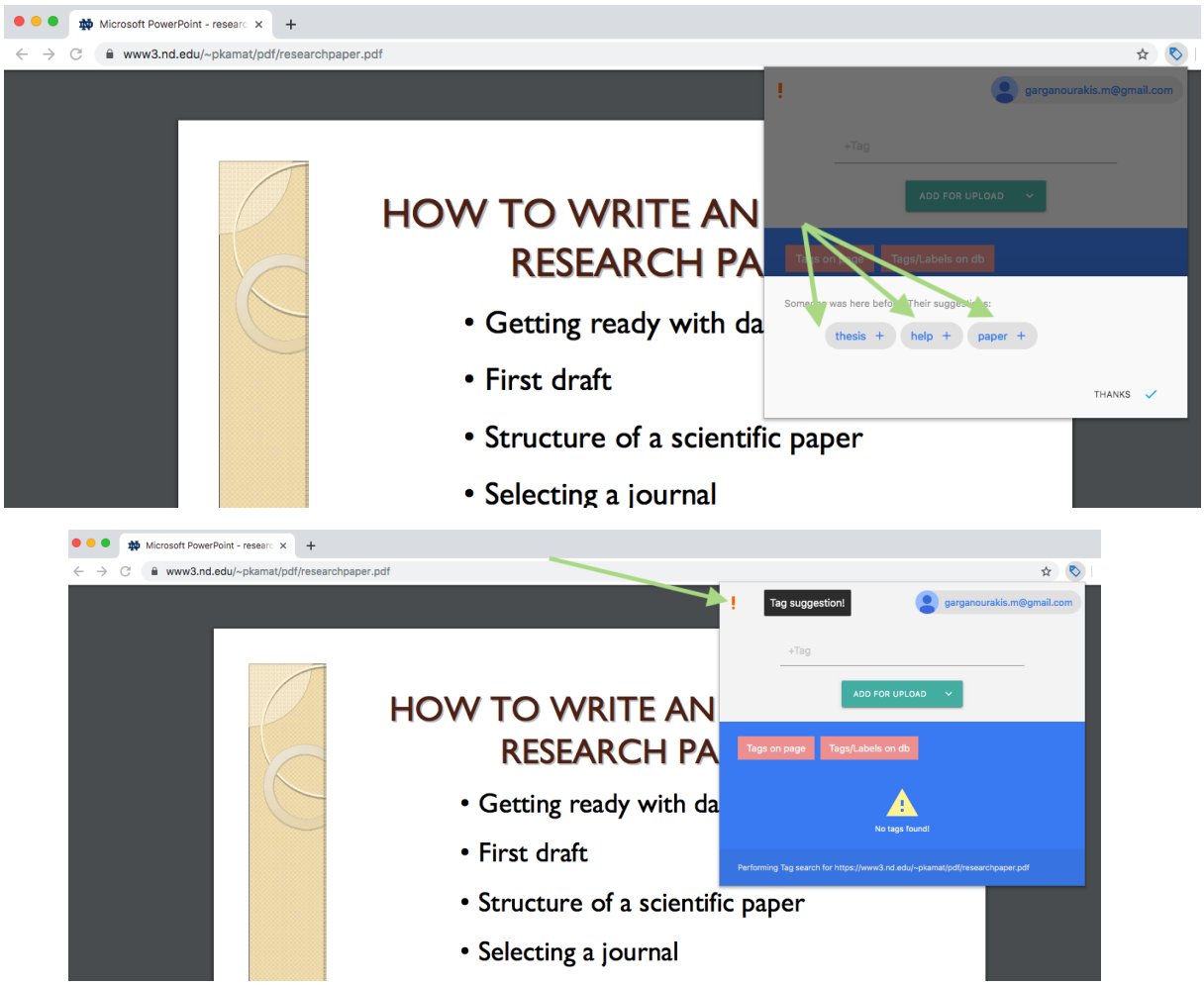


Εικόνα 37: Collection view, από τον πρώτο χρήστη που του διαμοιράστηκε ο πόρος (μήνυμα με το σχετικό email)



Εικόνα 39: Σχολιασμός χρήστη που του διαμοιράστηκε ο πόρος

Στις δύο παρακάτω εικόνες, φαίνεται πως το σύστημα ειδοποιεί τον χρήστη ότι κάποιος άλλος έχει περάσει από τον συγκεκριμένο ψηφιακό πόρο και ταυτόχρονα τις δυνατότητες που έχει (για παράδειγμα, με ένα κλικ να χρησιμοποιήσει και αυτός τα ίδια labels). Επομένως, το συγκεκριμένο feature προστατεύει την συλλογή διπλού υλικού, αλλά ταυτόχρονα βοηθάει και τους χρήστες στην κατάλληλη οργάνωση.



Εικόνα 40: Επίδειξη προτάσεων συστήματος σε ήδη συλλεγμένους πόρους

5 Αποτελέσματα

Η συγκεκριμένη σουίτα, θα μπορούσε να βοηθήσει τόσο άτομα που ψάχνουν τρόπο να οργανώσουν περιεχόμενο που τους ενδιαφέρει ή που συλλέγουν για συγκεκριμένο σκοπό ή δουλεία μέσω του διαδικτύου, αλλά ταυτόχρονα και στην οργάνωση μίας ομάδας με ένα συγκεκριμένο σκοπό και στόχο. Η μοντέρνα σχεδιασμένη και προσαρμοσμένη για όλες τις συσκευές διαδικτυακή εφαρμογή, μπορεί να δώσει λύση σε μία μεγάλη γκάμα ζητουμένων, ενώ παράλληλα, κρατάει καθαρή την εικόνα και το συλλεγμένο περιεχόμενο του χρήστη, χωρίς να γεμίζει με θόρυβο την εμπειρία που του προσφέρει. Στοχεύει στην αύξηση της παραγωγικότητας μίας ομάδας, ενώ με μηχανισμούς awareness ενημερώνει τους χρήστες της σε περιπτώσεις που θεωρεί απαραίτητο.

5.1 Συμπεράσματα

Έχοντας ολοκληρώσει την συγκεκριμένη εργασία με επιτυχία, προσθέτω στο προσωπικό μου portfolio, μία ολοκληρωμένη σουίτα οργάνωσης, δομημένη από την αρχή έως το τέλος με προσωπική έρευνα. Η έρευνα αυτή μου έδωσε την δυνατότητα να δουλέψω και να μάθω νέες τεχνολογίες, ενώ ταυτόχρονα να δοκιμάσω τεχνικές και υλοποιήσεις που χρησιμοποιούνται κατά κόρον από εταιρίες στο κλάδο. Επίσης η ευκαιρία να δουλέψω σε μία τέτοια λύση, με βοήθησε να καταλάβω πως λειτουργεί μία ομάδα, πως δρουν τα άτομα μέσα σε αυτή και πως μπορούμε εμείς σαν προγραμματιστές να δώσουμε λύσεις σε πολλά από αυτά.

5.2 Μελλοντική Επέκταση

Η συγκεκριμένη εργασία, έχει αναπτυχθεί με σκοπό το backend και ο κεντρικός μηχανισμός να μπορεί να διαμοιραστεί ώστε άλλες εφαρμογές τρίτων, να μπορούν να εκμεταλλευτούν πλήρως τις δυνατότητες και να τις επεκτείνουν. Το API έχει χτιστεί με γνώμονα τη συγκεκριμένη προσέγγιση και είναι δυνατή η οποιαδήποτε παραμετροποίηση του. Επίσης, η δυνατότητα χρήσης και άλλων τρίτων υπηρεσιών εξουσιοδότησης για παράδειγμα Facebook, μπορούν να χρησιμοποιηθούν χωρίς ιδιαίτερο κόπο, ενώ με μικρή προσπάθεια, μπορεί να χτιστεί και αυτόνομο σύστημα εξουσιοδότησης.

Βιβλιογραφία

[1] «IEEE» αριθμημένο σύστημα αναφοράς, στο σύνδεσμο: https://www.ieee.org/content/dam/ieee-org/ieee/web/org/conferences/style_references_manual.pdf
[2] «Charles Bachman», στο σύνδεσμο: https://en.wikipedia.org/wiki/Charles_Bachman
[3] «IDS (Integrated Data Store)», στο σύνδεσμο: https://en.wikipedia.org/wiki/Integrated_Data_Store
[4] «CODASYL (Conference/Committee on Data Systems Languages)», στο σύνδεσμο: https://en.wikipedia.org/wiki/CODASYL
[5] «hash», στο σύνδεσμο: https://en.wikipedia.org/wiki/Hash_function
[6] «Edgar F. Codd», στο σύνδεσμο: https://en.wikipedia.org/wiki/Edgar_F._Codd
[7] «IBM», στο σύνδεσμο: https://en.wikipedia.org/wiki/IBM
[8] «System R», στο σύνδεσμο: http://db.cs.berkeley.edu/cs262/SystemR-annotated.pdf
[9] «query language», στο σύνδεσμο: https://en.wikipedia.org/wiki/Query_language\
[10] «Larry Ellison», στο σύνδεσμο: https://en.wikipedia.org/wiki/Larry_Ellison
[11] «Michael Stonebraker», στο σύνδεσμο: https://en.wikipedia.org/wiki/Michael_Stonebraker
[12] «XML» γλώσσα σήμανσης με κανόνες, για την ηλεκτρονική κωδικοποίηση κειμένων, στο σύνδεσμο: https://el.wikipedia.org/wiki/XML
[13] «έγγραφο» αναφερόμενο σε εξυπηρετητή, στο σύνδεσμο: https://tools.ietf.org/html/rfc5
[14] «ARPANET» πρόγονος του διαδικτύου, στο σύνδεσμο: https://en.wikipedia.org/wiki/ARPANET
[15] «clients», στο σύνδεσμο: https://en.wikipedia.org/wiki/Client_(computing)
[16] «open-source», στο σύνδεσμο: https://en.wikipedia.org/wiki/Open-source_software
[17] «Apache HTTP Server Project», στο σύνδεσμο: https://httpd.apache.org/ABOUT_APACHE.html
[18] «caching», ή γνωστό και ως web cache στο σύνδεσμο: https://en.wikipedia.org/wiki/Web_cache
[19] «JavaScript», στο σύνδεσμο: https://developer.mozilla.org/en-US/docs/Web/JavaScript
[20] «V8 JavaScript engine» μηχανή βασισμένη στη JavaScript και WebAssemply ανεπτυγμένη από την Google, στο σύνδεσμο: https://v8.dev
[21] «npm», στο σύνδεσμο: https://www.npmjs.com
[22] «endopoints», στο σύνδεσμο: https://www.paloaltonetworks.com/cyberpedia/what-is-an-endpoint
[23] «JSON» JavaScript Object Notation, στο σύνδεσμο: http://www.json.org
[24] «HTML» Hypertext Markup Language, στο σύνδεσμο: https://html.spec.whatwg.org/#toc-iana
[25] «scripting», στο σύνδεσμο: https://en.wikipedia.org/wiki/Scripting_language
[26] «CSS» Cascading Style Sheets, στο σύνδεσμο: https://www.w3.org/Style/CSS/
[27] «Typography» η τέχνη και τεχνική της σχεδίασης γραμμάτων, στο σύνδεσμο: https://en.wikipedia.org/wiki/Typography
[28] «Carnegie Mellon University», στο σύνδεσμο: https://www.cmu.edu
[29] «pg», στο σύνδεσμο: https://node-postgres.com
[30] «morgan», στο σύνδεσμο: https://github.com/expressjs/morgan#readme
[31] «cookie-parser», στο σύνδεσμο: https://www.npmjs.com/package/cookie-parser
[32] «body-parser» , στο σύνδεσμο: https://www.npmjs.com/package/body-parser
[33] «express-session» , στο σύνδεσμο: https://github.com/expressjs/session#readme

[34] «path», στο σύνδεσμο: https://www.npmjs.com/package/path
[35] «passport», στο σύνδεσμο: http://www.passportjs.org
[36] «dbdiagram.io», στο σύνδεσμο: http://dbdiagram.io
[37] «EJS», στο σύνδεσμο https://ejs.co
[38] «script element», στο σύνδεσμο https://developer.mozilla.org/en/docs/Web/HTML/Element/script
[39] «above the fold», στο σύνδεσμο: Above the fold is also used in website design (along with "above the scroll") to refer to the portion of the webpage that is visible without scrolling