Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Μελέτη του Google Realtime API και των πιθανών εφαρμογών του σε συνεργατικά περιβάλλοντα

Ζερβουδάκης Στέφανος

Α.Μ 4085

Επιβλέπων καθηγητής : Δημοσθένης Ακουμιανάκης

Επιτροπή Αξιολόγησης :

Ημερομηνία Παρουσίασης :

Abstract

The goal of this dissertation is the familiarization of the student with a category of web applications, which nowadays constitute the coherent web in a wide scope of tools and software systems. These applications are called Application Programming Interfaces (APIs) and are offered from software and services makers to third parties, to support the development of specialized web applications. Therefore, an API is a library of executable services that constitute a kind of a programmer's interface with the functions performed by a library or a service during it's execution. The special feature of these routines, is that they can be coded by a developer, in order to create applications tha make use of the possibilities offered by the API.

For the needs of this dissertation, we focused on Google's Realtime API study and the analysis of the opportunities it offers for the development of cooperative applications.The approach adopted was to develop an application tha will use the Google Realtime API in order to provide new services to users of an existing platform. For this purpose the Trello platform was selected , a widespread online collaborative project implementation service. The detrimental purpose of the application developed is to get data from Trello and with the help of Realtime API to create new information opportunities (for the members involved in the application) as well as new communication possibilities among the partners of the application. Let us note here that our goal was primarily to support information and communication functions of partners that are not supported by the Trello platform itself. A secondary objective was to support basic operations already in place mainly for confirmation purposes (that is, there is communication between the two web applications Trello and the application developed).

Σύνοψη

Στόχος της πτυχιακής εργασίας είναι η εξοικείωση του φοιτητή με μια κατηγορία διαδικτυακών εφαρμογών που στις μέρες μας συνιστούν το συνεκτικό ιστό σε ένα ευρύ φάσμα εργαλείων και συστημάτων λογισμικού. Οι εφαρμογές αυτές ονομάζονται Διεπαφές Προγραμματισμού Εφαρμογών (Application Programming Interfaces, APIs) και προσφέρονται από κατασκευαστές λογισμικού και υπηρεσιών προς τρίτους προκειμένου να υποστηριχθεί η ανάπτυξη εξειδικευμένων διαδικτυακών εφαρμογών. Επομένως, ένα API είναι μια βιβλιοθήκη από εκτελέσιμες / υπηρεσίες που συνιστούν ένα είδος διεπαφής του προγραμματιστή με τις λειτουργίες που επιτελεί κατά την εκτέλεσή της ένα λογισμικό ή βιβλιοθήκη. Το ιδιαίτερο χαρακτηριστικό των ρουτινών αυτών είναι ότι μπορούν να ενσωματωθούν σε κώδικα από ένα προγραμματιστή προκειμένου να δημιουργηθούν εφαρμογές που να αξιοποιούν τις δυνατότητες που παρέχει το κάθε API.

Για της ανάγκες της παρούσας πτυχιακής εστιάσαμε στη μελέτη του Google Realtime API και την ανάλυση των δυνατοτήτων που προσφέρει για την ανάπτυξη συνεργατικών εφαρμογών. Η προσέγγιση που υιοθετήθηκε ήταν να αναπτυχθεί μια εφαρμογή που θα αξιοποιήσει το Google Realtime API προκειμένου να παρέχει νέες υπηρεσίες σε χρήστες που χρησιμοποιούν μια ήδη υπάρχουσα πλατφόρμα. Για το σκοπό αυτό επιλέχθηκε η πλατφόρμα Trello, μια ευρέως διαδεδομένη διαδικτυακή υπηρεσία συνεργατικής εκτέλεσης έργου. Απώτερος σκοπός της εφαρμογής που αναπτύχθηκε είναι να παίρνει πληροφορίες (data) από το Trello και με την βοήθεια του Realtime API να δημιουργεί νέες δυνατότητες ενημέρωσης (για τα μέλη που δραστηριοποιούνται στην εφαρμογή) όπως επίσης και νέες δυνατότητες επικοινωνίας μεταξύ των συνεργατών της εφαρμογής. Ας σημειωθεί εδώ ότι στόχος μας ήταν πρωτίστως να υποστηρίξουμε λειτουργίες ενημέρωσης και επικοινωνίας εταίρων οι οποίες δεν υποστηρίζονται από την ίδια την πλατφόρμα του Trello. Δευτερεύων στόχος ήταν η υποστήριξη βασικών λειτουργιών που διαθέτει ήδη το Trello κυρίως για λόγους επιβεβαίωσης (δηλ ότι υπάρχει επικοινωνία μεταξύ των 2 διαδικτυακών εφαρμογών , Trello – εφαρμογή που αναπτύχθηκε).

Πίνακας Περιεχομένων

Πίνακας Εικόνων

1 Εισαγωγή

Η εκπόνηση μιας πτυχιακής εργασίας έχει σαν πρωταρχικό στόχο την επίγνωση, κατανόηση και εξοικείωση του φοιτητή με θέματα της τρέχουσας τεχνολογικής στάθμισης, τόσο σε θεωρητικό όσο και σε εφαρμοσμένο επίπεδο. Επομένως, ολοκληρώνοντας την πτυχιακή εργασία ο φοιτητής αναμένει να έχει αποκομίσει δεξιότητες τέτοιες που να είναι σε θέση να δρομολογήσει την ανάλυση ενός σύνθετου προβλήματος, να εντοπίσει τις βασικές γνώσεις και εργαλεία που απαιτεί η επίλυση του, καθώς και να τεκμηριώσει με ακρίβεια την κάθε αποδεκτή μεθοδολογία που δημιούργησε για την επίλυση αυτού .

Όσον αφορά την παρούσα πτυχιακή εργασία, υπήρχαν κάποιοι βασικοί στόχοι που πραγματοποιήθηκαν με επιτυχία ολοκληρώνοντας την εφαρμογή και τεκμηριώνοντας όλες τις ενέργειες που χρειάστηκαν για να περατωθεί το σύστημα αυτό. Όσον αφορά τους στόχους , σαν κύριος παράγοντας ήταν οι αυξημένες απαιτήσεις για διαλειτουργικότητα των εφαρμογών-υπηρεσιών. Πιο συγκεκριμένα , εφαρμογές που χρησιμοποιούν κατά κόρον οι χρήστες σε αρκετές περιπτώσεις δεν ανταποκρίνονται στις προσδοκίες (είτε σε χρόνους απόκρισης ενεργειών , είτε σε ελλιπής λειτουργίες που διαθέτουν), γι’ αυτό και εμείς με την σειρά μας αναλύουμε την πολυπλοκότητα αυτού του προβλήματος. Στην συνέχεια , θα αναλυθεί η χρήση διεπαφών προγραμματισμού εφαρμογών σε γενικές γραμμές. Το γνωστικό πεδίο αυτό (API) επισημάνθηκε διότι με την χρήση αυτού μπορεί να επέλθει εξέλιξη στην τεχνολογία και γενικά στον τομέα της πληροφορικής. Εφαρμογές που αξιοποιούν API άλλων υπηρεσιών, μπορούν με αυτόν τον τρόπο να φέρουν βελτίωση στην διαλειτουργικότητα και λύσεις στην πολυπλοκότητα των λειτουργιών που διαθέτει η τελευταία. Δοκιμάζοντας λοιπόν νέες προγραμματιστικές τεχνικές που δεν υπήρχαν προηγουμένως επέρχεται η εξέλιξη που προαναφέραμε πριν. Όλα αυτά τεκμηριώθηκαν δημιουργώντας μια εξωτερική εφαρμογή που αξιοποιεί το Google Realtime API και με την αξιοποίηση μιας ήδη υπάρχουσας δημοφιλής πλατφόρμας (Trello) δημιουργήθηκαν νέες υπηρεσίες στους χρήστες που χρησιμοποιούν την εφαρμογή , όπως επίσης και λειτουργίες που υπήρχαν στην πλατφόρμα (για λόγους πληρότητας). Πελάτες που κάνουν χρήση της εφαρμογής που αναπτύχθηκε εξυπηρετούνται γρηγορότερα και με λιγότερη πολυπλοκότητα σε υπηρεσίες που επιθυμούν να αξιοποιήσουν.

1.1 Συνεργατικές εφαρμογές στο διαδίκτυο και η χρήση αυτών

Κάθε διαδικτυακή εφαρμογή είναι διαθέσιμη στους χρήστες της μέσω του διαδικτύου και για να την χρησιμοποιήσει κανείς χρειάζεται έναν περιηγητή. Το θετικό τέτοιων υπηρεσιών είναι ότι ο καθένας μπορεί να έχει άμεση πρόσβαση από οποιαδήποτε είδους συσκευή χωρίς να πρέπει να εγκατασταθεί κάποιο άλλο πρόγραμμα στο τερματικό τους πλην του browser. Συνήθως τέτοιες εφαρμογές εκτελούνται σε δυνατές υπολογιστικές συσκευές οι οποίες έχουν τον ρόλο του σταθμού εξυπηρέτησης (server) ώστε να παρέχουν υπηρεσίες σε παραπάνω του ενός χρήστη. Κάποια από τα συστήματα διαδικτυακών εφαρμογών έχουν να κάνουν με κρατήσεις (π.χ. σε ξενοδοχεία , ταβέρνες κ.τ.λ.), με διανομή τροφίμων, διαχείριση περιεχομένου , και διάφορα άλλα.

Όσον αφορά εφαρμογές διαχείρισης περιεχομένου , σε αρκετές περιπτώσεις κρίνεται απαραίτητο πολλοί χρήστες (π.χ. σε εταιρίες) να διαχειρίζονται την ίδια διεργασία (common – task). Για τέτοιου είδους υπηρεσίες ιδανικές είναι οι συνεργατικές εφαρμογές , και οι χρήστες που είναι στο περιβάλλον αυτό ονομάζονται συνεργάτες. Κάποιες από τις δυνατότητες που έχουν οι συνεργάτες χρησιμοποιώντας τέτοια λογισμικά είναι οι εξής:

- Επικοινωνία μέσω μηνυμάτων (text chat)
- Επικοινωνία μέσω κλήσης (call)
- Διαμοιρασμός αρχείων ή φακέλων (sharing)
- Επεξεργασία αρχείων ή φακέλων (processing)

Ο όρος συνεργασία διαχωρίζεται σε 2 μέρη , η μια έχει να κάνει με την συγχρονισμένη συνεργασία (Synchronous) όπου επιτυγχάνεται όταν όλοι αλληλεπιδρούν (interacts) σε πραγματικό χρόνο(π.χ. Skype call) και η ασύγχρονη (Asynchronous) όπου η αλληλεπίδραση των χρηστών δεν επιτυγχάνετε σε πραγματικό χρόνο [**1**].

1.2 Διαλειτουργικότητα εφαρμογών στο διαδίκτυο

Η έννοια της διαλειτουργικότητας στην σημερινή εποχή είναι σημαντικό θέμα. Η ανάγκη επίτευξής της , έχει ως αποτέλεσμα το ανθρώπινο είδος να εξελίσσεται προσπαθώντας να εκμεταλλευτεί τις τεχνολογικές δυνατότητες που προσφέρονται στο διαδίκτυο και να προσαρμόζεται με βάση αυτών στις ανάγκες τις κάθε εποχής. Το επίπεδο της διαλειτουργικότητας των εφαρμογών αυξάνεται ραγδαία με την πάροδο των χρόνων , αυτό έχει ως συνέπεια όμως σε αρκετές περιπτώσεις να δυσκολεύει τους χρήστες η χρήση τέτοιων υπηρεσιών , όπου και αποτελεί

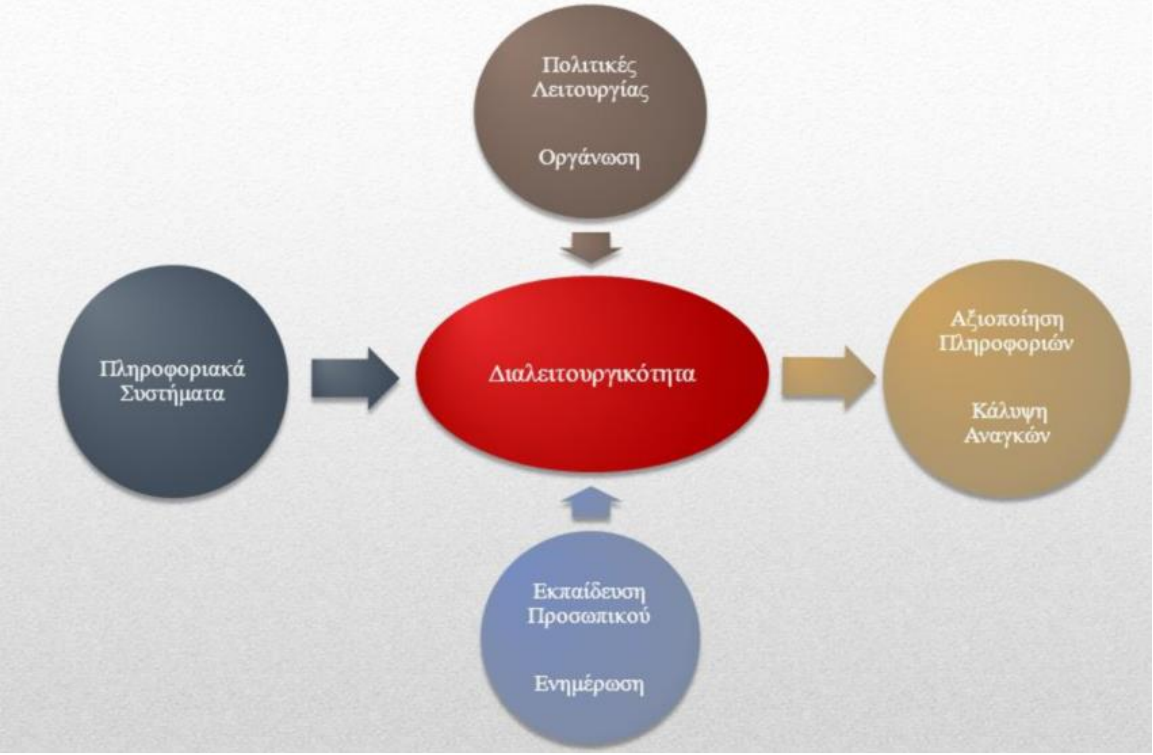
αρνητικό ζήτημα. Έχοντας υπόψη τα προαναφερθέντα κρίνεται σκόπιμο να δοθεί ένας εννοιολογικός ορισμός που αφορά την διαλειτουργικότητα συστημάτων . Σε γενικές γραμμές λοιπόν , διαλειτουργικότητα είναι η δυνατότητα ενός προϊόντος ή συστήματος -του οποίου οι διεπαφές είναι πλήρως δημόσια τεκμηριωμένες – να συνδέετε και να λειτουργεί με άλλα προϊόντα ή συστήματα , χωρίς περιορισμούς στην πρόσβασή τους ή φραγμούς στην υλοποίηση. Παρακάτω αναφέρω εννοιολογικούς προσδιορισμούς που δόθηκαν από οργανισμούς για την έννοια αυτή [**2**].

- **IEEE:** 'Η ικανότητα δύο ή περισσοτέρων συστημάτων ή βασικών στοιχείων να ανταλλάσσουν πληροφορίες και να τις επαναχρησιμοποιούν προς όφελός τους'.
- **Ευρωπαϊκό Δίκτυο Interop-NoE** : 'Η ικανότητα δύο ή περισσοτέρων συστημάτων να συνεργάζονται, χωρίς την ύπαρξη μεταβατικού σταδίου (ειδικού προγράμματος μετατροπής και «συνεννόησης») ανταλλάσσοντας πληροφορίες και υιοθετώντας τις νέες συμπεριφορές που προκύπτουν από αυτή τη συνεργασία, σε βάθος χρόνου' [**2**].
- **ISO/IEC 2382-01, Λεξιλόγιο Τεχνολογίας της Πληροφορικής : '**Η δυνατότητα επικοινωνίας, εκτέλεσης προγραμμάτων ή μεταφοράς δεδομένων μεταξύ ποικίλων και διαφορετικών λειτουργικών μονάδων, με τρόπο που απαιτείται από το χρήστη λίγη έως καθόλου γνώση, επί των μοναδικών χαρακτηριστικών αυτών των μονάδων' [**2**].
- **TMA-bridge :** 'Ο απώτερος στόχος της διαλειτουργικότητας της ηλεκτρονικής υγείας είναι να επιτρέπει σε διαφορετικούς ανθρώπους από διαφορετικές χώρες (με διαφορετικές συνήθειες , παραδόσεις , κουλτούρα, γλώσσα) να επικοινωνούν εύκολα με διαφορετικά δεδομένα και σε αλληλεπίδραση με διάφορα συστήματα που προέρχονται από διαφορετικούς κατασκευαστές ή πωλητές έχοντας όμως το ίδιο αποτέλεσμα' [**3**] .

Συμπερασματικά, ο κάθε οργανισμός δίνει τον δικό του εννοιολογικό προσδιορισμό της διαλειτουργικότητας, όσο συνεχίζει να αλλάζει ραγδαία το επίπεδο του διαδικτύου τόσοι περισσότεροι ιδικοί επί του θέματος θα βρίσκουν νέους ορισμούς για την ακριβή -όσο το δυνατόν- αποτύπωση της διαλειτουργικότητας.

Άλλος τρόπος για να αποτυπωθεί η έννοια της είναι με την χρήση τεκμηριωμένων παραδειγμάτων που αναλύουν την χρήση εφαρμογών όπου επιτελούν συγκεκριμένα καθήκοντα τα οποία στην συνέχεια μπορούν να αξιοποιηθούν από άλλες εφαρμογές ώστε οι τελευταίες να εστιάσουν σε επιμέρους ζητήματα και να επιτύχουν υψηλή προστιθέμενη αξία. Πιο συγκεκριμένα θα μπορούσαμε να εστιάσουμε στον τομέα της ηλεκτρονικής υγείας (όπου και επισημάνθηκε

προηγουμένως από τον ορισμό που δίνει το TMA-bridge). Ο όρος ηλεκτρονική υγεία σύμφωνα με το Eysenbach [**4**] αφορά στο σημείο τομής της πληροφορικής, της δημόσιας υγείας και των επιχειρήσεων. Εμπεριέχει τις υπηρεσίες υγείας και τις πληροφορίες που παραδίδονται ή εμπλουτίζονται μέσω του Διαδικτύου και των συναφών τεχνολογιών. Ένα διαδικτυακό πρότυπο που θα μπορούσε να προκύψει ώστε να διατυπωθεί ο όρος διαλειτουργικότητα είναι ένα πληροφοριακό σύστημα υγείας που ο κάθε ενδιαφερόμενος πολίτης θα μπορεί να κάνει εγγραφή αναγράφοντας τα προσωπικά του στοιχεία , όπου στην συνέχεια θα λάβει σαν απάντηση ένα φάκελο. Αυτός ο φάκελος θα καταγράφει το ιστορικό του ασθενή και την πορεία αντιμετώπισης των προβλημάτων υγείας που έχει και κατόπιν συγκατάθεσης του ασθενή να επιτρέπει την πρόσβαση στον ιατρό που τον περιθάλπει. Κάθε φορά που ολοκληρώνεται η ιατρική περίθαλψη του ασθενή αυτός ο φάκελος θα ενημερώνεται με νέα δεδομένα (π.χ. συμπέρασμα γιατρού για τις εξετάσεις που πραγματοποίησε ο ασθενής) ώστε να υπάρχει η δυνατότητα να αξιοποιηθούν στο μέλλον. Τεχνικά αυτό επιτυγχάνεται κάνοντας κλήσεις συστήματος προς το πληροφοριακό σύστημα που υπάρχουν αποθηκευμένα όλα τα ιατρικά δεδομένα που με την χρήση αυτών επιστρέφεται ως απόκριση ιατρικές πληροφορίες που κρίνονται απαραίτητες ώστε ο ιατρός να ενεργήσει ανάλογα στον ασθενή (π.χ. με βάση το ιστορικό του ασθενή να δεικνύει τι φάρμακα πρέπει να πάρει). Μπορούμε επίσης να καταλάβουμε τον σκοπό της διαλειτουργικότητας παρατηρώντας το παρακάτω σχεδιάγραμμα (βλέπε Εικόνα 1), όπου τρεις διαφορετικές οντότητες μπορούν να λειτουργήσουν με σκοπό να προκύψει εξαγωγή νέων πληροφοριών. Είναι προφανές ότι βασική προϋπόθεση για την υποστήριξη της μορφής της διαλειτουργικότητας που μόλις περιγράψαμε είναι η ύπαρξη και υιοθέτηση προτύπων που θα προσδιορίζουν επαρκώς το είδος των κλήσεων που θα επιτρέπονται και το είδος των δεδομένων που θα διακινούνται. Τέτοια πρότυπα κατά το παρελθόν ήταν δύσκολο να καθιερωθούν εξαιτίας των διαφορετικών αρχιτεκτονικών που ακολουθούσαν οι εκάστοτε κατασκευαστές εφαρμογών. Με την ανάπτυξη και ωρίμανση του διαδικτύου ωστόσο η καθιέρωση κοινά αποδεκτών προτύπων αποτέλεσε κυρίαρχη τάση με αποτέλεσμα στις μέρες μας να υπάρχει μια πληθώρα. Θα αναφερθούμε ενδεικτικά στο πρότυπο REST καθώς αποτελεί το κατεξοχήν σχετικό με την τρέχουσα εργασία πρότυπο.



Εικόνα 1 Διαλειτουργικότητα και εξαγωγή πληροφοριών

1.3 Διαδικτυακό πρότυπο REST

Το διαδικτυακό πρότυπο REST προσδιορίστηκε (πήρε την ονομασία αυτή) από τον Roy Fielding το 2000 στην διδακτορική διατριβή με τίτλο 'Architectural Styles and the Design of Network-based Software Architectures' στο UC Irvine [**5**]. Ο ορισμός που έχει επισημοποιηθεί για το πρότυπο REST (Representational state transfer) είναι ένας τρόπος παροχής διαλειτουργικότητας μεταξύ υπολογιστικών συστημάτων στο διαδίκτυο [**6**] .Το κύριο αντιπροσωπευτικό τέτοιο σύστημα είναι ο παγκόσμιος ιστός (World Wide Web) , που ουσιαστικά είναι ο βασικός λόγος που αναπτύχθηκε η αρχιτεκτονική REST , ώστε να περιγράψει ένα σύνολο από κανόνες και λειτουργίες που πρέπει να διέπουν ένα σύστημα. Η αρχιτεκτονική αυτή βασίζεται στις εξής 6 βασικές αρχές.

- Πελάτη-Εξυπηρετητή: Είναι το βασικό στοιχείο , πάνω στο οποίο βασίζονται όλα τα υπόλοιπα , ακολουθεί το πλάνο πελάτη-εξυπηρετητή με τέτοιο τρόπο ώστε η κατάσταση καθενός από τα δυο αυτά μέρη να είναι τεχνολογικά ανεξάρτητη , διατηρώντας διακριτούς ρόλους στην επικοινωνία τους.

- Ομοιομορφία διεπαφής: Η επικοινωνία μεταξύ πελάτη-εξυπηρετητή θα πρέπει να ακολουθούν τα καθιερωμένα πρότυπα για τα αιτήματα(request) που λαμβάνουν. Κάθε αίτημα που γίνεται από τον πελάτη , περιέχει όλη την απαραίτητη πληροφορία για τις υπηρεσίες που παρέχονται στο αίτημα , και η κατάσταση της συνεδρίας παραμένει στον πελάτη. Εάν δεν τηρηθούν τα καθιερωμένα πρότυπα τότε η επικοινωνία μεταξύ των δυο οντοτήτων θα καταρρεύσει.
- Λανθάνουσα μνήμη (Cash): Υπάρχει η δυνατότητα αποθήκευσης πληροφορίας σε μνήμη cache σε οποιοδήποτε επίπεδο επικοινωνίας , όπου έχει ως συνέπεια να μεταφέρετε λιγότερος όγκος πληροφορίας με αποτέλεσμα η εφαρμογή να είναι γρηγορότερη και με αξιοπιστία.
- Σύστημα πολλαπλών επιπέδων (Layered system): Ένας πελάτης δεν έχει την δυνατότητα συνήθως να αναφέρει ότι είναι συνδεδεμένος με τον διακομιστή γι' αυτό χρησιμοποιείται ένα ενδιάμεσο επίπεδο. Το ενδιάμεσο επίπεδο δεν θα πρέπει να έχει άμεση αλληλεπίδραση με τον τελικό εξυπηρετητή.
- Code on demand: Ο πελάτης κάποιες φορές κρίνεται απαραίτητο να χρησιμοποιεί μια τεχνολογία μόνο εάν το ζητήσει ο server, εάν γίνει αυτό τότε θα πρέπει να αναγνωρίσει και να εκτελέσει με επιτυχία το περιεχόμενο αυτής της τεχνολογίας. Τέτοιων ειδών τεχνολογίας θα μπορούσε να ήταν κώδικας JavaScript , Flash video , Java applet.
- Ανεξαρτησία κατάστασης: Δεν είναι υποχρεωμένος ο server να ξέρει τις προηγούμενες καταστάσεις που πραγματοποιήθηκαν από τον πελάτη , απαιτώντας με αυτόν τον τρόπο να γίνονται γνωστές όλοι οι παράμετροι για τη σωστή επικοινωνία μεταξύ αυτών.

Ακολουθούνται συγκεκριμένοι κανόνες για να δημιουργηθεί ένα REST API . Το σημαντικότερο είναι ο αναγνωριστικός σχεδιασμός της επικοινωνίας μεταξύ των δυο πλευρών μέσω του URI. Είναι μια συμβολοσειρά από χαρακτήρες που προσδιορίζει την πηγή στο διαδίκτυο. Η διεύθυνση URI(Uniform Resource Identifier) μπορεί να θεωρηθεί και ως URL (Uniform Resource Locator) κάποιες φορές. Χαρακτηριστικά παραδείγματα URI's είναι τα εξής:

- https://example.org/absolute/URI/with/absolute/path/to/resource.txt
- https://example.org/absolute/URI/with/absolute/path/to/resource
- ftp://example.org/resource.txt

Η αναπαράσταση των πόρων γίνεται τις περισσότερες φορές σε κάποια μορφή κειμένου , όπου οι πιο διαδεδομένες είναι οι XML(eXtensible Markup Language) , JSON (JavaScript Object Notation). Στην παρούσα πτυχιακή θα χρησιμοποιηθεί η μορφή JSON , όμως για να ακολουθηθεί η αρχιτεκτονική REST , μια αναπαράσταση JSON απαιτείται να ακολουθεί τουλάχιστον τις στοιχειώδης αρχές που αναφέρθηκαν προηγουμένως. Έστω ένας χρήστης χρησιμοποιεί μια εφαρμογή εταιρίας , που αναγράφει πληροφορίες για τον κάθε εργαζόμενο. Οι πληροφορίες που αναφέρονται για τον κάθε εργαζόμενο είναι το όνομα , ο μισθός και η ηλικία του , εάν ο χρήστης θέλει να μάθει τα στοιχεία αυτών θα στείλει το request από την εφαρμογή (για παράδειγμα μέσω κάποιου button) προς το σύστημα , η απάντηση(response) που θα πάρει είναι της μορφής που εμφανίζεται στην Εικόνα 2 παρακάτω.

```
{ "empinfo" :
   {
        "employees" : [
        {
          "name" : "Scott Philip",
          "salary" : £44k,
          "age" : 27,

        },

        {
          "name" : "Tim Henn",
          "salary" : £40k,
          "age" : 27,
        },

        {
         "name" : "Long Yong",
         "salary" : £40k,
         "age" : 28,

        }
                  ]
    }
}
```

Εικόνα 2 μορφή JSON σε παράδειγμα

1.4 Εστίαση και στόχοι

Έχοντας παρουσιάσει τις βασικές πτυχές της έννοιας της διαλειτουργικότητας εφαρμογών / υπηρεσιών του διαδικτύου, είναι σκόπιμο να αναφερθούμε συνοπτικά στην εστίαση και τους

στόχους της παρούσας εργασίας. Η κύρια έμφαση της εργασίας είναι η εξοικείωση με μια κατηγορία διαδικτυακών εφαρμογών που στις μέρες μας συνιστούν το συνεκτικό ιστό σε ένα ευρύ φάσμα εργαλείων και συστημάτων λογισμικού. Οι εφαρμογές αυτές ονομάζονται διεπαφές προγραμματισμού εφαρμογών (APIs) και προσφέρονται από κατασκευαστές λογισμικού ή/και πλατφορμών λογισμικού προς τρίτους προκειμένου να υποστηριχθεί η ανάπτυξη και διαλειτουργικότητα διαδικτυακών εφαρμογών. Με άλλα λόγια, ένα API είναι μια βιβλιοθήκη από εκτελέσιμες / υπηρεσίες που συνιστούν ένα είδος διεπαφής του προγραμματιστή με τις λειτουργίες που επιτελεί κατά την εκτέλεσή της ένα λογισμικό ή πλατφόρμα. Το ιδιαίτερο χαρακτηριστικό των ρουτινών αυτών είναι ότι μπορούν να ενσωματωθούν σε κώδικα από ένα προγραμματιστή προκειμένου να δημιουργηθούν εφαρμογές που να αξιοποιούν τις δυνατότητες που παρέχει το κάθε API. Στη παρούσα εργασία θα εστιάσουμε σε δύο διαφορετικά API΄s που όπως θα δούμε και στη συνέχεια ο συνδυασμός τους προσφέρει αυξημένες δυνατότητες για εφαρμογές υψηλής προστιθέμενης αξίας.

1.5 Οργάνωση πτυχιακής

Η αναφορά περιλαμβάνει 7 κεφάλαια. Στο τρέχων κεφάλαιο 1 παρουσιάσαμε συνοπτικά το ευρύτερο πεδίο και τη στόχευση της πτυχιακής εργασίας. Το κεφάλαιο 2 παρουσιάζει την έννοια του API και συνοψίζει ενδεικτικά παραδείγματα ανά κατηγορία διαδικτυακών υπηρεσιών. Το κεφάλαιο 3 εξειδικεύει και αναλύει τα API των υπηρεσιών που αποτέλεσαν αντικείμενο μελέτης στη παρούσα εργασία. Ακολούθως, το κεφάλαιο 4 εστιάζει στην αναλυτική περιγραφή της χρήσης του Trello API και στο κεφάλαιο 5 παρουσιάζεται η αναλυτική περιγραφή του Realtime API. Στο κεφάλαιο 6 παρουσιάζονται ενδεικτικές περιπτώσεις συνδυασμού των δυο διαδικτυακών υπηρεσιών και τα συμπεράσματα που προέκυψαν. Τέλος στο κεφάλαιο 7, παρουσιάζεται ένα γενικό συμπέρασμα από την ολοκλήρωση της εφαρμογής όπως και τυχόν μελλοντικές επεκτάσεις που θα μπορούσαν να δρομολογηθούν.

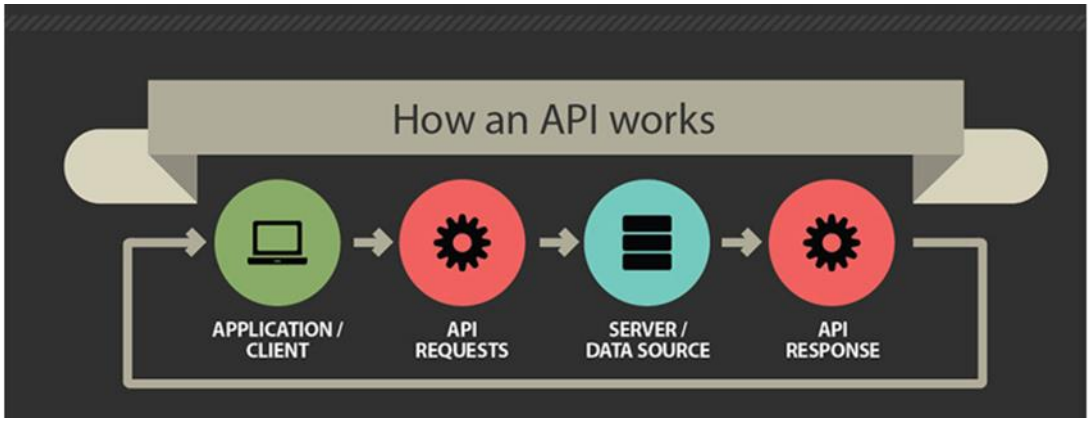
2 APIs διαδικτυακών υπηρεσιών

2.1 API σε γενικές γραμμές

2.1.1 Τι είναι ένα API

Η Διεπαφή Προγραμματισμού Εφαρμογών ή Application Programming Interface (API) αποτελεί το σύνολο των προγραμματιστικών διαδικασιών που διαθέτει ένα υπολογιστικό σύστημα ή βιβλιοθήκη προκειμένου να επιτραπούν κλήσεις προς αυτό από άλλα λογισμικά για την ανάκτηση ή ανταλλαγή δεδομένων. Πιο συγκεκριμένα το API ενός συστήματος ή υπηρεσίας ορίζει με ποιες εξωτερικές εντολές υποστηρίζεται η αμφίδρομη (request – response) επικοινωνία του συστήματος ή της υπηρεσίας με άλλα συστήματα ή υπηρεσίες. Πριν αναφερθούμε σε λεπτομέρειες θα ήταν ίσως σκόπιμο να κατανοήσουμε την έννοια του API κάνοντας ένα παραλληλισμό με τη λειτουργία ευρέως γνωστών υπηρεσιών (π.χ. κοινής ωφέλειας) όπως η υπηρεσία αποστολής γραμμάτων που προσφέρεται από το ταχυδρομείο. Αξίζει να σημειωθεί ότι ανεξαρτήτως παραρτήματος ή υπαλλήλου οι κανόνες οι οποίοι πρέπει να ακολουθηθούν για την υποβολή ενός αιτήματος αποστολής μιας επιστολής (δηλ. δήλωση διεύθυνσης παραλαβής, γραμματόσημο, ονοματεπώνυμο παραλήπτη, κτλ) είναι καλώς και αυστηρά ορισμένοι. Καλώς ορισμένα είναι και τα γεγονότα ή βήματα που διεκπεραιώνει ο μηχανισμός του ταχυδρομείου για την υλοποίηση του αιτήματος (παρότι πρόκειται για εν πολλοίς αθέατα γεγονότα προς τον χρήστη της υπηρεσίας). Επομένως, στο εν λόγω παράδειγμα, θα μπορούσαμε να ισχυριστούμε ότι το σύστημα του ‘ταχυδρομείου’ προσφέρει στους πελάτες του ένα σύνολο υπηρεσιών των οποίων η κλήση – αξιοποίηση πραγματοποιείται με προδιαγεγραμμένο τρόπο ο οποίος ενεργοποιεί συγκεκριμένες ενέργειες από το μηχανισμό και τους υπαλλήλους του ταχυδρομείου .

Όταν τα εμπλεκόμενα μέρη (δηλ. σύστημα που υποβάλλει αιτήματα και σύστημα που τα διεκπεραιώνει) είναι αμιγώς ψηφιακά, τα APIs καλούνται να τυποποιήσουν τη διασύνδεση των μερών. Οι εντολές κλήσης των APIs συνήθως είναι γραμμένες σε συγκεκριμένη προγραμματιστική γλώσσα ανάλογα τον τύπο του API. Έτσι, έχουν αναπτυχθεί πλήθος προτύπων που μπορούν να αξιοποιηθούν προκειμένου να δρομολογηθούν διασυνδέσεις με γνωστές υπηρεσίες και συστήματα. Ενδεικτικά παραδείγματα είναι τα διαδικτυακά πρότυπα όπως τα Facebook API , Youtube API , Twitter API και πολλά άλλα με εκατομμύρια κλήσεις καθημερινά ανά τον κόσμο.



Εικόνα 3 Λειτουργία ενός API γενικά

Στην Εικόνα 3 αποτυπώνεται διαγραμματικά η λογική ενός API που διαμεσολαβεί μεταξύ υπηρεσιών του διαδικτύου. Το κάθε συστατικό αναλύεται ως εξής :

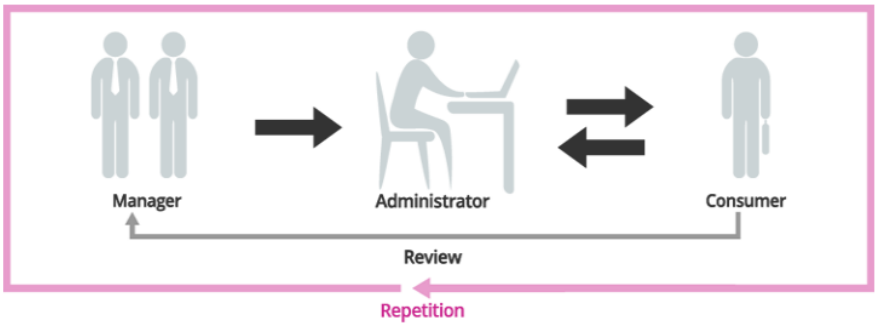
- Application/Client: Είναι η εφαρμογή που τρέχει στην μηχανή του πελάτη και αλληλοεπιδρά μαζί της μέσω των διεπαφών που υπάρχουν για να στείλει requests ανάλογα κάθε φορά με τι λειτουργία θέλει να αξιοποιήσει.
- API requests: Αποτελείτε από μηχανισμούς που δημιουργούνται από την εφαρμογή, και είναι αυστηρά καθορισμένα.
- Server/Data Source: Είναι αποθηκευμένα όλα τα απαραίτητα δεδομένα για να επέλθει απάντηση ώστε να ενημερωθεί ο χρήστης.
- API Response: Στο συστατικό αυτό περιγράφεται ο μηχανισμός που στέλνει απάντηση στο σύστημα (application) το οποίο διαχειρίζεται ο πελάτης.

Το ένα συστατικό με το άλλο συνδέεται άμεσα , εάν επέλθει κάποιο σφάλμα σε ένα από αυτά τότε δεν θα επέλθει σωστή επικοινωνία μεταξύ των δυο άκρων , δηλαδή μεταξύ Client – Server. Το ίδιο συμβαίνει και στο REST πρότυπο , όλα είναι αυστηρά καθορισμένα όπως επίσης και στα δύο υπάρχουν ενδιάμεσα επίπεδα που έχουν ως στόχο την ομαλή επικοινωνία πελάτη-εξυπηρετητή.

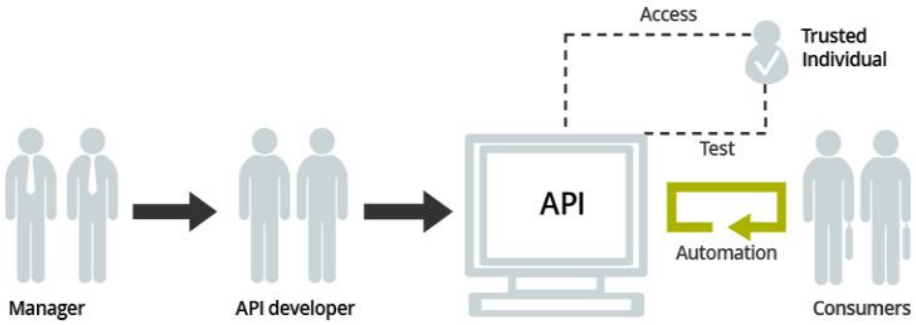
2.1.2 Πλεονεκτήματα χρήσης API

Τα πλεονεκτήματα που προσφέρει ένα API είναι αρκετά, με το σημαντικότερο να είναι αυτό της αλληλεπίδρασης μεταξύ συστημάτων λογισμικού. Ειδικότερα, δημιουργώντας μια εφαρμογή που αξιοποιεί ένα API διαδικτυακής πλατφόρμας, επιτρέπει στην ίδια την εφαρμογή να ενημερώνεται για επιλεγμένα γεγονότα που έχουν πραγματοποιηθεί στην πλατφόρμα χωρίς να είναι απαραίτητη

η σύνδεση με την πλατφόρμα.Επίσης χρησιμοποιώντας το API μιας πλατφόρμας (με σκοπό την δημιουργία άλλης εφαρμογής), δίδεται η δυνατότητα ανάπτυξης νέων υπηρεσιών προς τους χρήστες, που δεν υποστηρίζονται από τη βασική έκδοση της πλατφόρμας, δημιουργώντας με τον τρόπο αυτό υψηλή προστιθέμενη αξία. Ας σημειωθεί τέλος ότι η διαχείριση εφαρμογών που δεν αξιοποιούν τέτοιου είδους υπηρεσίες εμπεριέχουν επιπλέον επίπεδα διαμεσολάβησης π.χ. επόπτες ή διαχειριστές (βλέπε Εικόνα 4) που συχνά απαιτούν υψηλό συνολικό κόστος και ρίσκο. Αντίθετα, η χρήση ενός API παρέχει αξιοπιστία, εμπιστοσύνη και ταχύτητα (στην εξυπηρέτηση των πελατών) λόγω των αυτοματοποιημένων ενεργειών που διαθέτει η κάθε κλήση συστήματος (βλέπε Εικόνα 5) και του ενιαίου τρόπου υπολογισμού της απόκρισης - αποτελέσματος.



Εικόνα 4 Σενάριο εφαρμογής χωρίς την χρήση API



Εικόνα 5 Σενάριο εφαρμογής με την χρήση API calls

Ωστόσο, ένα μειονέκτημα εφαρμογών ή υπηρεσιών που αξιοποιούν APIs για τη διασύνδεση με τρίτους είναι αυτό της εξάρτησης τους από αυτούς (δηλ. τις λειτουργίες της πλατφόρμας στις οποίες επιτρέπει πρόσβαση το API). Αν για κάποιο λόγο οι υπηρεσίες αυτές ή η ίδια η πλατφόρμα τεθούν εκτός λειτουργίας (είτε εξαιτίας λάθους είτε για λόγους ενημέρωσης) τότε είναι προφανές ότι επηρεάζεται το ευρύτερο eco-σύστημα και οι εφαρμογές που το συγκροτούν. Συνοψίζοντας, η χρήση τέτοιων υπηρεσιών έχει περισσότερα θετικά αποτελέσματα, γι' αυτό και υπάρχει υψηλή

αποδοχή των APIs από την κοινότητα των προγραμματιστών, κάτι που αναγνωρίζεται διεθνώς (βλέπε programmable web).

2.2 API ανά κατηγορία υπηρεσιών

Έχουν δημιουργηθεί κατά καιρούς πολλά API's, τα οποία δεν σχετίζονται μεταξύ τους (διότι κάθε πρότυπο παρέχει τις δικές του υπηρεσίες) αλλά σίγουρα διευκολύνουν τον χρήστη και επίσης καθιστούν τις εφαρμογές περισσότερο αξιόπιστες και γρήγορες. Στις παρακάτω υπό-ενότητες θα αναφερθούμε σε μερικές κατηγορίες API's και μερικά παραδείγματα εφαρμογών τους στην κάθε μια από αυτές.

2.2.1 Διαμοιρασμός αρχείων (file sharing) στο διαδίκτυο

Οι υπηρεσίες αποθήκευσης στο «νέφος» (cloud storage) παρέχουν τη δυνατότητα εναπόθεσης και ανάκτησης αρχείων σε διασυνδεδεμένα αποθηκευτικά μέσα που συνήθως είναι διαθέσιμα μέσω του διαδικτύου και προσβάσιμα από εξουσιοδοτημένο χρήστη που έχει στην κατοχή του ένα υπολογιστή ή συσκευή με σύνδεση στο Internet. Επιπλέον, οι εφαρμογές αυτές παρέχουν τη δυνατότητα διαμοιρασμού αρχείων με άλλους (π.χ. με συναδέλφους ή μαθητές μας) καθώς και δημιουργία και διαχείριση κοινόχρηστων φακέλων (shared folder) με άλλους. Με την χρήση τέτοιων υπηρεσιών μειώνεται ο κίνδυνος να χαθούν πολύτιμα αρχεία και φάκελοι, διότι είναι πολύ αξιόπιστες. Οι πιο δημοφιλής εφαρμογές που έχουν αναπτυχθεί από κορυφαίες εταιρείες του κλάδου είναι οι iCloud, Dropbox, SkyDrive και Google Drive. Σε ορισμένες περιπτώσεις, όπως αυτή του google Drive, προσφέρονται και περισσότερο εξειδικευμένες υπηρεσίες που προσφέρουν επιπλέον δυνατότητες όπως π.χ. δημιουργία αρχείων κειμένου, παρουσιάσεων, υπολογιστικών φύλλων, ερωτηματολόγια και δημοσιοποίηση αρχείων προκειμένου αυτά να μπορούν να αναζητηθούν με μηχανές αναζήτησης [**7**] .

Πιο συγκεκριμένα, το Google Drive κυκλοφόρησε στις 24 Απριλίου του 2012 [**8**] και προσφέρει σε όλους τους χρήστες έναν αρχικό online χώρο αποθήκευσης 15GB που μπορεί να χρησιμοποιηθεί από τις τρείς πιο διαδεδομένες υπηρεσίες Google Drive, το Gmail και της φωτογραφίες του Google+. Επίσης ο κάθε χρήστης μπορεί να λάβει επιπλέον χώρο αποθήκευσης, από 100 GB μέχρι και 16TB μέσω της καταβολής μηνιαίας συνδρομής (US4.99$ το μήνα για 100GB).

Το Google Drive προσφέρει ένα αξιόπιστο API το οποίο επιτρέπει σε εφαρμογές τρίτων να αξιοποιήσουν κάποιες από τις λειτουργίες του. Ένα σενάριο χρήσης που θα μπορούσαμε να αναφέρουμε είναι η διαδικασία διαμοιρασμού φακέλου που φιλοξενείται στο Google Drive με κάποιον συνεργάτη. Στη τρέχουσα έκδοση της, η υπηρεσία απαιτεί αρχικά από τον χρήστη να δημιουργήσει τον φάκελο και στην συνέχεια πατώντας το share button να λάβει ως απόκριση ένα παράθυρο στο οποίο μπορεί να πληκτρολογήσει το e-mail του χρήστη με τον οποίο θα διαμοιραστεί τον φάκελο αυτό. Εάν ολοκληρωθεί η διαδικασία αυτή με επιτυχία τότε θα μπορούν να βλέπουν -και οι 2- τα αρχεία που υπάρχουν στον φάκελο εκείνο. Επισημάνθηκε αυτό το παράδειγμα διότι αποτυπώνει μια λειτουργία που αξιοποιήθηκε στην εφαρμογή που αναπτύχθηκε (πιο αναλυτικά στην ενότητα 5.1). Ειδικότερα, το API του Google Drive προσφέρει υλοποιημένη τη λειτουργία διαμοιρασμού φακέλου (βλέπε Εικόνα 6) απαιτώντας μόνο από τον χρήστη του API την κατάλληλη διεπαφή ενεργοποίησης της κλήσης, χωρίς καμία ενασχόληση με τον τρόπο που το Google Drive επιτελεί την εν λόγω λειτουργία (δηλ το διαμοιρασμό εγγράφου) . Έτσι, για παράδειγμα θα μπορούσε να αξιοποιηθεί ένα κουμπί που πατώντας το θα δρομολογούσε την εμφάνιση ενός διαδραστικού αντικειμένου που θα επέτρεπε στο χρήστη να εισάγει το e-mail του συνεργάτη με τον οποίο θα διαμοιραστεί τον φάκελο.

```
init = function() {
      s = new gapi.drive.share.ShareClient();
      s.setOAuthToken('<OAUTH_TOKEN>');
      s.setItemIds(['<FILE_ID>']);
}
```

Εικόνα 6 Συνάρτηση διαμοιρασμού φακέλου

2.2.2 Συστήματα διαχείρισης επαφών

Η κατηγορία αυτή είναι αρκετά χρήσιμη στους χρήστες που θέλουν να διαχειριστούν επαφές (π.χ. να δημιουργήσουν νέα, να ενημερώσουν υπάρχουσα ή να διαγράψουν κάποια επαφή). Μια από της πλέον δημοφιλής υπηρεσίες που επιτρέπουν τέτοιου είδους διαχείριση είναι το Google Contacts το οποίο δημιουργήθηκε το Μάρτιο του 2015. Σε αυτή την έκδοση επιτρέπεται σε ένα εξουσιοδοτημένο πελάτη να αποκτήσει πρόσβαση και να επεξεργαστεί τις επαφές των χρηστών που έχει εξουσιοδοτηθεί. Όλες οι επαφές είναι αποθηκευμένες στον λογαριασμό του χρήστη, ο οποίος απαιτείται να είναι λογαριασμός Google. Μέσω της εφαρμογής του πελάτη είναι δυνατή η δημιουργία νέας επαφής η οποία και αποθηκεύεται στο περιβάλλον νέφους της υπηρεσίας.

Παρόμοια, είναι δυνατή η επεξεργασία ή ακόμα και η διαγραφή επαφών. Ένα θετικό της υπηρεσίας είναι ότι προσφέρει εξειδικευμένο API που επιτρέπει μεταξύ άλλων και τη διαλειτουργικότητα της με εφαρμογές τρίτων και άλλες υπηρεσίες που έχει δημιουργήσει η Google, όπως για παράδειγμα με το Gmail. Ενδεικτικό παράδειγμα τέτοιας διαλειτουργικότητας μεταξύ του Google Contacts και του Gmail είναι ο τρόπος αναζήτησης στο Gmail. Η τρέχουσα έκδοση υλοποιεί ένα μηχανισμό προσαρμοσμένης προτροπής (adaptive prompting) όπου προτείνονται στο χρήστη, μεταξύ άλλων και επαφές που ταιριάζουν στο απόσπασμα κειμένου που έχει δακτυλογραφηθεί. Η λειτουργία αυτή είναι γνωστή και με τον όρο ‘αυτόματη συμπλήρωση (autocomplete)’ όπου δεδομένα συλλέγονται από τις επαφές που έχει ο χρήστης στο Google Contacts.

Ένα σενάριο χρήσης για το Google Contacts API προκύπτει όταν μια εφαρμογή απαιτήσει τη διαχείριση (π.χ. διαγραφή) κάποιας επαφής από την ήδη υπάρχουσα λίστα επαφών ενός Google λογαριασμού. Για τη δρομολόγηση αυτού του καθήκοντος, το Google Contacts API προσφέρει στον προγραμματιστή της εφαρμογής τη δυνατότητα να αξιοποιήσει τη λειτουργία του Google Contacts χωρίς να απαιτείται γνώση του πως αυτή διεκπεραιώνεται από την υπηρεσία Google Contacts. Ειδικότερα, ο προγραμματιστής θα πρέπει να προσθέσει στον κώδικα της εφαρμογής του την κατάλληλη κλήση (με παραμέτρους το e-mail (userEmail) του προς διαγραφή χρήστη και το μοναδικό κωδικό της επαφής αυτής (contactId), σε μορφή URL). Παρακάτω στην**Error! Reference source not found.** Εικόνα 7 παρουσιάζεται τμήμα συνάρτησης που παίρνει σαν όρισμα το URL της μορφής που προαναφέραμε και στην συνέχεια με τις κατάλληλες εντολές διαγράφεται η επαφή. Οπότε ο προγραμματιστής στην προκείμενη περίπτωση θα πρέπει να φτιάξει το ανάλογο γραφικό περιβάλλον (έστω ένα παράθυρο το οποίο θα υποχρεώνει τον χρήστη να δώσει το mail του χρήστη που επιθυμεί να διαγράψει από την λίστα επαφών του) και στην συνέχεια να προσθέσει το παραπάνω τμήμα κώδικα για να πραγματοποιηθεί με επιτυχία η ενέργεια αυτή.

```
public static void deleteContact(ContactsService myService, URL contactURL)
    throws ServiceException, IOException {
  // Retrieving the contact is required in order to get the Etag.
  ContactEntry contact = myService.getEntry(contactURL, ContactEntry.class);

  try {
    contact.delete();
  } catch (PreconditionFailedException e) {
    // Etags mismatch: handle the exception.
  }
}
```

Εικόνα 7 Τμήμα κώδικα για την διαγραφή επαφής

2.2.3 Υπηρεσίες διαμοιρασμού φωτογραφιών

Αρκετά δημοφιλής έχουν καταστεί και οι υπηρεσίες διαχείρισης και διαμοιρασμού φωτογραφιών. Οι υπηρεσίες αυτές αξιοποιούν το νέφος για την αποθήκευση και οργάνωση ψηφιακών εικόνων και συνήθως διακρίνονται σε δυο κατηγορίες. Η μια έχει να κάνει με την αυτόματη οργάνωση εικόνων (δηλ. ανάγνωση της πληροφορίας που τους παρουσιάζεται και αξιοποίησή της χρησιμοποιώντας μια αυτοματοποιημένη δομημένη οργάνωση φωτογραφιών). Η δεύτερη κατηγορία είναι η χειροκίνητη οργάνωση εικόνων όπου παρέχεται η δυνατότητα απευθείας προβολής φωτογραφιών από τους φακέλους που τις φιλοξενούν στο σκληρό δίσκο. Εφαρμογές που έχουν δημιουργηθεί για να παρέχουν τέτοιες υπηρεσίες είναι ACDSee (τρέχει στα Windows), Adobe Photoshop Album (τρέχει στα Windows , macOS) όπως και DBGallery (τρέχει στα Windows), Google Photos.

Τα τελευταία χρόνια η υπηρεσία Instagram έχει αποκτήσει ιδιαίτερη δημοφιλία . Πρόκειται για μια δωρεάν υπηρεσία κοινωνικής δικτύωσης που προσφέρει δυνατότητες επεξεργασίας και κοινοποίησης φωτογραφιών και βίντεο στο διαδίκτυο. Εγγεγραμμένοι χρήστες μπορούν να διαμοιράζονται εικόνες με επιλεγμένη ομάδα φίλων αφού συνάψουν κάποιας μορφής δεσμό – δηλ. γίνουν ακόλουθοι (followers)-. Αυτοί με την σειρά τους μπορούν να υποβάλουν σχόλια στις εικόνες αυτές όπως επίσης να διατυπώσουν γνώμη (π.χ. να δηλώσουν ότι τους αρέσει) κάποια φωτογραφία.

Όσον αφορά τώρα το API που προσφέρει η υπηρεσία, αυτό επιτρέπει σε κάποιο ενδιαφερόμενο που αναπτύσσει μια εφαρμογή να προσθέσει στην εφαρμογή του λειτουργίες που υλοποιεί το Instagram. Ένα σενάριο που μπορεί να προκύψει είναι η εμφάνιση και παρουσίαση μέσω της

εφαρμογής όλων των μέσων (βίντεο ή εικόνα) που ο χρήστης έχει δηλώσει ότι του αρέσουν στο Instagram. Μια τέτοια λειτουργία μπορεί εύκολα να ενσωματωθεί στην εφαρμογή με ένα απλό αίτημα (βλέπε Εικόνα 8) προς το API του Instagram το οποίο θα επιστρέψει στην εφαρμογή την λίστα που προαναφέραμε η οποία περιλαμβάνει μόνο τα μέσα (εικόνες,βίντεο) που είναι δημόσια.

```
https://api.instagram.com/v1/users/self/media/liked?access_token=ACCESS-TOKEN
```

Εικόνα 8 Αίτημα προς το Instagram API

2.3 Στόχος πτυχιακής

Έχοντας παρουσιάσει συνοπτικά τις βασικές αρχές ενός API και έχοντας σχολιάσει ενδεικτικές υπηρεσίες που προσφέρουν API's θα ήταν σκόπιμο να επικαιροποιήσουμε το στόχο της πτυχιακής με μεγαλύτερη σαφήνεια. Ειδικότερα, στην παρούσα εργασία εστιάζουμε σε δύο συγκεκριμένα API, αυτά της πλατφόρμας Trello και του Realtime API. Η επιλογή τους βασίζεται κυρίως στη δημοφιλία τους και κατά δεύτερο λόγο στις δυνατότητες που προσφέρονται. Η χρήση των δύο API's ήταν συνδυαστική και πραγματοποιήθηκε κατά τη φάση ανάπτυξης μιας διαδικτυακής εφαρμογής η οποία προσφέρει επιλεγμένες λειτουργίες που δεν υπάρχουν στην βασική έκδοση της πλατφόρμας Trello. Ειδικότερα, όσον αφορά την χρήση του Trello, αξιοποιήθηκαν βασικές λειτουργίες όπως η δημιουργία πίνακα, λίστας, κάρτας αλλά και η δυνατότητα διαγραφής λιστών και καρτών. Αυτές είναι κάποιες από τις λειτουργίες που παρέχει ήδη το Trello από μόνο του και οι οποίες με κατάλληλες κλήσεις του Trello API μπορούν και ενημερώνουν την εφαρμογή που δημιουργήθηκε. Όσον αφορά τις υπηρεσίες που δημιουργήθηκαν και δεν υπάρχουν στο Trello αυτές πραγματοποιήθηκαν είτε με τον συνδυασμό των δυο API's είτε με την χρήση μόνο του Realtime API στη εφαρμογή που αναπτύχθηκε..

Ο συνδυασμός των δυο API's αξιοποιήθηκε για την βελτίωση της ενημέρωσης των χρηστών που είναι συνδεδεμένοι και ενεργοί στην εφαρμογή μας σχετικά με γεγονότα που διεκπεραιώνονται μέσω της εφαρμογής και σχετίζονται με boards του Tello στα οποία οι χρήστες είναι μέλη. Ειδικότερα, προσφέρεται η δυνατότητα σε ένα συνδεδεμένο χρήστη (έστω ότι αυτός ονομάζετε Α) που χρησιμοποιεί την εφαρμογή όταν επιλέξει ένα board (π.χ. Board1, το οποίο υπάρχει και στο Trello) να μπορεί να λαμβάνει ειδοποιήσεις για ενέργειες (προσθήκη/διαγραφή λίστας,

προσθήκη/διαγραφή κάρτας, ανάρτηση σχολιασμού σε κάρτα) που πραγματοποιούνται από άλλους συνδεδεμένους χρήστες της εφαρμογής που είναι μέλη σε board στα οποία είναι μέλος και ο χρήστης Α.

Τέλος, με τη χρήση μόνο του Realtime API η εφαρμογή επιτρέπει την επικοινωνία μεμονωμένων χρηστών (κάτι που δεν υποστηρίζεται από το ίδιο το Trello) καθώς και την ενημέρωση για το πότε ένας συνεργάτης είναι συνδεδεμένος. Για την επικοινωνία μεμονωμένων χρηστών θα πρέπει να τονίσουμε ότι εξυπηρετεί την ανάγκη απευθείας επικοινωνίας μεταξύ χρηστών / συνεργατών ή μελών μιας ομάδας χωρίς κατ' ανάγκη αυτή η επικοινωνία να δεσμεύεται ή να αφορά κοινό board ή το εκάστοτε board που μπορεί να βρίσκεται ένας χρήστης εκείνη την στιγμή. Με άλλα λόγια, ο συγκεκριμένος τρόπος επικοινωνίας εξυπηρετεί μια μορφή ιδιωτικότητας εντός των πλαισίων μιας ομάδας και είναι ιδιαίτερα χρήσιμος σε περιπτώσεις όπου ένας χρήστης θέλει να επικοινωνήσει με έναν άλλο χρήστη – μέλος της ομάδας χωρίς αυτή η επικοινωνία να γίνει ευρύτερα γνωστή στους υπόλοιπους συνεργάτες. Ας σημειωθεί εδώ ότι η δυνατότητα αυτή υπερβαίνει τον κλασικό τρόπο επικοινωνίας μεταξύ χρηστών – μελών ενός board του Trello ο οποίος επιτυγχάνετε με την προσθήκη κάποιου σχόλιου σε μια κάρτα η οποία όμως είναι ορατή σε όλα τα μέλη του πίνακα. Άρα, το Trello δεν διαθέτει τη δυνατότητα αποκλειστικής αποστολής μηνύματος (ή ιδιωτικής επικοινωνίας μεταξύ μελών), γεγονός που οδήγησε στη ενσωμάτωση της συγκεκριμένης απαίτησης στην εφαρμογή που δημιουργήθηκε προκειμένου να υποστηριχθούν λειτουργίες που δεν μπορούν να πραγματοποιηθούν χρησιμοποιώντας μόνο το Trello.

3 Ανάλυση των επιλεγμένων APIs στόχου

Σ' αυτό το κεφάλαιο θα εξετάσουμε τα δύο API's που αποτέλεσαν το κατ' εξοχήν αντικείμενο μελέτης στα πλαίσια της τρέχουσας πτυχιακής εργασίας. Αυτά είναι το API της πλατφόρμας Trello και το Realtime API της Google.

3.1 Το Trello

Το Trello είναι μια διαδικτυακή εφαρμογή η οποία κατασκευάστηκε από την Fog Creek Software το 2011 και το 2017 πουλήθηκε στην Atlassian [**9**]. Βασικές λειτουργίες της πλατφόρμας είναι διαθέσιμες από όλων των ειδών τις συσκευές (ταμπλέτες, υπολογιστές, κινητά) ενώ υποστηρίζεται πλήρης συγχρονισμός μεταξύ τους. Μέχρι σήμερα υπάρχουν πάνω από 14 εκατομμύρια εγγεγραμμένοι χρήστες και πάνω από 1 εκατομμύριο καθημερινούς / τακτικούς χρήστες παγκοσμίως. Η πλατφόρμα αυτή είναι αρκετά χρήσιμη για την συνεργατική οργάνωση έργων (π.χ. καταγραφή καθηκόντων και εκκρεμοτήτων, online συνεργασία μεταξύ μελών ομάδας, ενημερώσεις, κλπ.). Εξαιτίας αυτών των λειτουργιών, το Trello είναι ιδανικό για χρήση από εταιρίες όπου πολλοί άνθρωποι διαμοιράζονται πληροφορίες μέσω διαδικτύου ή ομάδες χρηστών (ανεξαρτήτως φορέα) που εμπλέκονται με κάποιο τρόπο στην από κοινού εκτέλεση έργου. Ας σημειωθεί επίσης ότι το Trello κέρδισε το βραβείο 'Best Project Management Software Award' για το 2016.

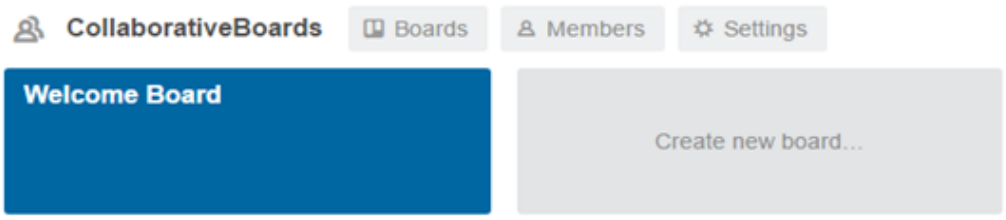
Οι λόγοι που επιλέχθηκε αυτή η πλατφόρμα για τις ανάγκες της παρούσας εργασίας είναι αρκετοί. Ο βασικότερος λόγος είναι ότι προσφέρει ένα ικανοποιητικό API και είναι εύκολο στη χρήση του. Θεωρήσαμε ότι περισσότεροι χρήστες θα προτιμήσουν τέτοια εφαρμογή, η οποία είναι αρκετά φιλική όσον αφορά την λειτουργικότητα απ' ότι μια διαφορετική πλατφόρμα που είναι σύνθετη στις λειτουργίες και δύσκολο να κατανοηθεί η χρήση της. Ουσιαστικά στην περίπτωση του Trello, μέσα σε μια κάρτα περιέχονται όλες οι απαραίτητες λειτουργίες που θα μπορεί ένας χρήστης να αξιοποιήσει, γι' αυτό είναι τόσο απλό στην χρήση του [**10**]. Επίσης η πλατφόρμα διακινεί τις πληροφορίες με ασφάλεια –χρησιμοποιεί SSL/HTTPS σύνδεση η οποία είναι τεχνολογία κρυπτογραφίας που χρησιμοποιούν και οι τράπεζες- και την κατάλληλη ιδιωτικότητα.

Το Trello μπορεί να χρησιμοποιηθεί κατά κύριο λόγο για τη διαχείριση συνεργατικών έργων. Το λογισμικό έχει όλα τα απαραίτητα εργαλεία για να μπορέσει κάποιος να διαχειριστεί οποιοδήποτε

μεγέθους project. Για παράδειγμα, θα ήταν ιδανικό για έναν φοιτητή ή μια ομάδα φοιτητών που καλούνται να ολοκληρώσουν εργασίες από κοινού ή σε συνεργασία / υπό την εποπτεία τρίτου (π.χ. του διδάσκοντα). Σε μια τέτοια περίπτωση θα μπορούσαν να δημιουργηθούν boards ανά εργασία και αντίστοιχες ομάδες. Το συνεργατικό πνεύμα που προάγει το Trello το καθιστά ιδανικό και για χρήση σε εταιρίες και οργανισμούς όπου απαιτείται διαμοιρασμός δεδομένων και πληροφορίας με άλλους χρήστες που θα ήταν συνάδελφοι σε κοινόχρηστα boards κάποιου project. Δόθηκε λοιπόν έμφαση στο εύρος των χρηστών που χρησιμοποιούν το λογισμικό αυτό , διότι μπορεί να αξιοποιηθεί από έναν χρήστη για προσωπική εξυπηρέτηση αλλά και για επαγγελματικό σκοπό.

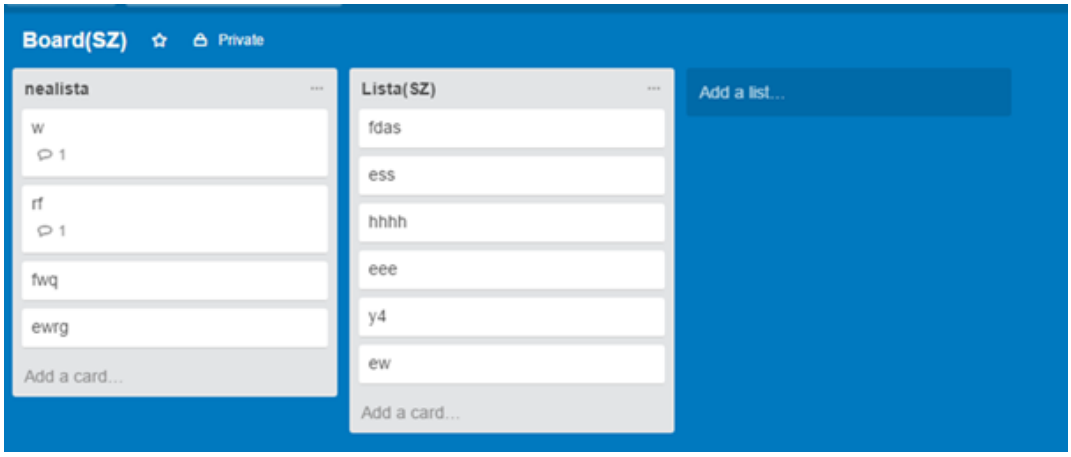
3.1.1 Λειτουργίες που υποστηρίζει το Trello

Η φιλοσοφία του Trello βασίζεται στην έννοια του board. Ένας εξουσιοδοτημένος χρήστης έχει την δυνατότητα να δημιουργήσει πίνακες (boards) και να καθορίσει τα δικαιώματα πρόσβασης. Έτσι μπορούν να δημιουργηθούν boards που να είναι είτε συνεργατικά, δηλ. τα διαμοιράζονται πολλά μέλη (team boards) όπως στην Εικόνα 9, είτε προσωπικά (personal boards) σαν αυτό στην Εικόνα 10. Επίσης, κάθε χρήστης μπορεί να έχει ή να είναι μέλος σε απεριόριστο αριθμό από boards [**11**] [**12**].



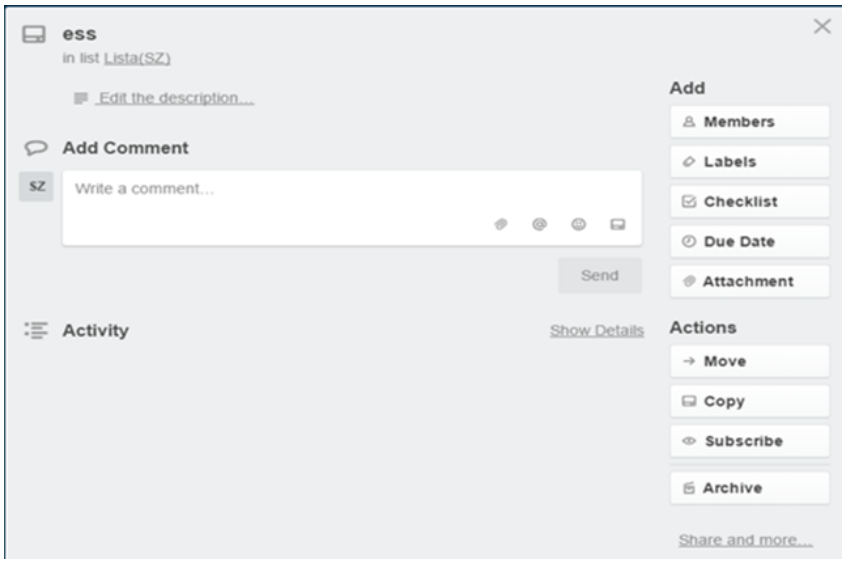
Εικόνα 9 Συνεργατικός πίνακας

Εικόνα 10 Προσωπικός πίνακας

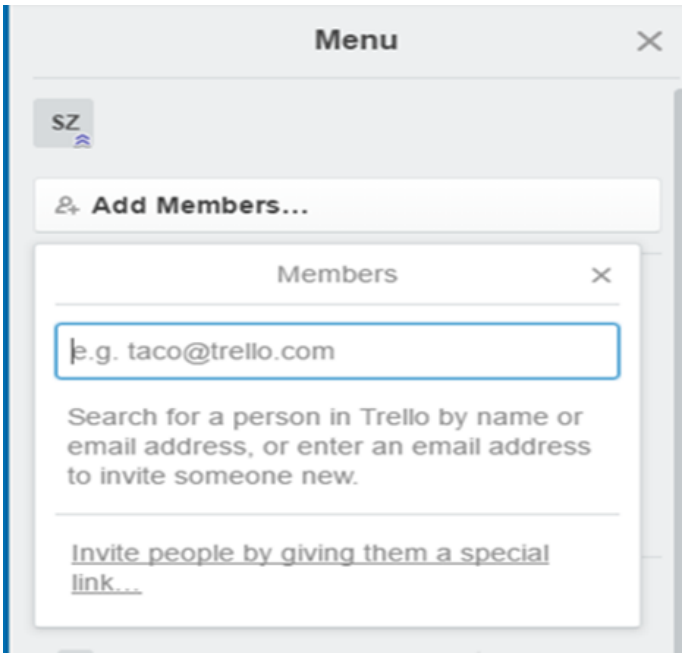


Εικόνα 11 Λίστες από κάρτες ενός πίνακα

Ένα board, ανεξαρτήτως τύπου, προσφέρει στο χρήστη πολλές δυνατότητες. Κατ' αρχάς υποστηρίζεται η δημιουργία λιστών από κάρτες (βλέπε Εικόνα 11) όπου στην κάθε κάρτα μπορεί να γίνει σχολιασμός, επισύναψη κάποιου αρχείου, προσθήκη φίλου/μέλους/συνεργάτη με δικαιώματα ενημέρωσης και πρόσβασης στο περιεχόμενο ή ανάρτηση μιας περιγραφής στην κάρτα (βλέπε Εικόνα 12). Επίσης ο χρήστης σε μια κάρτα μπορεί να αποτυπώνει κάποια εργασία, όπου έχει την δυνατότητα να καθορίσει συγκεκριμένη ώρα και ημερομηνία ολοκλήρωση της (deadline) , ενημερώνοντας έτσι όλους τους συμμετέχοντες (members) του board. Υποστηρίζεται επίσης η προσθήκη μελών σε boards, είτε με το πάτημα του κουμπιού Add Members (Εικόνα 13), είτε με πρόσκληση του χρήστη μέσω e-mail, είτε με αποστολή του (μοναδικού) συνδέσμου που αντιστοιχεί στο συγκεκριμένο πίνακα.



Εικόνα 12 Κάρτα

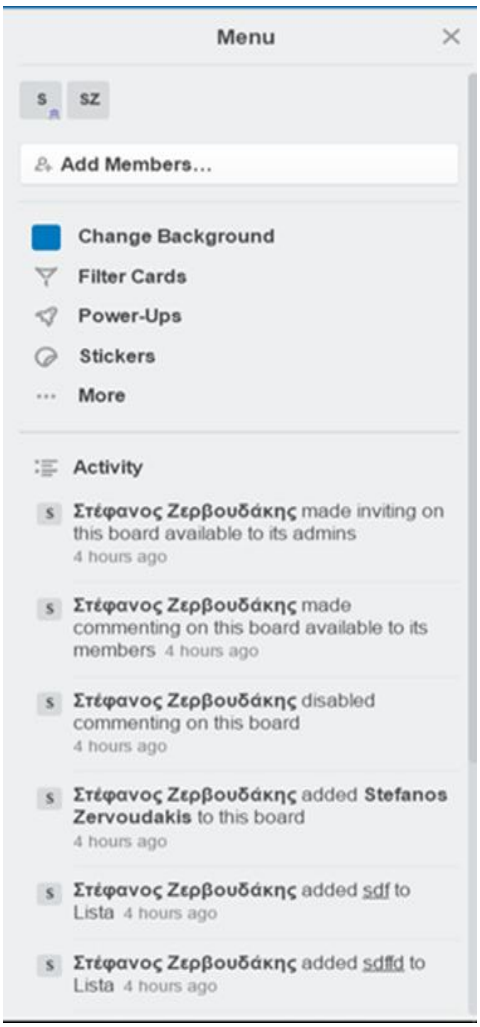


Εικόνα 13 Προσθήκη μέλους

Οι χρήστες ενός board διαχωρίζονται σε διαχειριστής και μέλος. Ο χρήστης ο οποίος είναι διαχειριστής (admin) του board έχει παραπάνω δυνατότητες από αυτή του απλού μέλος. Ένας που είναι διαχειριστής μπορεί να διαγράψει ή να κλείσει προσωρινά το συγκεκριμένο πίνακα, όπως επίσης να απορρίψει την πρόσβαση στους απλούς χρήστες να κάνουν σχόλια σε κάρτες , όπως και να προσθέσουν νέο μέλος στο board . Εν πολλοίς, οι δυνατότητες που αποκτά ένας απλός χρήστης καθορίζονται αποκλειστικά και μόνο από τον διαχειριστή.

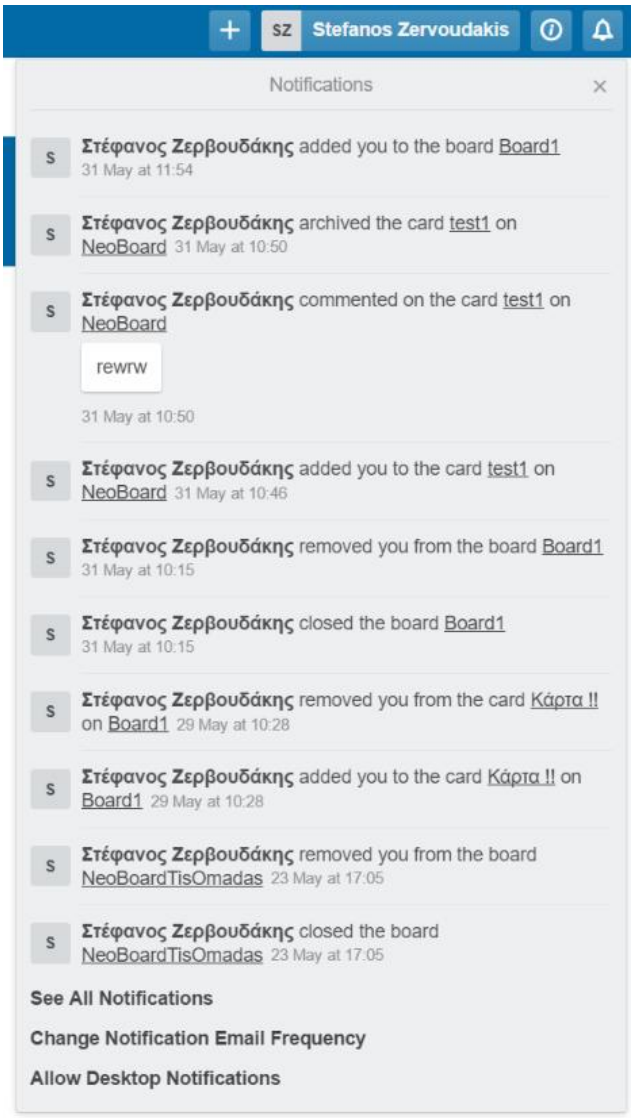
Επίσης, επιτρέπεται η προσθήκη μίκρο-εφαρμογών (Power Ups) σε ένα board από τον διαχειριστή του. Ενδεικτικές τέτοιες μίκρο-εφαρμογές που διαλειτουργούν με ένα board είναι το ημερολόγιο (calendar), ψηφοφορίες (voting), η πλαφόρμα microbloging twitter, υπηρεσίες διαμοιρασμού αρχείων όπως google drive, onedrive και διάφορα άλλα.

Σχετικά με την ενημέρωση χρηστών, υποστηρίζεται η έννοια του ιστορικού των δραστηριοτήτων (βλέπε Εικόνα 14), όπου ο χρήστης μπορεί να ενημερωθεί ανά πάσα στιγμή για ότι νέο γεγονός πραγματοποιήθηκε.



Εικόνα 14 Ιστορικό δραστηριότητας

Στο ιστορικό αυτό καταγράφεται ότι αφορά το συγκεκριμένο board (π.χ. διαγραφή κάποιας λίστας ή κάρτας , προσθήκη κάποιου σχόλιου , προσθήκη νέου μέλους). Διαθέτει επίσης και ένα πεδίο στο οποίο αναγράφονται events από πίνακες στους οποίους είναι μέλος ο χρήστης. Τα γεγονότα που αναγράφονται αφορούν την διαγραφή κάποιου board (στο οποίο ανήκει ο χρήστης), την προσθήκη σχόλιου σε κάρτα όπου είναι μέλος (βλέπε Εικόνα 15) [**13**] [**14**]. Επίσης λειτουργεί σε πραγματικό χρόνο , ουσιαστικά για ότι γεγονός πραγματοποιηθεί μπορεί ο χρήστης να το μάθει χωρίς να χρειαστεί να κάνει ανανέωση στην σελίδα. Έχει την δυνατότητα ο χρήστης να ανεβάσει κάποιο σχόλιο σε κάρτα μέσω ηλεκτρονικού ταχυδρομείου , όπως επίσης και να δημιουργήσει κάρτα και όλα αυτά διότι κάθε board διαθέτει μοναδική διεύθυνση e-mail.



Εικόνα 15 Notifications box

Ένα πραγματικό σενάριο χρήσης της εφαρμογής αυτής θα μπορούσε να ήταν, για ένα φοιτητή, ο οποίος κρατάει σημειώσεις για τις εργασίες που έχει σε κάθε μάθημα. Το κάθε μάθημα θα αποτελούνταν από ένα πίνακα (personal-board). Το περιεχόμενο του κάθε board να ήταν 3 λίστες από κάρτες. Η μια λίστα θα ήταν με τις εργασίες που έχει ολοκληρώσει μέχρι στιγμής , η άλλη λίστα από κάρτες θα ήταν με εργασίες που έχουν ολοκληρωθεί αλλά δεν έχουν εξεταστεί , και η τελευταία λίστα θα περιέχει όλες τις εργασίες που ολοκληρώθηκαν και έχουν βαθμολογηθεί. Ένα σενάριο χρήσης όπου στην προκειμένη περίπτωση η εφαρμογή θα έχει συνεργατική χρήση, είναι για ένα καθηγητή που διδάσκει ένα μάθημα , δημιουργώντας ένα πίνακα (team-board) , όπου θα μπορούσε να προσθέσει όλους τους φοιτητές (add members) που έχουν πάρει αυτό το μάθημα ,

για να τους ενημερώνει από εκεί για όποια νέα ανακοίνωση που αφορά το συγκεκριμένο μάθημα , δημιουργώντας της ανάλογες λίστες από κάρτες.

Συνοψίζοντας, το Trello είναι μια πλήρης λύση οργάνωσης στο διαδίκτυο , όπου ένας χρήστης μπορεί να το χρησιμοποιήσει είτε για προσωπική χρήση είτε για συνεργατική, ή ακόμα καλύτερα και για τα δύο.Τέλος διαθέτει αναλυτικές οδηγίες στην σελίδα της εφαρμογής για όποιον έχει δυσκολίες στο να καταλάβει την λειτουργικότητα που διαθέτει.

3.1.2 Βασικά χαρακτηριστικά του Trello API

Το Trello API, είναι μια εξαιρετικά δυνατή υπηρεσία που επιτρέπει σε προγραμματιστές χρησιμοποιώντας της μεθόδους που διαθέτει να αλληλεπιδρούν με την πλατφόρμα μέσω εξωτερικών διαδικτυακών εφαρμογών. Πιο συγκεκριμένα οι ρουτίνες που αξιοποιήθηκαν στην εξωτερική εφαρμογή που δημιουργήθηκε είναι get και post. Κάθε κλήση ρουτίνας τύπου get η τιμή επιστροφής της είναι ένα JSON (απλή μορφή δεδομένων την οποία μπορεί να αναπαράγει μια διαδικτυακή εφαρμογή). Ένα αρχείο τύπου JSON αποτελείται από πεδία (που σχετίζονται με την ρουτίνα του API) και τιμές που συνήθως είναι μοναδικά id's και ονόματα των αντικειμένων που σχετίζονται με αυτήν. Όταν γίνει κλήση ρουτίνας post, αυτό που αποστέλλεται είναι ένα JavaScript object με πεδία ανάλογα με την ρουτίνα αυτή (π.χ. για μια κάρτα τα πεδία είναι το όνομα της και το μοναδικό κωδικό που διαθέτει η λίστα στην οποία θα τοποθετηθεί). Κάποιες από τις μεθόδους που χρησιμοποιήθηκαν είναι.

- Trello.get(/member/me/boards) : Επιστρέφει τους πίνακες που είναι μέλος ο χρήστης
- Trello.get(/board/boardID/lists) : Επιστρέφει τις λίστες από ένα επιλεγμένο πίνακα
- Trello.get (/board/boardID/actions? filter=commentCard): Επιστρέφει όλα τα σχόλια που έγιναν σε επιλεγμένο πίνακα

Τέτοιου είδους μέθοδοι θα αναλυθούν διεξοδικά στην ενότητα 4.1.

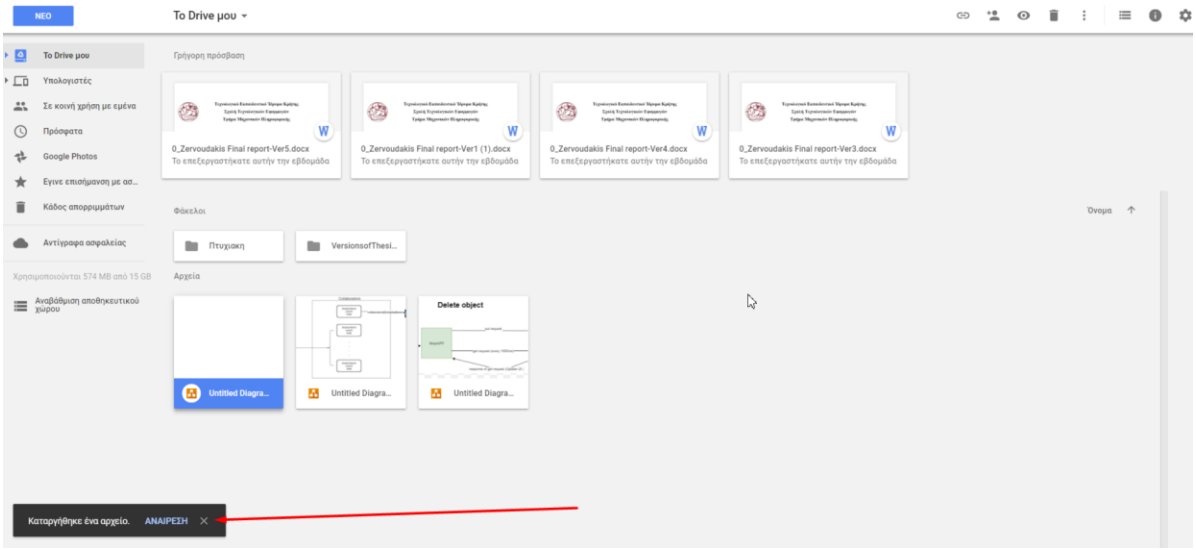
Ας σημειωθεί εδώ ότι όλες οι μέθοδοι που υπάρχουν ακολουθούν παρόμοια λογική στην σύνταξη είτε είναι τύπου get είτε post, γεγονός που καθιστά ολόκληρο το API ιδιαιτέρως προσιτό προς τον προγραμματιστή. Επιπλέον, είναι δυνατή η εύκολη δημιουργία υπηρεσιών που δεν υπήρχαν προηγουμένως αξιοποιώντας κάποιο άλλο API που δεν σχετίζεται με την πλατφόρμα, όπως για παράδειγμα στην παρούσα πτυχιακή με το Real time API (πιο αναλυτικά στην επόμενη υπό-ενότητα) .

3.2 Realtime API

Το Realtime API είναι μια JavaScript βιβλιοθήκη, την οποία δημιούργησε η Google, και την αξιοποιεί σε διάφορες εφαρμογές όπως για παράδειγμα το Google Drive και το Google docs. Το API αυτό παρέχει, συνεργατικά αντικείμενα, μεθόδους , με σκοπό την δημιουργία συνεργατικών εφαρμογών (collaborative applications). Όταν υπάρχουν εφαρμογές στις οποίες πολλαπλοί χρήστες χρειάζονται να επεξεργαστούν την ίδια πληροφορία, για παράδειγμα ένα αρχείο, η συγκεκριμένη βιβλιοθήκη είναι ιδανική. Για να γίνει αντιληπτή η λειτουργία και ενδεδειγμένη χρήση του API θα αναφερθούμε σε ένα σενάριο χρήσης. Παραδείγματος χάριν η λειτουργία της αναίρεσης είναι μια δυνατότητα που την προσφέρεται από το Google Drive και είναι αρκετά σημαντική. Μόλις ο χρήστης διαγράψει ένα φάκελο ή αρχείο εμφανίζεται ένα παράθυρο που του επιτρέπει πατώντας undo να αναιρέσει την ενέργεια που πραγματοποίησε προηγουμένως (βλέπε Εικόνα 17). Ουσιαστικά ένας προγραμματιστής που επιθυμεί να δημιουργήσει μια εφαρμογή που αξιοποιεί αυτή την λειτουργία, καλείτε να φτιάξει μόνο την διεπαφή (π.χ. ένα διαδραστικό κουμπί) καθώς το Realtime API προσφέρει υλοποιημένη την λειτουργία αναίρεσης (Εικόνα 16).

```
if (model.canUndo) {
  model.undo();
} else {
  console.log("No events to undo.");
}
```

Εικόνα 16 Μέθοδος εκτέλεσης λειτουργίας αναίρεσης



Εικόνα 17 Δυνατότητα Undo στο Google Drive

3.2.1 Μέρη από τα οποία αποτελείται εφαρμογή που κάνει κλήσεις στο Realtime API

Οι υπηρεσίες που διαθέτει μια εφαρμογή που κάνει κλήσεις στο Realtime API έχουν να κάνουν με την συνεργασία μεταξύ χρηστών. Για να πραγματοποιηθεί όμως η συνεργασία με επιτυχία θα πρέπει η εφαρμογή να διαθέτει τα παρακάτω συστατικά .

- Realtime Document : Έγγραφο (document) , το οποίο αποτελείται από πληροφορίες (data) που διαμοιράζονται μεταξύ τους οι χρήστες. Μέσα σε αυτό το έγγραφο υπάρχει το Realtime Model της εφαρμογής.
- Realtime Model : Στο Model (που αλλιώς μπορούμε να το ονομάσουμε collaborative data model) αποθηκεύονται όλες οι πληροφορίες της εφαρμογής. Ο προγραμματιστής έχει την δυνατότητα να δημιουργήσει μεθόδους και αντικείμενα ώστε με την αξιοποίηση αυτών να επέλθει συνεργασία μεταξύ των χρηστών.
- Collaborator : Έτσι ονομάζεται ο χρήστης που έχει την δυνατότητα να δει αλλά και να επεξεργαστεί τις πληροφορίες που διαθέτει μια εφαρμογή που αξιοποιεί το Realtime API.
- Drive File: Το έγγραφο που δημιουργείται αποθηκεύετε στο Google Drive του χρήστη που χρησιμοποιεί την Realtime εφαρμογή.
- Drive App: Μια εφαρμογή που αξιοποιεί το Realtime API θα πρέπει να χαρακτηριστεί ως Drive application. Ο προγραμματιστής για να το καταφέρει αυτό θα πρέπει μέσω του Google Developer's Console (πλατφόρμα που χρησιμοποιείται από προγραμματιστές που ενεργοποιούν ότι API της Google χρησιμοποιήσουν) να ενεργοποιήσει το Drive API . Όταν γίνει αυτό τότε θα έχει την δυνατότητα να αξιοποιήσει τις υπηρεσίες που διαθέτει το Drive API (π.χ. file sharing , Undo/Redo κ.τ.λ.) [**15**].

3.2.2 Collaborative Data Model

Το API παρέχει δομημένα μπλοκ (blocks), ώστε με αυτά να μπορούν να αποθηκευτούν διαφορετικών ειδών δεδομένα τα οποία θα υπάρχουν μέσα στο Data Model. Τα αντικείμενα αυτά μπορούν να τα επεξεργαστούν οι συνεργάτες όπως επίσης και να διαβάσουν την πληροφορία που υπάρχει μέσα σε αυτά. Ενδεικτικά αναφέρονται τρία αντικείμενα παρακάτω:

- Collaborative String: Αντικείμενο το οποίο είναι χαρακτήρα προς χαρακτήρα συνεργατικό . Όπου για παράδειγμα , όταν κάποιος από τους συνεργάτες αλλάξει την συμβολοσειρά (π.χ. προσθέσει χαρακτήρες) , τότε και οι υπόλοιποι θα δουν την νέα συμβολοσειρά.

- Collaborative List: Μία διατεταγμένη λίστα, την οποία μπορούν να δουν και να επεξεργαστούν όλοι οι συνεργάτες. Για παράδειγμα , σε μια τέτοια λίστα θα μπορούσαν να αναγράφονται ονόματα φακέλων και αρχείων που διαμοιράζονται μεταξύ τους οι χρήστες.
- Collaborative Object: Διαθέτει πεδία που έχει δημιουργήσει ο προγραμματιστής όπου εάν πραγματοποιηθεί αλλαγή σε ένα από τα πεδία αυτά τότε οι συνεργάτες θα ενημερωθούν για τα νέα δεδομένα που υπάρχουν σε αυτό το object. Για παράδειγμα θα μπορούσε να ήταν ένα αντικείμενο που περιέχει πληροφορίες για μια ταινία και όταν προστεθεί νέα ταινία να ενημερωθούν οι συνεργάτες για την ενέργεια αυτή.

Στο Collaborative Data Model εκτός από αντικείμενα υπάρχουν και συναρτήσεις (καλούνται ακροατές γεγονότων – event listeners) οι οποίες πυροδοτούνται κάθε φορά που πραγματοποιείται αλλαγή (π.χ. προσθήκη/αφαίρεση στοιχείων) στα παραπάνω αντικείμενα που περιγράφθηκαν με στόχο την ενημέρωση των συνεργατών. Επίσης τα γεγονότα , διαχωρίζονται σε τοπικά και απομακρυσμένα γεγονότα (local – remote events) . Εάν πυροδοτηθεί ένα event και είναι υπαίτιος κάποιος συνεργάτης τότε γι' αυτόν θα είναι local-event , ενώ για τους υπόλοιπους συνεργάτες που θα ενημερωθούν για το γεγονός αυτό θα είναι remote-event [**16**]. Αυτός είναι ένας τρόπος διαχωρισμού ενός event για τον κάθε τύπο συνεργάτη (συνεργάτης που πυροδοτεί το event με κάποια ενέργεια – συνεργάτης που ενημερώνετε για την ενέργεια αυτή) .

3.2.3 Αντικείμενα και γεγονότα που αξιοποιήθηκαν στην εφαρμογή που αναπτύχθηκε

Η αξιοποίηση του API αυτού έχει να κάνει με την ενημέρωση αλλά και την επικοινωνία μεταξύ των συνεργατών. Όσον αφορά την ενημέρωση των χρηστών που χρησιμοποιούν την εφαρμογή αξιοποιήθηκε σε όλες τις περιπτώσεις το αντικείμενο CollaborativeList και ο ακροατής γεγονότων VALUES_ADDED (που πυροδοτείτε όταν προστεθεί κάποιο στοιχείο σε συνεργατική λίστα). Πιο συγκεκριμένα, στην συνεργατική λίστα (collaborative list) ωθούνται κάποια στοιχεία (το όνομα του board που έγινε το event και το ανάλογο μήνυμα) και στην συνέχεια πυροδοτείτε ο ακροατής για να ενημερώσει τους χρήστες για τα στοιχεία που υπάρχουν στην λίστα αυτή. Επίσης για την ενημέρωση των χρηστών για τους συνδεδεμένους χρήστες αξιοποιούνται οι ακροατές CollaboratorJoinedEvent, CollaboratorLeftEvent. Ειδικότερα, ο ακροατής CollaboratorJoinedEvent πυροδοτείτε κάθε φορά που ένας συνεργάτης ανοίξει την εφαρμογή με σκοπό να ενημερώσει την διεπαφή των συνεργατών που αναγράφει το πλήθος των συνδεδεμένων

χρηστών. Ο ακροατής CollaboratorLeftEvent πυροδοτείτε κάθε φορά που ένας συνεργάτης κλείσει την εφαρμογή (document closed) όπου ενημερώνει τους χρήστες με το όνομα του συνεργάτη που μόλις αποσυνδέθηκε από την εφαρμογή. Όσον αφορά την επικοινωνία -μέσω μηνυμάτων- μεταξύ συνεργατών , εκεί αξιοποιείται το αντικείμενο collaborative list και ο ακροατής γεγονότων VALUES_ADDED (που πυροδοτείτε κάθε φορά όταν προστεθεί στοιχείο σε συνεργατική λίστα) . Μέσα σε αυτή την λίστα ωθούνται στοιχεία όπως –το μήνυμα του αποστολέα , ο μοναδικός κωδικός συνεδρίας του αποστολέα session id , μοναδικός κωδικός συνεδρίας παραλήπτη- και στην συνέχεια πυροδοτείτε ο ακροατής για να ενημερώσει τον παραλήπτη για το μήνυμα.

Συνοψίζοντας λοιπόν , χρησιμοποιήσαμε το συγκεκριμένο API , κατά βάση για να παρέχουμε στους χρήστες δυνατότητες που δεν υπήρχαν προηγουμένως στην πλατφόρμα Trello. Αυτό το καταφέραμε είτε χρησιμοποιώντας μόνο το Realtime API , χωρίς την παρέμβαση του Trello API (βλέπε κεφάλαιο Ανάλυση χρήσης Realtime API5) είτε με τον συνδυασμό των δύο API’s που μελετήθηκαν διεξοδικά (πιο αναλυτικά κεφάλαιο 0) .

4 Ανάλυση χρήσης Trello API

Η κλήση ρουτινών του συγκεκριμένου API που αξιοποίησα διαχωρίζονται σε 2 κατηγορίες , η μια είναι τύπου get όπου η απάντηση από το Trello είναι ένα JSON αρχείο , που τα περιεχόμενα αυτού είναι αντίστοιχα με την κλήση που έγινε , και η άλλη κατηγορία είναι τύπου post , όπου στέλνονται events από την εξωτερική εφαρμογή που δημιουργήθηκε προς το Trello . Αυτό έχει ως αποτέλεσμα να υπάρχει αμφίδρομη επικοινωνία μεταξύ των δυο διαδικτυακών εφαρμογών , που ήταν ένας από τους στόχους μας. Η κλήση τύπου get είναι της μορφής που φαίνεται στην Εικόνα 18

```
Trello.get('',functionForSucces,functionForFailed);
```

Εικόνα 18 Γενική μορφή κλήσης τύπου Get

Όπου σαν πρώτο όρισμα παίρνει το μονοπάτι της κλήσης συστήματος , ως δεύτερο όρισμα παίρνει την συνάρτηση που αποθηκεύονται τα στοιχεία των JSON πεδίων σε JavaScript arrays και σαν τρίτο όρισμα μια συνάρτηση που καλείτε μόνο εάν υπάρξει κάποιο σφάλμα κατά την κλήση που έγινε. Όσον αφορά κλήση τύπου post είναι της παρακάτω μορφής (βλέπε Εικόνα 19). Σαν πρώτο όρισμα παίρνει το μονοπάτι (όπως και στην get) και σαν δεύτερο όρισμα ένα JavaScript object , το οποίο περιέχει μοναδικούς κωδικούς όπως και ονόματα αντικειμένων .

```
Trello.post('',objectForPost)
```

Εικόνα 19 Γενική μορφή κλήσης τύπου Post

4.1 Εφαρμογή Get κλήσεων

Βασικό μου μέλημα ήταν η ανάκτηση όλων των πινάκων στα όποια ανήκει ο χρήστης (είτε σαν member είτε σαν admin) . Με την κλήση της συγκεκριμένης ρουτίνας (βλέπε Εικόνα 20) σαν απάντηση επιστρέφει ένα json αρχείο (Εικόνα 21) , όπου από αυτό αποθηκεύονται σε JavaScript arrays τα ονόματα των board , το μοναδικό κωδικό που διαθέτει το καθένα από αυτά , όπως και το σύνδεσμο (URL) για απευθείας μετάβαση στο Trello του πίνακα που επιλέχθηκε [**17**] .

```
Trello.get('/members/me/boards/', loadedNameofBoards, function() {console.log("failed to load boards");;
});
```

Εικόνα 20 Κλήση συστήματος για τους Πίνακες

Εικόνα 21 Μορφή JSON μετά από κλήση συστήματος για τους πίνακες

Η απεικόνιση των board του χρήστη στην εξωτερική εφαρμογή είναι ως λίστα επιλογής (option list - Εικόνα 22) , όπου όταν θέλει να μεταβεί σε ένα από τα board που είναι μέλος , το επιλέγει μέσα από την λίστα αυτή.

Εικόνα 22 Απεικόνιση των board στην εφαρμογή

Η κλήση ρουτίνας για την ανάκτηση των board γίνεται μια φορά , όταν φορτωθεί το document , δηλαδή μόλις ο χρήστης ανοίξει την εφαρμογή ή κάνει ανανέωση στην σελίδα και το πλήθος επιλογών είναι όσα και τα board που ανήκει ο χρήστης. Όταν επιλεχθεί ένα από τα board της λίστας τότε γίνονται κάποιες κλήσης συστήματος προς το API του Trello. Μια κλήση (βλέπε Εικόνα 23) αφορά τα μέλη στα οποία ανήκουν στον επιλεγμένο πίνακα. Στο μονοπάτι αναφερόμαστε σε πιο πίνακα θέλουμε να μας επιστρέψει ως απάντηση τα μέλη που ανήκουν σε αυτό , αυτό επιτυγχάνετε χάριν στο μοναδικό αναγνωριστικό κωδικό που διαθέτει (id) ο κάθε πίνακας , εάν επιστρέψει αληθής απάντηση (JSON αρχείο) τότε καλείται η συνάρτηση που εμφανίζει στην διεπαφή της εφαρμογής τα ονόματα των μελών (βλέπε Εικόνα 24) .

```
Trello.get('/boards/' + globalBoardId + '/members', printMembers, function() {console.log("failed to load members of selected board");});
```

Εικόνα 23 Κλήση συστήματος για τα μέλη ενός επιλεγμένου πίνακα

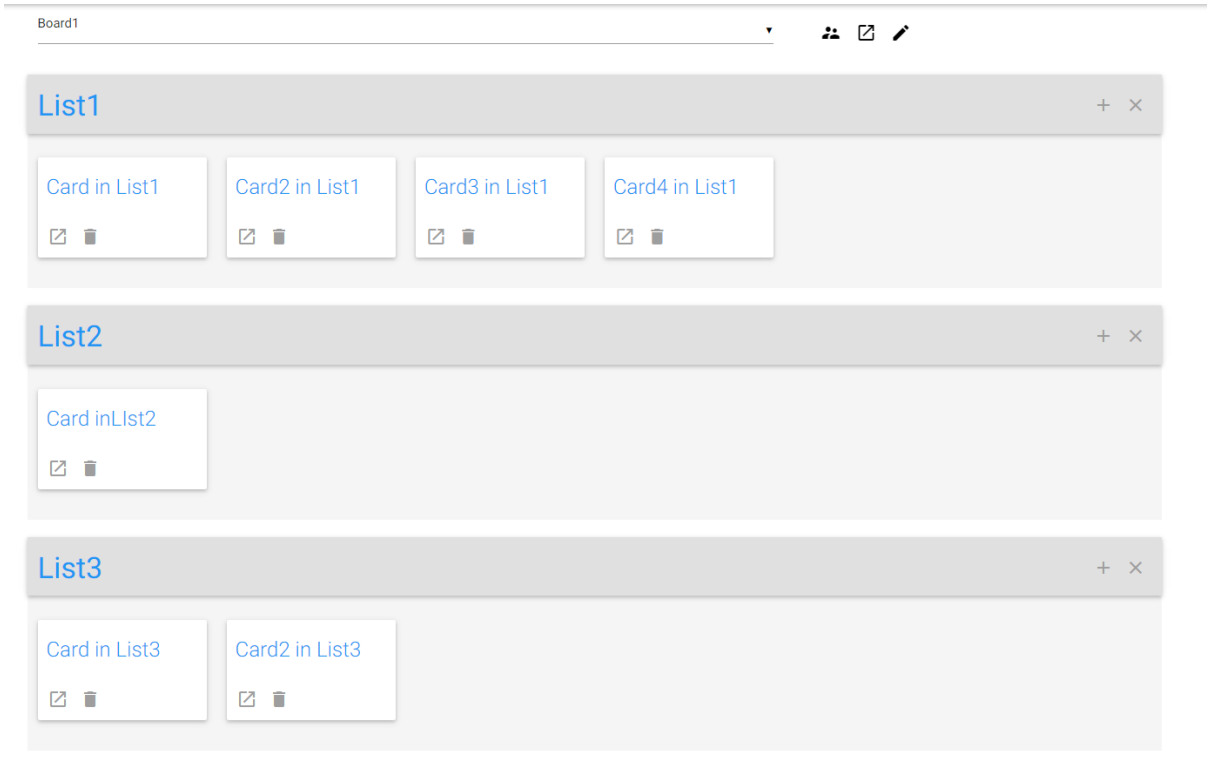
Εικόνα 24 Απεικόνιση των μελών στην διεπαφή

Η άλλη κλήση συστήματος που γίνεται , είναι για τις λίστες από κάρτες (lists of cards) του τρέχοντος πίνακα και σε αυτήν την περίπτωση θα πρέπει να κάνουμε γνωστό για πιο board

θέλουμε την ανάλογη πληροφορία , όπου γίνεται με βάση το id. Η τιμή επιστροφής της ρουτίνας διαθέτει πεδία όπως [**18**].

1. Μοναδικό κωδικό της κάθε λίστας
2. Το όνομα της κάθε λίστας
3. Μοναδικό κωδικό της κάθε κάρτας
4. Τα ονόματα των καρτών
5. Τον σύνδεσμο που διαθέτει η κάθε κάρτα (για απευθείας μετάβαση στην κάρτα στο Trello)

Από τα παραπάνω αυτά που βλέπει στην διεπαφή ο χρήστης είναι τα ονόματα των αντικειμένων δηλαδή τα ονόματα των λιστών από κάρτες (Εικόνα 25), τα μοναδικά αναγνωριστικά (id's) χρησιμεύουν για την δημιουργία ή την διαγραφή λιστών/καρτών κάτι που θα αναλυθεί στην επόμενη υπο-ενότητα διότι αφορά την χρήση post ρουτινών.



Εικόνα 25 Λίστες από κάρτες ενός επιλεγμένου board

Εκτός από τις κλήσεις που πραγματοποιούνται όταν ο χρήστης κάνει κάποια ενέργεια (π.χ. επιλέξει ένα πίνακα από την λίστα) , υπάρχουν και κλήσεις συστήματος που γίνονται αυτοματοποιημένα , δηλαδή εκτελούνται ανά συγκεκριμένο χρονικό διάστημα. Αυτές αφορούν

την ενημέρωση των συνεργατών εάν υπάρξει κάποιο γεγονός που πραγματοποιήθηκε (πιο αναλυτικά στο 6ο κεφάλαιο όπου συνδυάζονται τα δυο APIs) , αλλά και κλήσης που έχουν να κάνουν με την ενημέρωση της διεπαφής του χρήστη . Πιο ειδικά , ένα σενάριο που θα πραγματοποιούταν με επιτυχία χάριν της αυτοματοποιημένης κλήσης που επιτυγχάνετε στο Trello API είναι αυτό που δυο χρήστες έχουν επιλέξει το ίδιο board αλλά ο καθένας σε διαφορετικές εφαρμογές (ο ένας στο Trello και ο άλλος στην εφαρμογή) , εάν προσθέσει μια νέα λίστα (από τον χρήστη που είναι στο Trello) τότε θα ενημερωθεί η γραφική διεπαφή του χρήστη της εφαρμογής με την νέα λίστα. Ένας αλγόριθμος που έχει να κάνει με την ενημέρωση της διεπαφής αποτυπώνεται στην Εικόνα 26.

```
Algorithm checkLength {

   while( p<JSONArray.length ) {
       listsOfCardsCurrentBoard.nameList = JSONArray.listName
       listsOfCardsCurrentBoard.nameCard = JSONArray.cardName
      p++
   }
    If listsOfCardsCurrentBoard.length != beforeLength
         updateUI ();

    beforeLength = listsOfCardsCurrentBoard.length
}
```

Εικόνα 26 Αλγόριθμος για την ενημέρωση της διεπαφής του χρήστη

Συνοψίζοντας, η χρήση των get μεθόδων έχουν σαν πρωταρχικό στόχο την κατασκευή της διεπαφής του χρήστη , και σαν δεύτερο την ενημέρωση αυτής μετά από ένα γεγονός που πραγματοποιήθηκε είτε χρήστη της εφαρμογής είτε από χρήστη του Trello .

4.2 Εφαρμογή Post μεθόδων

Οι ρουτίνες κλήσης τέτοιου είδους έχουν να κάνουν με γεγονότα που γίνονται από την εφαρμογή με σκοπό την αλλαγή των αντικειμένων (στο πλήθος) . Οι υπηρεσίες που διαθέτει η εφαρμογή αξιοποιώντας τις post μεθόδους έχουν να κάνουν με την δημιουργία νέων αντικειμένων ή την διαγραφή υπαρχόντων αντικειμένων.

4.2.1 Δημιουργία νέων αντικειμένων μέσω της εφαρμογής

Ο χρήστης που χρησιμοποιεί την εφαρμογή έχει την δυνατότητα να δημιουργήσει νέο πίνακα , νέα λίστα όπως και νέα κάρτα. Πιο συγκεκριμένα για την δημιουργία νέου πίνακα (βλέπε Εικόνα 27), ο χρήστης θα πρέπει να δώσει το όνομα που επιθυμεί να έχει το συγκεκριμένο και πατώντας το κουμπί που ορίζει την δημιουργία πίνακα γίνεται η κλήση συστήματος (Εικόνα 28) που αφορά το γεγονός αυτό.

Εικόνα 27 Δημιουργία Πίνακα

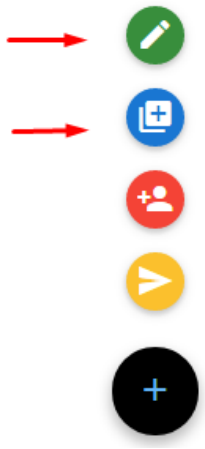
```
var board = {
  name: nameOfBoard
}

Trello.post('/board/', board);
```

Εικόνα 28 Κλήση συστήματος για την δημιουργία πίνακα

Όσον αφορά την δημιουργία λίστας σε κάποιο board , αυτό γίνεται μόνο όταν έχει μεταβεί ο χρήστης στον πίνακα που επιθυμεί να δημιουργήσει την νέα λίστα . Κρίνεται απαραίτητο για την δημιουργία λίστας να έχει επισημανθεί το id του board, όπως επίσης και το όνομα που θα έχει το νέο αντικείμενο . Από την στιγμή που ο χρήστης της εφαρμογής έχει επιλέξει τον πίνακα , το id αυτού είναι γνωστό , αρκεί μόνο να δώσει ο χρήστης το όνομα που θα έχει η λίστα αυτή , οπότε το JavaScript object θα έχει δυο πεδία , τον κωδικό του πίνακα και το όνομα της λίστας. Όσον αφορά την δημιουργία κάρτας σε κάποια λίστα η εφαρμογή παρέχει δυο διαφορετικές υπηρεσίες. Η μια είναι όταν ο χρήστης επιθυμεί να δημιουργήσει κάρτα σε πίνακα διαφορετικό από αυτόν που βρίσκεται εκείνη την στιγμή, ενώ η δεύτερη αφορά τη δημιουργία κάρτας στο board που βρίσκεται ο χρήστης μια συγκεκριμένη στιγμή. Στην πρώτη περίπτωση, πατώντας το κουμπί για την δημιουργία κάρτας, εμφανίζεται ένα παράθυρο το οποίο διαθέτει μια λίστα επιλογής απ' όπου επιλέγεται ένα από τα υπάρχοντα board , μια λίστα επιλογής απ' όπου επιλέγεται μία από τις λίστες

καρτών του επιλεγμένου πίνακα από το προηγούμενο βήμα , προσδιορισμός θέσης της νέας κάρτας (με επιλογές top και bottom δηλ. σαν πρώτη κάρτα στην λίστα ή σαν τελευταία) στη λίστα και τέλος το πεδίο δήλωσης του ονόματος της νέας κάρτας .



Εικόνα 29 Κουμπιά για την δημιουργία πίνακα και κάρτας αντίστοιχα

Η δεύτερη υπηρεσία αφορά την δημιουργία κάρτας στο board που βρίσκεται ο χρήστης εκείνη την στιγμή , που σε αυτή την περίπτωση χρειάζεται να δώσει μόνο το όνομα της κάρτας. Η ενημέρωση της διεπαφής μετά από τις παραπάνω ενέργειες γίνεται με την χρήση των ρουτινών που καλούνται ανά συγκεκριμένο χρονικό διάστημα , αρχικά πρώτα γίνεται το request τύπου post ώστε το νέο αντικείμενο να προστεθεί στο Trello , και στην συνέχεια με την αυτοματοποιημένη κλήση ρουτίνας τύπου get ενημερώνεται και η διεπαφή του χρήστη που χρησιμοποιεί την εφαρμογή. Στην Εικόνα 29 η διεπαφή για τις δύο υπηρεσίες που αναφέρθηκαν

4.2.2 Διαγραφή αντικειμένων μέσω της εφαρμογής

Μια υποκατηγορία των post ρουτινών είναι η διαγραφή των αντικειμένων (π.χ. λίστας), όπου ο χρήστης έχει την δυνατότητα να διαγράψει κάποιο από αυτά που δεν χρειάζεται πλέον. Μια κλήση τύπου ‘put’ είναι της μορφής που φαίνεται στην Εικόνα 30. Το πρώτο όρισμα είναι το μονοπάτι της κλήσης στο οποίο προσδιορίζεται το μοναδικό αναγνωριστικό του αντικειμένου (είτε είναι λίστα είτε είναι κάρτα) που θα διαγραφθεί. Το δεύτερο όρισμα δηλώνει τη συνάρτηση που θα ενημερώσει την διεπαφή του χρήστη.

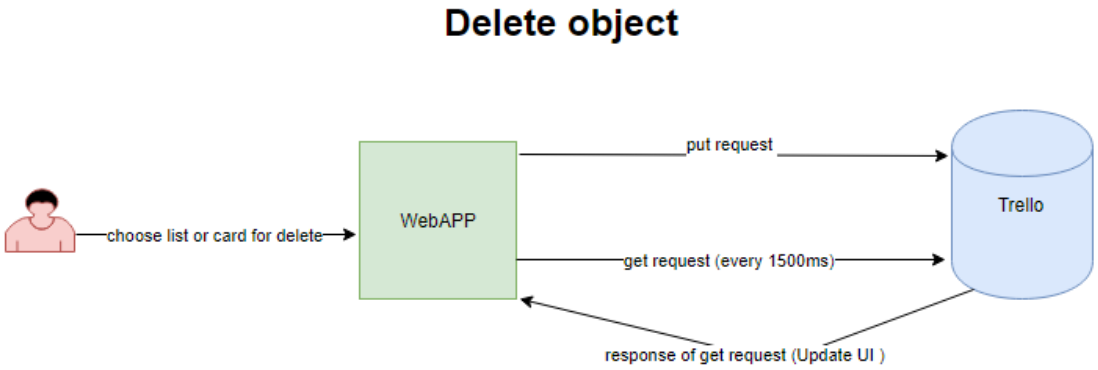
```
Trello.put('',FunctionForUpdateUI)
```

Εικόνα 30 Κλήση συστήματος για διαγραφή αντικειμένου

Σε περίπτωση που ο χρήστης επιθυμεί να διαγράψει ολόκληρη την λίστα τότε πατώντας το κατάλληλο κουμπί (βλέπε Εικόνα 31) εμφανίζεται ένα παράθυρο επιβεβαίωσης για την οριστικοποίηση της απόφασης ώστε να διαγραφθεί η λίστα που επιλέχθηκε. Η κλήση για το συγκεκριμένο γεγονός εάν πραγματοποιηθεί τότε θα ενημερωθεί η διεπαφή του χρήστη με τις λίστες που έχουν απομείνει. Το ίδιο συμβαίνει εάν ο χρήστης θέλει να διαγράψει μια κάρτα. Η κλήση είναι προφανώς της ίδιας λογικής, όπου θα πρέπει να γίνει γνωστό το μοναδικό αναγνωριστικό της κάρτας που θα διαγραφθεί όπως και η συνάρτηση ενημέρωσης UI.

Εικόνα 31 Κουμπί διαγραφής λίστας

Η αμφίδρομη επικοινωνία που επιτυγχάνεται μεταξύ εφαρμογής - Trello όσον αφορά την διαγραφή των αντικειμένων είναι αυστηρά καθορισμένη. Αρχικά στέλνονται αιτήματα από την εφαρμογή (put) ώστε να αλλάξει το πλήθος των αντικειμένων που βρίσκονται στο Trello και στην συνέχεια με τις αυτοματοποιημένες κλήσεις (get) αλλάζει και η διεπαφή του χρήστη με τις λίστες που έχουν απομείνει στον επιλεγμένο πίνακα. Παρακάτω ένα σχεδιάγραμμα για την επικοινωνία που προαναφέρθηκε (βλέπε Εικόνα 32).



Εικόνα 32 Put and Get Requests

5 Ανάλυση χρήσης Realtime API

Σε αυτή την ενότητα θα αναλυθούν οι κλήσεις που γίνονται στο Realtime API μέσω της εξωτερικής εφαρμογής με υπηρεσίες που διαθέτει προς τους χρήστες. Τέτοιου είδους κλήσεις δεν έχουν να κάνουν με το Trello. Κατά βάση είναι ενέργειες που πραγματοποιούνται ώστε να δημιουργηθούν νέες υπηρεσίες που δεν υπήρχαν προηγουμένως στην πλατφόρμα , όπως την επικοινωνία μεταξύ των συνεργατών μέσω μηνυμάτων , αλλά και την real time ενημέρωση των συνεργατών για συνδεδεμένους χρήστες.

5.1 Δυνατότητα προσθήκης συνεργατών (add people)

Ο χρήστης μπορεί μέσω της εφαρμογής να προσθέσει κάποιον συνεργάτη απλά κάνοντας γνωστό το mail του συγκεκριμένου χρήστη . Προσθέτοντας κάποιον συνεργάτη αυτό έχει ως αποτέλεσμα , χρήστες να έχουν την δυνατότητα επικοινωνίας μέσω μηνυμάτων , όπως επίσης και να ενημερώνονται για γεγονότα που επιτεύχθηκαν από αυτούς (π.χ. διαγραφή κάποιας λίστας- κάτι που θα αναλυθεί στο επόμενο κεφάλαιο) . Τεχνικά για να επιτραπεί η πρόσβαση ενός συνεργάτη στο ίδιο διαμοιραζόμενο αρχείο (στην προκειμένη περίπτωση τον ρόλο αυτού τον έχει η εφαρμογή) θα πρέπει να γίνουν γνωστά το μοναδικό id που διαθέτει το έγγραφο όπως επίσης και το access token . Το id του αρχείου δημιουργείται όταν ο χρήστης περιηγηθεί για πρώτη φορά στην σελίδα και δεν αλλάζει. Όσον αφόρα το token είναι ένας 12-ψηφιος κωδικός που δίνεται στον προγραμματιστή που κάνει κλήσεις του API που διαθέτει η Google (βλέπε Εικόνα 33).

```
s = new gapi.drive.share.ShareClient();
s.setOAuthToken('<OAUTH_TOKEN>');
s.setItemIds(['<FILE_ID>']);
```

Εικόνα 33 Dialog Script for adding collaborators

Η κλήση αυτή επιτυγχάνεται όταν ο χρήσης πατήσει το κουμπί add-people , η απάντηση που παίρνει είναι ένα modal παράθυρο , με text-πεδίο για να πληκτρολογήσει ο χρήστης το mail αυτού με τον οποίο θέλει να συνεργαστεί-επικοινωνήσει. Εάν επιτευχθεί αυτό τότε θα μπορεί ο συνεργάτης που προστέθηκε στην λίστα των συνεργατών να επικοινωνήσει με άλλους χρήστες , να ενημερώνεται για το ποιοί είναι συνδεδεμένοι ή ποιος αποσυνδέθηκε , να προσθέσει κάποιον όπως και να λαμβάνει ειδοποιήσεις σε πραγματικό χρόνο για event που πραγματοποιήθηκαν σε board που είναι μέλος. Προγραμματιστικά η αποθήκευση του μοναδικού κωδικού του κάθε

document γίνεται όταν επιτευχθεί η κλήση της συνάρτησης start (που καλείται πάντα όταν ανοίξει ο χρήστης την εφαρμογή Εικόνα 34) , όπου μέσα σε αυτή παράγεται ένα id το οποίο ελέγχεται εάν υπάρχει ήδη ή εάν δημιουργείτε για πρώτη φορά , διότι εκτελούνται διαφορετικές εντολές. Εάν ο χρήστης μπει για πρώτη φορά στην εφαρμογή (δηλαδή δεν υπήρχε προηγουμένως το id) τότε θα δημιουργηθεί το Realtime document (το οποίο αποθηκεύεται στο Drive του χρήστη της εφαρμογής) και στο σύνδεσμο τοποθετείτε το id που παράχθηκε , έπειτα καλούνται οι κατασκευαστικές συναρτήσεις για να ενσωματώσουν στο model όλα τα blocks (συνεργατικές λίστες κτλ) . Εάν όμως το id προϋπήρχε τότε απλά προσθέτει το id στο τέλος του συνδέσμου και στην συνέχεια καλούνται οι κατασκευαστικές συναρτήσεις.

```
var id;
function start() {
    id = realtimeUtils.getParam('id');
    if (id) {
            // Load the document id from the URL
            realtimeUtils.load(id.replace('/', ''), onFileLoaded, onInitialize);
    } else {
            // Create a new document, add it to the URL (when user get access to the app for first time)
            realtimeUtils.createRealtimeFile('MyWebb App', function (createResponse) {
            window.history.pushState(null, null, '?id=' + createResponse.id);
            realtimeUtils.load(createResponse.id, onFileLoaded, onInitialize);
        });
    }
     gapi.load('drive-share', init);
}
```

Εικόνα 34 Function start

5.2 Συνδεδεμένοι χρήστες στην εφαρμογή

Μια δυνατή υπηρεσία που διαθέτει η εφαρμογή είναι η real time ενημέρωση των συνεργατών για τους συνδεδεμένους χρήστες ανά πάσα ώρα και στιγμή . Πιο συγκεκριμένα , παρουσιάζονται τα ονόματα των συνεργατών που έχουν ανοιχτή την εφαρμογή , το πλήθος αυτών, όπως επίσης και το όνομα του συνεργάτη που μόλις αποσυνδέθηκε από την εφαρμογή (σε μορφή ειδοποίησης). Όλα αυτά επιτυγχάνονται με την αξιοποίηση των ακροατών CollaboratorJoinedEvent , CollaboratorLeftEvent και της συνάρτησης getCollaborators(). Οι δύο ακροατές πυροδοτούνται κάθε φορά που ένας συνεργάτης ανοίξει ή κλείσει την εφαρμογή αντίστοιχα , ενώ η τιμή επιστροφής της συνάρτησης - getCollaborators()- όταν καλεστεί είναι ένα JavaScript Object με πληροφορίες για τους συνδεδεμένους συνεργάτες της εφαρμογής. Οι πληροφορίες που αναγράφονται στο object παρατίθενται παρακάτω σε bullets.

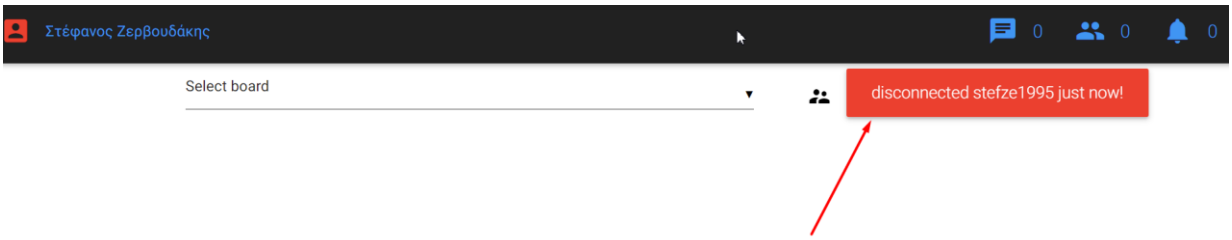
- Ονόματα συνεργατών (displayName)
- Μοναδικός κωδικός για κάθε συνεργάτη (userId)
- Μοναδικός κωδικός συνεδρίας (sessionId)
- Τιμή επιστροφής αληθής – ψευδής , εάν ο συνεργάτης είναι ανώνυμος ή όχι (isAnonymous)
- Τιμή επιστροφής αληθής-ψευδής εάν ο συνεργάτης είναι τοπικός ή όχι (isMe)

Με την αξιοποίηση λοιπόν του αντικειμένου αυτού παρουσιάζονται στην διεπαφή της εφαρμογής το πλήθος των συνδεδεμένων συνεργατών σε μια ετικέτα (βλέπε Εικόνα 35) όπως και τα ονόματα αυτών σε ένα modal παράθυρο (βλέπε Εικόνα 36).

Εικόνα 35 Ετικέτα που αναγράφει το πλήθος των συνδεδεμένων συνεργατών

Εικόνα 36 Παράθυρο που αναγράφει τα ονόματα των συνδεδεμένων χρηστών

Η διεπαφή όμως της εφαρμογής των χρηστών/συνεργατών θα πρέπει να αλλάζει δυναμικά όταν ένας συνεργάτης συνδεθεί σε αυτή , όπου αυτό επιτυγχάνεται όταν πυροδοτηθεί ο ανάλογος ακροατής και με τις απαραίτητες εντολές αλλάζει το πλήθος των συνδεδεμένων συνεργατών και το παράθυρο που αναγράφει τα ονόματα αυτών. Όσον αφορά τον ακροατή που πυροδοτείτε πάντα όταν κλείσει την εφαρμογή ένας συνεργάτης, διαθέτει τις απαραίτητες εντολές για την δημιουργία ειδοποίησης (βλέπε Εικόνα 37) αλλά και την ενημέρωση της ετικέτας που αναγράφει το πλήθος των συνεργατών.



Εικόνα 37 Ειδοποίηση συνεργατών για τον χρήστη που μόλις αποσυνδέθηκε

Οι εντολές που υπάρχουν μέσα στους ακροατές γεγονότων φαίνονται στις παρακάτω δύο εικόνες (Εικόνα 38 Εικόνα 39)

```
var NumOfColls = function (event) {
    var collaborators = doc.getCollaborators(); //το αντικείμενο που αποθηκευονται οι πληροφορίες σ
    var collaboratorCount = collaborators.length; //το πληθος των συνδεδεμενων χρηστων
    var user = event.collaborator;
    listTest.length=0;
     for(var i=0;i<collaborators.length;i++){
         if(collaborators[i].isMe){
           UserId=collaborators[i].sessionId;
           NameAccount=collaborators[i].displayName;
         }

     }
     document.getElementById("NameOfColl").innerHTML=NameAccount;

      if(collaboratorCount==0){
        document.getElementById("Num").innerHTML=0;
      }
      else{
         var num=collaboratorCount-1;
         document.getElementById("Num").innerHTML=num; //grafei to plh8os twn sunergatwn pou einai
      }

};
```

Εικόνα 38 Ακροατής γεγονότων που πυροδοτείτε όταν ένας συνεργάτης συνδεθεί στην εφαρμογή

```
var NumOfColls1 = function (event) {
   var collaborators = doc.getCollaborators();
   var collaboratorCount = collaborators.length;
   var user = event.collaborator;
   var message="disconnected "+user.displayName+" just now!";
   var isTrue;
    for(var i=0;i<collaborators.length;i++){
     if(user.displayName==collaborators[i].displayName){
           isTrue=true;
           break;
     }

    }
   if(!isTrue){
     Materialize.toast(message,"2300","card red","");
     collaboratorCount--;
     document.getElementById("Num").innerHTML=collaboratorCount;
   }


};
```

Εικόνα 39 Ακροατής που πυροδοτείτε όταν κλείσει ένας συνεργάτης την εφαρμογή

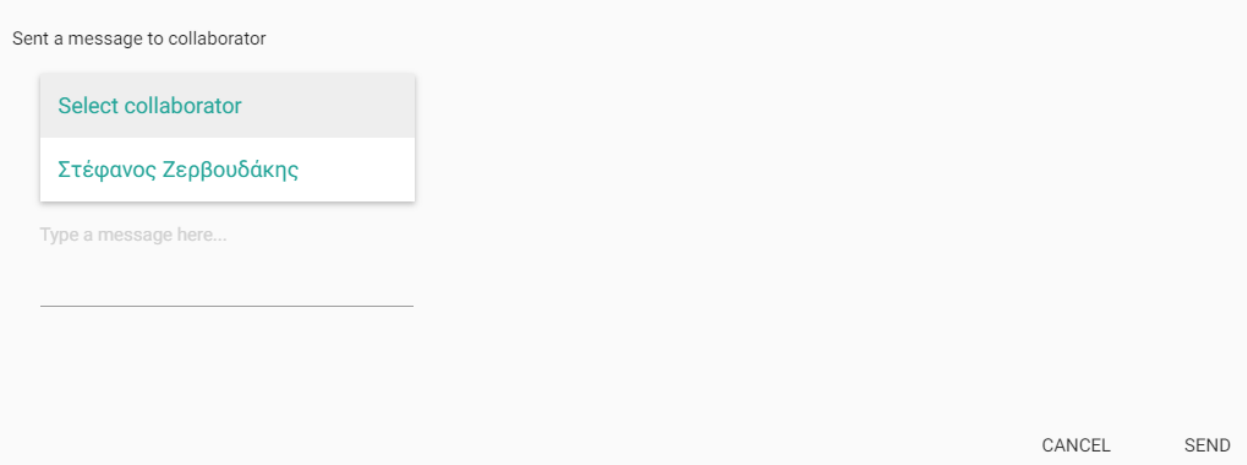
5.3 Επικοινωνία χρηστών μέσω μηνυμάτων (chat)

Μια υπηρεσία που διαθέτει η εφαρμογή και δεν υπάρχει στην πλατφόρμα Trello έχει να κάνει με την Realtime επικοινωνία μεταξύ των συνεργατών. Με τον όρο επικοινωνία εννοούμε τον διάλογο που μπορεί να προκύψει (μέσω μηνυμάτων) ανάμεσα σε δύο συνδεδεμένους χρήστες. Αυτοί οι δύο θα μπορούσαν να είναι μέλη σε ίδιο board , αλλά δεν κρίνεται απαραίτητο αυτό , διότι μπορεί να αξιοποιηθεί αυτή η υπηρεσία χωρίς να είναι μέλη σε κοινά board. Η λειτουργία επιτυγχάνεται αποκλειστικά και μόνο με την χρήση του Realtime API , χωρίς να παρέμβει κάποια κλήση συστήματος προς το API του Trello. Κάθε χρήστης διαθέτει ένα μοναδικό κωδικό (sessionId) που δημιουργείται κάθε φορά που θα ανοίξει την εφαρμογή, αυτό έχει ως συνέπεια να μπορεί ο συνεργάτης να επιλέξει σε ποιον θα σταλεί το μήνυμα αυτό μέσα από την λίστα με όλους τους συνδεδεμένους χρήστες. Στην διεπάφη του χρήστη , η παρουσίαση των ενεργών χρηστών γίνεται σε μια λίστα επιλογής (option list) , όπου διαθέτει τόσα πεδία όσοι είναι οι συνδεδεμένοι συνεργάτες –στην εφαρμογή- και εδώ γίνεται η χρήση της συνάρτησης getCollaborators , για να αξιοποιηθούν τα πεδία -από το object που επιστρέφει - displayName (αναγράφεται το όνομα στην λίστα επιλογής) sessionId (χρησιμοποιείται σαν value στο κάθε στοιχείο της λίστας επιλογής) για τον κάθε συνεργάτη. Όσον αφορά την προσωρινή αποθήκευση του μηνύματος αυτό γίνεται με την χρήση collaborative list. Αρχικά δημιουργείτε ένα JavaScript Object με στοιχεία που κρίνονται απαραίτητα ώστε να σταλεί το περιεχόμενο του μηνύματος που πληκτρολόγησε ο αποστολέας με επιτυχία στον σωστό αποδέκτη , χωρίς οι υπόλοιποι συνεργάτες να ενημερωθούν για το γεγονός αυτό. Τα πεδία που διαθέτει η οντότητα αυτή είναι τα εξής :

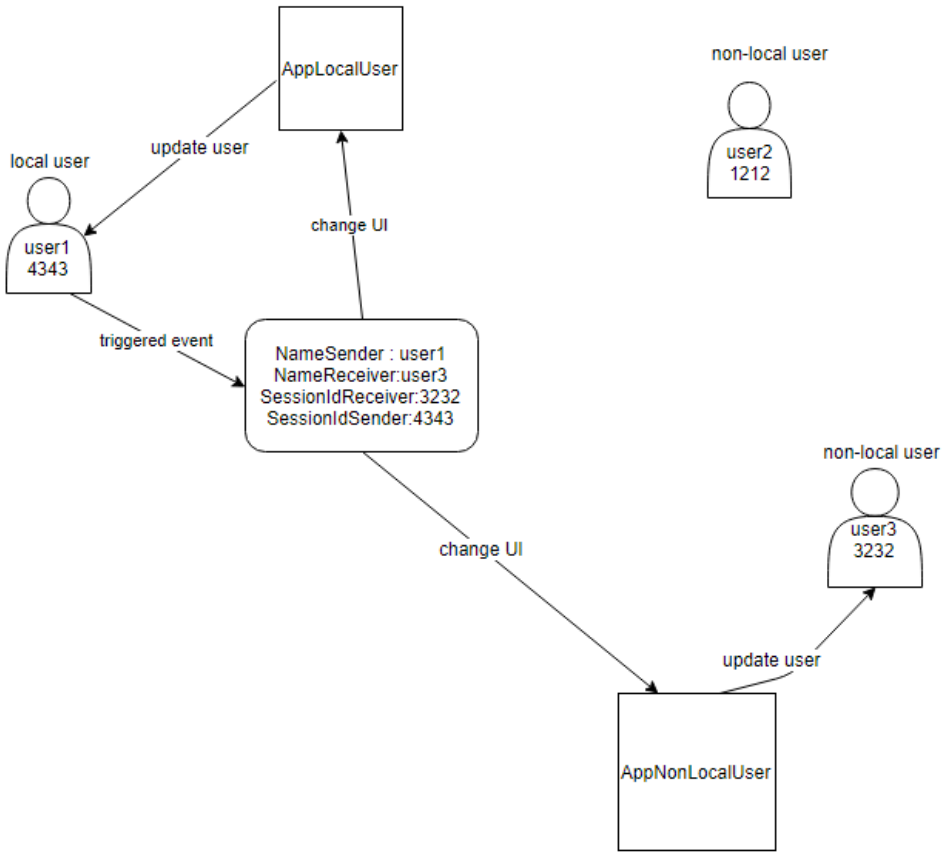
- Το περιεχόμενο του μηνύματος που πληκτρολόγησε ο αποστολέας (τύπου συμβολοσειράς)
- Το μοναδικό αναγνωριστικό του αποστολέα (sessionId)
- Το μοναδικό αναγνωριστικό του παραλήπτη του μηνύματος (sessionId)
- Το όνομα του παραλήπτη του μηνύματος

Για την αποστολή του μηνύματος στον παραλήπτη υπεύθυνος είναι ο ακροατής που πυροδοτείτε κάθε φορά που ένα αντικείμενο με τα παραπάνω πεδία προστεθεί στην collaborative list. Ο ακροατής εκτός από την αποστολή του μηνύματος , υπεύθυνος είναι και για την ενημέρωση της γραφικής διεπαφής των χρηστών .

Πιο συγκεκριμένα , γίνεται ο διαχωρισμός των χρηστών , σε τοπικούς και μη-τοπικούς χρήστες(local user – non local) οι οποίοι είναι όλοι οι ενεργοί χρήστες. Εάν στάλθηκε το μήνυμα τότε η διεπαφή του τοπικού χρήστη (αποστολέα) ενημερώνεται για το γεγονός αυτό και η διεπαφή του μη-τοπικού χρήστη (παραλήπτης) διαθέτει ένα μετρητή που καταγράφει το πλήθος των μη διαβασμένων μηνυμάτων όπου αυξάνεται κάθε φορά που ένα γεγονός αφορά αυτόν. Για να αλλάζει κάθε φορά μόνο ο μετρητής του παραλήπτη (δηλαδή να ενημερώνεται μόνο ένας για το γεγονός) καθορίζεται με το sessionId που διαθέτουν οι συνδεδεμένοι συνεργάτες. Ουσιαστικά πυροδοτείτε ο ακροατής γεγονότων για να αλλάξει την διεπαφή 2 χρηστών την φορά , του αποστολέα και του παραλήπτη. Η εμφάνιση του μηνύματος του παραλήπτη γίνεται σε ένα modal παράθυρο με το όνομα του αποστολέα αλλά και το περιεχόμενο του μηνύματος. Ένα σενάριο που θα πραγματοποιούταν με επιτυχία είναι μεταξύ 3 ενεργών χρηστών (Εικόνα 41), όταν ο τοπικός χρήστης πυροδοτήσει το γεγονός με αντικείμενο που περιέχει όνομα παραλήπτη , όνομα αποστολέα και τα μοναδικά τους αναγνωριστικά θα έχει ως συνέπεια να αλλάξει την διεπαφή των δυο συνεργατών και στην συνέχεια να ενημερωθούν οι συνεργάτες για το γεγονός αυτό (δηλαδή για την αποστολή και το περιεχόμενο του μηνύματος αντίστοιχα) , προφανώς ο 3^{ος} συνεργάτης δεν θα ενημερωθεί για το γεγονός αυτό που προέκυψε , δεν θα αλλάξει δηλαδή η διεπαφή του.



Εικόνα 40 Scenario with one active user



Εικόνα 41 Scenario with 3 active users

5.4 Σενάρια χρήσης υπηρεσιών που δημιουργήθηκαν στην εφαρμογή

Κρίνεται απαραίτητο σε αυτό το σημείο να αναφέρουμε μερικά ενδεικτικά παραδείγματα χρήσης της εφαρμογής ώστε να διατυπωθούν με μεγαλύτερη σαφήνεια οι λειτουργίες που δημιουργήθηκαν. Ένα σενάριο που θα μπορούσε να πραγματοποιηθεί γρήγορα και με αξιοπιστία χρησιμοποιώντας την εφαρμογή είναι για 2 συνεργαζόμενα καταστήματα που πραγματοποιούν διανομή τροφίμων στο σπίτι του εκάστοτε πελάτη. Έστω ότι το κάθε κατάστημα αποτελείτε από ένα board τα τρόφιμα που διαθέτει η αποθήκη παρουσιάζονται σε λίστες από κάρτες (καταγράφει το όνομα του προϊόντος και την τιμή) , ο διαχειριστής του board είναι ο υπάλληλος που εξυπηρετεί τους πελάτες-μέλη του board. Για να γίνει η παραγγελία θα πρέπει ο πελάτης να δώσει , την διεύθυνση και τι τρόφιμα επιθυμεί να παραλάβει (ονομαστικά) . Αυτός ο διάλογος μεταξύ του υπαλλήλου και πελάτη θα πρέπει να παραμείνει κρυφός προς τους υπόλοιπους πελάτες για λόγους ασφαλείας (διότι πρέπει να γίνει γνωστή η διεύθυνση του χρήστη) όπου η λειτουργία ανταλλαγής μηνυμάτων της εφαρμογής είναι ιδανική και έχει ως συνέπεια να εξυπηρετούνται οι

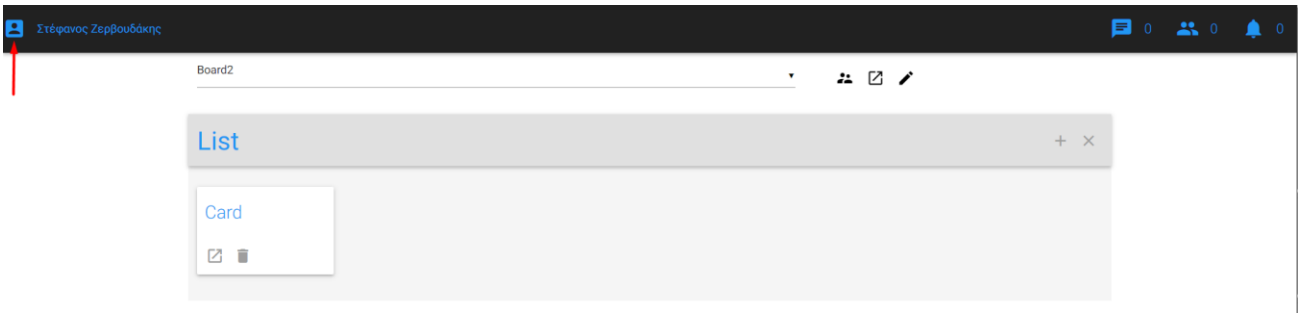
πελάτες γρήγορα και σε πραγματικό χρόνο. Κάτι παρόμοιο εάν χρησιμοποιούσαν οι χρήστες μόνο το Trello θα έπρεπε να εκτελούσαν παραπάνω ενέργειες (προσθήκη ενδιαφερόμενου σε διαφορετικό board κατόπιν συνεννόησης ώστε να παραμείνει κρυφή η συζήτηση) και αυτό θα κόστιζε σε χρόνο με αποτέλεσμα λιγότεροι πελάτες να εξυπηρετούνται στον ίδιο χρόνο. Επίσης θα μπορούσε να πραγματοποιηθεί επικοινωνία μεταξύ των διαχειριστών των board αφού τα δύο καταστήματα αναφέραμε ότι συνεργάζονται , ώστε να παρουσιάσουν νέα προϊόντα στις λίστες από κάρτες. Σε γενικές γραμμές , board που δημιουργούνται για να εξυπηρετήσουν πελάτες (μέλη του board) ιδανική λειτουργία είναι η παρουσίαση των συνδεδεμένων χρηστών στην εφαρμογή , όπου θα μπορούσαμε να υποθέσουμε ότι το πλήθος των συνδεδεμένων χρηστών είναι η ουρά εξυπηρέτησης των πελατών (αλλά δεν συμβαίνει κατά ανάγκη αυτό). Επίσης εάν κάποιος χρήστης επιθυμεί συνεργασία σε πραγματικό χρόνο χρησιμοποιώντας την συγκεκριμένη λειτουργία μπορεί να δει ποιοι είναι διαθέσιμοι συνεργάτες εκείνη την στιγμή για να πραγματοποιηθεί το γεγονός αυτό , κάτι τέτοιο δεν μπορεί να επιτευχθεί χρησιμοποιώντας το Trello αποκλειστικά.

Συνοψίζοντας , η αξιοποίηση του Realtime API στην εφαρμογή έχει σαν πρωταρχικό στόχο την γρηγορότερη εξυπηρέτηση των χρηστών αλλά και την επίτευξη γεγονότων χωρίς πολύ πολυπλοκότητα όσον αφορά της ενέργειες που θα πρέπει να καταβάλει ο πελάτης ώστε να φέρει εις πέρας διάφορα σενάρια χρήσης.

6 Συνδυασμός των δυο API's στην εφαρμογή

Ο συνδυασμός των δυο APIs έχει σαν στόχο την δημιουργία νέων δυνατοτήτων ενημέρωσης για γεγονότα που πραγματοποιούν χρήστες της εφαρμογής . Οι κλήσεις συστήματος προς το Trello είναι αυτοματοποιημένες , δηλαδή επιτυγχάνονται κάθε συγκεκριμένο χρονικό διάστημα , ενώ για να ενημερωθούν οι συνεργάτες υπεύθυνοι είναι οι ακροατές γεγονότων που διαθέτει το Realtime API . Εάν η απάντηση που επιστρέψει το Trello έχει διαφοροποιήσεις σχετικά με το JSON που επέστρεψε στην προηγούμενη κλήση τότε πυροδοτείτε ο ανάλογος ακροατής που διαθέτει το Realtime API για να ενημερώσει τους συνεργάτες για την αλλαγή που έγινε (π.χ. μια αλλαγή στο JSON αρχείο θα μπορούσε να θεωρηθεί η προσθήκη κάρτας μέσω της εφαρμογής). Σε γενικές γραμμές οι κλήσεις που γίνονται και αφορούν το Trello έχουν να κάνουν με τις λίστες από κάρτες (το πλήθος αυτών) αλλά και για τα σχόλια που υπάρχουν στο board που έχει μεταβεί ο χρήστης. Επίσης οι αυτοματοποιημένες κλήσεις συστήματος ξεκινάνε μετά από ένα συγκεκριμένο χρονικό διάστημα εφόσον έχει μεταβεί (δηλ. το έχει επιλέξει από την λίστα επιλογής) ο χρήστης στο board που επιθυμεί. Αυτό αποτυπώνεται στην διεπαφή του χρήστη αλλάζοντας το χρώμα ενός εικονιδίου (από κόκκινο σε μπλε βλέπε Εικόνα 42 Εικόνα 43) ενημερώνοντας με αυτόν τον τρόπο τον χρήστη ότι εάν πραγματοποιήσει κάποιο event θα το μάθουν και οι συνδεδεμένοι συνεργάτες της εφαρμογής.

Εικόνα 42 Χρήστης που μόλις επέλεξε το board



Εικόνα 43 Χρήστης που βρίσκεται σε board μετά από ένα συγκεκριμένο χρονικό διάστημα

Ο σκοπός της λειτουργίας αυτής είναι να μην γίνονται άσκοπα κλήσεις συστήματος σε σενάρια όπου χρήστες μεταβαίνουν από το ένα board στο άλλο σε πολύ σύντομο χρονικό διάστημα. Τέλος , ως γραφική απεικόνιση στην διεπαφή του χρήστη η λειτουργία για τις ενημερώσεις είναι μια ετικέτα με έναν μετρητή που αυξάνετε όταν ένα γεγονός πραγματοποιηθεί από συνδεδεμένους συνεργάτες της εφαρμογής .

6.1 Ενημέρωση συνεργατών για προσθήκη ή διαγραφή λίστας

Η υπηρεσία αυτή έχει να κάνει με την ενημέρωση των συνεργατών όταν προστεθεί ή αφαιρεθεί κάποια λίστα. Συνεργάτες στην προκείμενη περίπτωση είναι τα μέλη του εκάστοτε board. Ουσιαστικά ο χρήστης μπορεί να ενημερωθεί σε πραγματικό χρόνο για γεγονότα (είτε προσθήκη είτε διαγραφή λίστας) που συμβαίνουν σε board στα οποία ανήκει (δηλ. είναι μέλος σε αυτά) ενώ έχει επιλέξει διαφορετικό board εκείνη την στιγμή. Όταν ο πελάτης επιλέξει ένα πίνακα από την λίστα επιλογής τότε θα γίνονται κλήσεις συστήματος προς το Trello ανά καθορισμένο χρονικό διάστημα. Η συνάρτηση που είναι υπεύθυνη για την λειτουργία αυτή είναι της μορφής setInterval(function,milliseconds). Όσον αφορά τους παραμέτρους που παίρνει , η $1^{η}$ έχει να κάνει με την συνάρτηση που θα καλείτε (όπου αυτή η συνάρτηση θα περιέχει την κλήση τύπου get και τις εντολές που εξετάζουν εάν υπάρχει αλλαγή στο μέγεθος του JSON) , η $2^{η}$ παράμετρος καθορίζει πόσο συχνά θα εκτελείτε η συνάρτηση (σε ms) που δόθηκε στο πρώτο όρισμα. Η απάντηση που επιστρέφει το Trello είναι ένα JSON με όλες τις λίστες από κάρτες του board που έχει επιλέξει ο χρήστης από την λίστα επιλογής (option list) και τα οποία αποθηκεύονται στα εξής array :

- Πίνακας για τα ονόματα των λιστών
- Πίνακας για τα ονόματα των καρτών

Κάθε φορά η συνάρτηση ελέγχει εάν το μέγεθος των arrays έχει αλλάξει σύμφωνα με την προηγούμενη κλήση που έγινε. Πιο συγκεκριμένα για την διαγραφή της λίστας το σύστημα έχει την δυνατότητα να διακρίνει εάν ο χρήστης διέγραψε μια άδεια λίστα (λίστα χωρίς κάρτες) ή εάν διέγραψε μια λίστα από κάρτες. Για την ενημέρωση των συνεργατών –της εφαρμογής- που είναι μέλη στο board που πραγματοποιήθηκε το γεγονός υπεύθυνο είναι το Realtime API . Γίνεται η χρήση της συνεργατικής λίστας , όπου μέσα σε αυτή ωθείτε το μήνυμα και το όνομα του board στο οποίο έγινε η προσθήκη ή αφαίρεση λίστας. Το σύστημα διαθέτει έναν ακροατή γεγονότων που πυροδοτείται κάθε φορά που προστίθεται ένα στοιχείο στην συνεργατική λίστα για να αλλάξει την διεπαφή των συνεργατών. Οι εντολές που εκτελούνται μέσα στον ακροατή γεγονότων έχουν να κάνουν με τους μη-τοπικούς χρήστες , δηλαδή με τους χρήστες που είναι μέλη στο board αλλά δεν είναι αυτοί που πυροδότησαν τον ακροατή. Στους μη-τοπικούς χρήστες γίνεται κλήση συστήματος προς το Trello για να ελέγξει εάν ο κάθε μη-τοπικός χρήστης είναι μέλος στο board στο οποίο έγινε το event , σε όποιον επιστρέψει θετικό αποτέλεσμα θα του εμφανιστεί στην διεπαφή το μήνυμα που αφορά την αλλαγή που έγινε στο πλήθος των λιστών. Παρακάτω παρουσιάζεται ο αλγόριθμος που ελέγχει εάν προστέθηκε ή αφαιρέθηκε μια λίστα:

```
Algorithm checkLengthOfList{
    if nameOfList.length > bef_lengthList and nameOfCards.length
==bef_lengthCard
   then
       Update Collaborators for adding a new list
else if nameOfList.length < bef_lengthList and nameOfCards.length = =
bef_lengthCard
   then
       Update Collaborators for deleted an empty list
else if nameOfList.length < bef_lengthList and nameOfCards.length =
bef_lengthCard
   then
       Update Collaborators for deleted a list of cards
}
```

6.2 Ενημέρωση συνεργατών για προσθήκη ή αφαίρεση κάρτας

Όπως αναφέρθηκε και προηγουμένως στην αυτοματοποιημένη κλήση η απάντηση που επιστρέφει το Trello είναι τα ονόματα των λιστών και τα ονόματα των καρτών του board που έχει επιλέξει ο

χρήστης από την λίστα επιλογής. Ουσιαστικά και σε αυτήν την λειτουργία συγκρίνουμε κάθε φορά το μέγεθος των JavaScript arrays σύμφωνα με την προηγούμενη κλήση που είχε πραγματοποιηθεί. Το σύστημα γνωρίζει ότι έχει αφαιρεθεί ή προστεθεί μια κάρτα εάν το πλήθος των λιστών έχει παραμείνει το ίδιο και έχει αλλάξει το μέγεθος του array όπου αποθηκεύονται οι κάρτες. Σε αντίστοιχη περίπτωση δημιουργείτε το ανάλογο μήνυμα που θα σταλθεί στους συνεργάτες , όπου από εκεί και πέρα αξιοποιείται η συνεργατική λίστα ώστε να ωθηθεί σε αυτήν το όνομα του board που έγινε η αλλαγή στις κάρτες , το μήνυμα ενημέρωσης όπως και το όνομα του χρήστη που πραγματοποίησε το event αυτό. Ο ακροατής γεγονότων πυροδοτείτε για να ενημερώσει τους μη-τοπικούς χρήστες που είναι μέλη στο board για το γεγονός αυτό. Στους μη-τοπικούς χρήστες γίνεται κλήση συστήματος προς το Trello για να ελέγξει εάν ο κάθε μη-τοπικός χρήστης είναι μέλος στο board στο οποίο έγινε το event , σε όποιον επιστρέψει θετικό αποτέλεσμα θα του εμφανιστεί στην διεπαφή το μήνυμα που αφορά την αλλαγή που έγινε στο πλήθος των καρτών. Ουσιαστικά οι μη-τοπικοί χρήστες διαχωρίζονται σε εκείνους που είναι μέλη στο board που έγινε το event και σε αυτούς που δεν είναι. Αυτό επιτυγχάνεται για να καθοριστεί μέσω του ακροατή να αλλάξει η διεπαφή πολύ συγκεκριμένων συνεργατών και όχι όλων (πως επιτυγχάνεται αυτό προγραμματιστικά διακρίνεται στην Εικόνα 44) .

```
if(!isLocal){ //εαν είναι μη-τοπικός ο χρήστης για το event τότε η συνθηκη είναι αληθής

  Trello.get('/members/me/boards/',function(boards){  //κληση συστηματος προς το trello  για να ελεγχθει εαν ο μη-τοπικος χρηστης ανήκει στο board που έγινε το γεγονος
    $.each(boards, function (index,value){
            listaMeOnomata.push(value.name);
    });
    var temp=[];
    var haveBoard;
    temp.push(collaborativeListCard.get(0));
   for(var i=0;i<listaMeOnomata.length;i++){
     if(listaMeOnomata[i]==temp[0]){
       haveBoard=true;  //γινεται αληθης η μεταβλητη εάν ανηκει στο συγκεκριμενο board ο χρήστης
       break;
     }
   }
   if(boardNameNonLocalUser!=collaborativeListCard.get(0) && haveBoard==true){//enhmerwsh gia ton non-local user o opoios exei to board sto opoio egine h pos8aferesh kartas
      var date = new Date();
      var timeOfEvent = date.toLocaleTimeString();
      notificationsBox+=collaborativeListCard.get(0)+": "+collaborativeListCard.get(1)+" | "+timeOfEvent.bold()+" | "+"<br>";
      document.getElementById("notifCollsMessage").innerHTML = notificationsBox;
      counterOfNotific++;  //o μετρητης ειδοποιησεων αυξανετε κάθε φορά
      document.getElementById("NumNot").innerHTML=counterOfNotific;
      //αλλαγη χρωματος της ετικέτας που αναγράφετε ο μετρητης ειδοποιησεων
      $('#ChangeCollor').removeClass('blue-text text-lighten 2');
      $('#ChangeCollor').addClass('red-text text-lighten 2');
  }
```

Εικόνα 44 Εντολές για την ενημέρωση των συνεργατών

Αυτή η δυνατότητα ενημέρωσης , δηλαδή να υπάρχει real time ειδοποίηση για τους συνεργάτες που έχουν επιλέξει διαφορετικό board , δεν θα ήταν εφικτό να γίνει χωρίς την βοήθεια του Real time API , στην πλατφόρμα για να δει ο χρήστης τέτοιων ειδών ενημερώσεων θα πρέπει να μεταβεί στο board που έγινε η προσθήκη της κάρτας ή της λίστας.

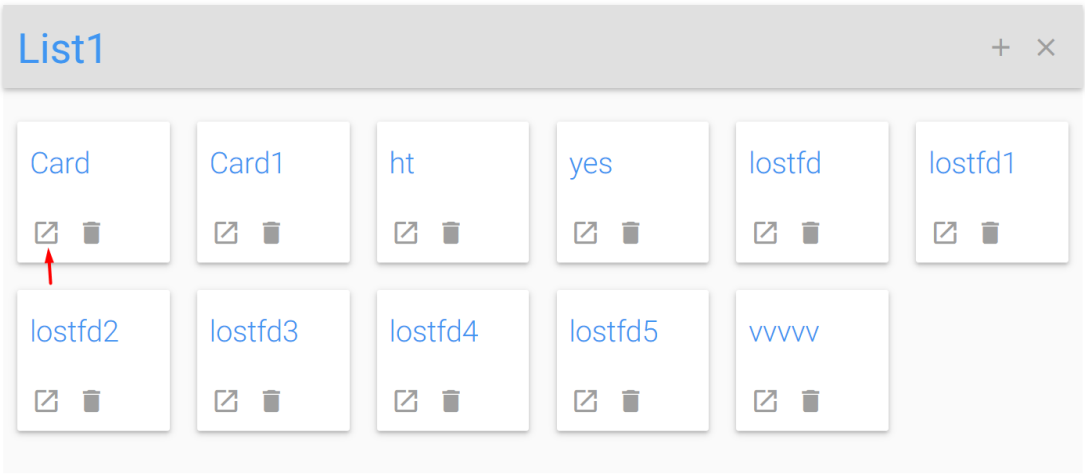
Σε αυτό το σημείο κρίνεται αναγκαίο να επισημανθεί ένα σενάριο χρήσης της εφαρμογής ώστε να γίνει πιο κατανοητή η υπηρεσία ενημέρωσης. Έστω για παράδειγμα σε ένα board οι συνεργάτες έχουν όλα τα project που θα πρέπει να ολοκληρώσουν για τους επόμενους μήνες (η κάθε λίστα αποτυπώνει και ένα project) . Για την δημοσίευση νέου project αρκεί ένας από τους συνεργάτες να δημιουργήσει μια νέα λίστα –μέσω της εφαρμογής- (με το όνομα που θα έχει το νέο project). Όταν ολοκληρωθεί ένα project , αρκεί ένας από τους συνδεδεμένους συνεργάτες της εφαρμογής να διαγράψει την λίστα. Θα μπορούσε χρήστης αξιοποιώντας τις υπηρεσίες (6.1 και 6.2) που διαθέτει η εφαρμογή να λαμβάνει ειδοποιήσεις (ανεξαρτήτως σε πιο board βρίσκεται εκείνη την στιγμή) για νέα project που μπορεί να πρόσθεσε κάποιος συνεργάτης –μέσω της εφαρμογής- , αρκεί να είναι μέλος σε αυτό χωρίς να χρειάζεται να μεταβαίνει κάθε φορά στο board για να δει εάν προστέθηκε κάποια λίστα που αντιστοιχεί σε νέο project.

6.3 Ενημέρωση συνεργατών για προσθήκη σχόλιου μέσω της εφαρμογής

Χρήστης έχει την δυνατότητα μέσα από την εφαρμογή να αναρτήσει σχόλιο ακολουθώντας τα παρακάτω βήματα-καθήκοντα.

- Βήμα 1^{ο} : Ο χρήστης θα πρέπει να πατήσει το διαδραστικό κουμπί που διαθέτει η κάρτα (βλέπε Εικόνα 45).
- Βήμα 2^{ο} : Θα εμφανιστεί σε νέα καρτέλα (redirect) η κάρτα αυτή στο Trello ώστε να κάνει την ανάρτηση σχολιασμού ο χρήστης.

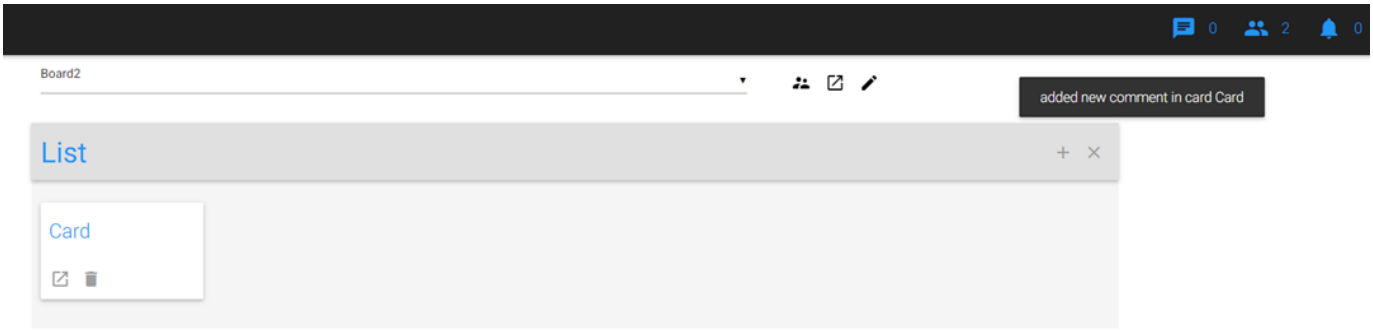
Εάν πραγματοποιήσει τις ενέργειες αυτές τότε όλοι οι συνδεδεμένοι συνεργάτες της εφαρμογής που είναι μέλη στο board που έγινε το event θα ενημερωθούν με το ανάλογο μήνυμα.



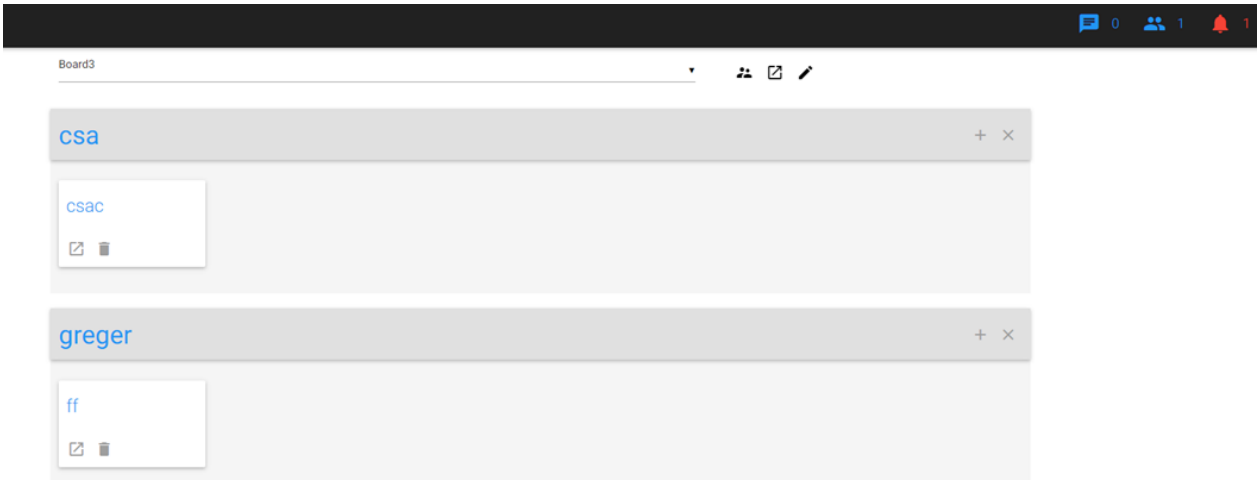
Εικόνα 45 Κουμπί για απευθείας μετάβαση(redirect) στην κάρτα στο Trello

Χρήστες που χρησιμοποιούν μόνο το Trello για να ενημερωθούν για νέα σχόλια θα πρέπει να μεταβούν στο board που έγινε ώστε να δουν στο ιστορικό δραστηριοτήτων το σχόλιο και το όνομα αυτού που το ανέβασε. Χρήστες που χρησιμοποιούν την εφαρμογή που δημιουργήθηκε θα έχουν την δυνατότητα να ενημερώνονται σε πραγματικό χρόνο για ότι σχόλιο έγινε σε board στα οποία είναι μέλη χωρίς να κρίνεται απαραίτητο να μεταβούν στο board εκείνο ώστε να δουν το νέο σχόλιο. Τεχνικά επιτυγχάνεται με τον ίδιο τρόπο που γίνονται και οι δυο προηγούμενες ενημερώσεις συνεργατών (6.1 και 6.2) . Δηλαδή όταν ένας χρήστης επιλέξει από την λίστα επιλογής ένα board τότε θα πραγματοποιούνται αυτοματοποιημένες (ανά συγκεκριμένο χρονικό διάστημα) κλήσεις συστήματος στο Trello API. Η απόκριση που θα παίρνει (JSON format) θα αποθηκεύετε σε JavaScript array. Εάν αλλάξει το μέγεθος του array , τότε θα ενημερωθούν οι συνεργάτες για την προσθήκη σχόλιου. Για αυτήν την ιδιότητα υπεύθυνο είναι το Realtime API. Μέσα στην συνεργατική λίστα ωθείται ένας πίνακας με 4 στοιχεία. Το όνομα του χρήστη που έκανε την προσθήκη του σχόλιου , το σχόλιο που αναρτήθηκε , το όνομα της κάρτας που αναρτήθηκε το σχόλιο και το όνομα του board που έγινε το event αυτό. Όταν στην συνεργατική λίστα ωθηθεί ο πίνακας , τότε θα πυροδοτηθεί ο ακροατής γεγονότων που έχει δημιουργηθεί για να ενημερώσει τους συνεργάτες που ανήκουν στο board που έγινε το σχόλιο ανεξαρτήτως σε πιο board βρίσκονται εκείνη την στιγμή. Μέσα στον ακροατή γεγονότων υπάρχουν οι ανάλογες εντολές που ελέγχουν εάν ο συνεργάτης είναι μέλος στο board ή όχι , όπου αυτό πραγματοποιείται με την κλήση που γίνεται προς το Trello.

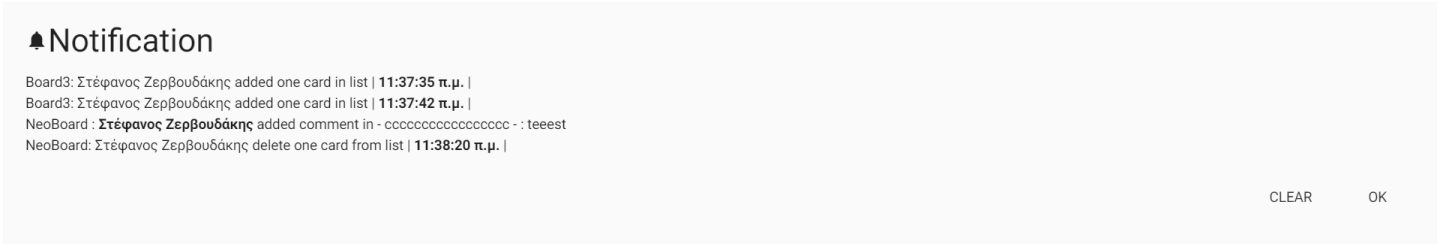
Οι διεπαφές των συνεργατών είναι διαφορετικές (βλέπε Εικόνα 46 και Εικόνα 47). Στο χρήστη ο οποίος είναι εκείνη την στιγμή στο board που έγινε η ανάρτηση θα εμφανιστεί στην οθόνη ένα μήνυμα που θα τον ενημερώνει ότι έγινε σχόλιο και το όνομα της κάρτας. Αντίθετα, σε χρήστη που δεν βρίσκεται εκείνη την στιγμή στο board που έγινε το γεγονός θα αυξηθεί ο μετρητής που υπάρχει στην ετικέτα ειδοποιήσεων. Πατώντας πάνω σ' αυτή, θα εμφανιστεί το μήνυμα σύμφωνα με τα στοιχεία που περιέχει η συνεργατική λίστα.



Εικόνα 46 Διεπαφή χρήστη που βρίσκεται στο board που έγινε το event



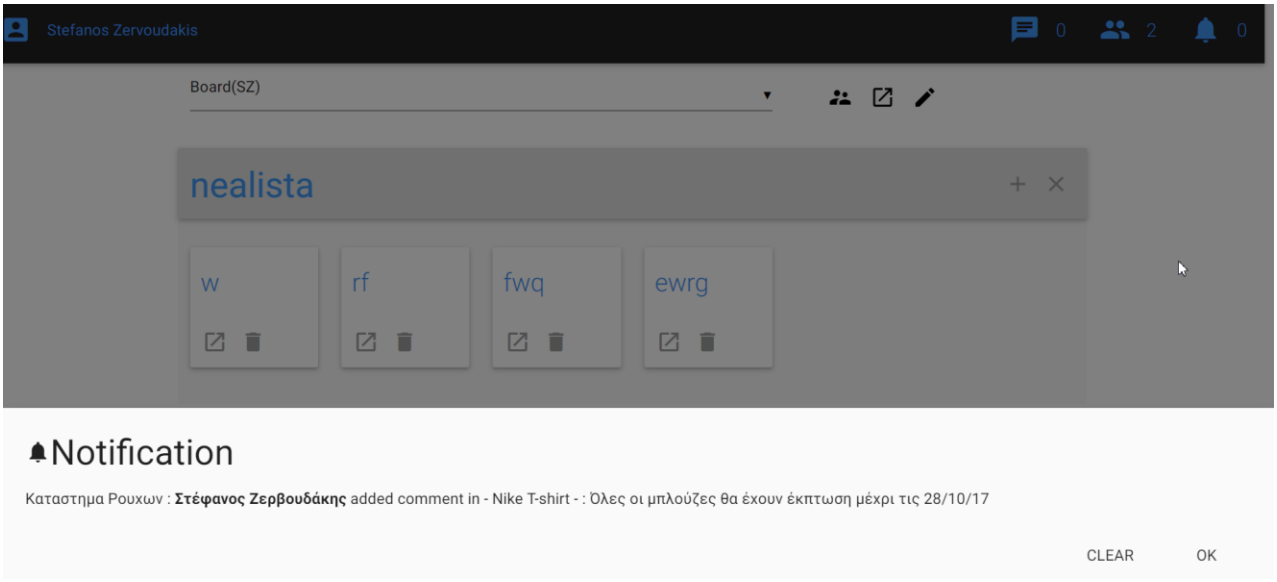
Εικόνα 47 Διεπαφή χρήστη που βρίσκεται σε διαφορετικό board από αυτό που έγινε το event



Εικόνα 48 Notification box της εφαρμογής

Το κουτί που αναγράφονται οι δραστηριότητες των συνεργατών (βλέπε Εικόνα 48) , εκτός από τα στοιχεία που έχει η συνεργατική λίστα , αναγράφει και την ώρα του κάθε γεγονός που πραγματοποιήθηκε όπως και την δυνατότητα να καθαρίσει (clear button) , ο χρήστης το κουτί ενημέρωσης.

Ο συνδυασμός των δυο APIs έχει θετικά αποτελέσματα σε αρκετά σενάρια. Έστω ένας χρήστης είναι μέλος σε board που αντιστοιχούν σε διαφόρων ειδών καταστημάτων. Ένα board διαθέτει προσφορές για κατάστημα ρούχων όπου κάθε λίστα αποτυπώνει το είδος των ρούχων (π.χ. φόρμες, μπλούζες, παπούτσια) που υπάρχουν στο κατάστημα και η κάθε κάρτα αποτυπώνει την μάρκα του εκάστοτε προϊόντος όπως και την τιμή. Η ενημέρωση των συνεργατών/πελατών για νέες προσφορές γίνεται μέσω σχολιασμού στην κάρτα (το προϊόν δηλαδή) όπου αναγράφει το ποσοστό που έχει η προσφορά. Ένα άλλο board αποτυπώνει όλες τις θεατρικές παραστάσεις που θα πραγματοποιηθούν κάθε μήνα όπου για την ενημέρωση των πελατών για νέα παράσταση γίνεται στην κάρτα με σχόλιο (π.χ. 'νέα θεατρική παράσταση –Τίτλος Παράστασης- 20 ευρώ/άτομο). Ο χρήστης θα λαμβάνει ειδοποιήσεις και θα γνωρίζει για της νέες προσφορές και τις νέες θεατρικές παραστάσεις χωρίς να πρέπει να μεταβεί στο εκάστοτε board κάθε φορά. Αυτό βοηθάει χρήστες οι οποίοι επιθυμούν να δουλεύουν σε ένα board και να ενημερώνονται ταυτόχρονα για γεγονότα που συμβαίνουν σε άλλα board στα οποία είναι μέλη. Στην Εικόνα 49 ένα σενάριο όπου στο κουτί ειδοποιήσεων του χρήστη επισημάνθηκε νέα προσφορά σε προϊόν.



Εικόνα 49 Ειδοποίηση χρήστη για έκπτωση προϊόντος

7 Συμπέρασμα και μελλοντικές επεκτάσεις

Η παρούσα πτυχιακή εργασία μελέτησε σύγχρονα APIs και τους τρόπους που μπορούν να αξιοποιηθούν για να υποστηρίξουν την διαλειτουργικότητα μεταξύ εφαρμογών και υπηρεσιών. Για το λόγο αυτό , αναπτύχθηκε μια διαδικτυακή εφαρμογή η οποία παρέχει νέες υπηρεσίες σε χρήστες που χρησιμοποιούν μια ήδη υπάρχουσα πλατφόρμα. Οι υπηρεσίες αυτές έχουν να κάνουν με την ενημέρωση και την επικοινωνία μεταξύ μιας ομάδας χρηστών που αποκαλούνται αλλιώς και συνεργάτες. Για την υλοποίηση των υπηρεσιών αυτών αξιοποιήθηκαν δυο API's , αυτό της πλατφόρμας Trello και το Realtime API όπου με τεκμηριωμένες ενέργειες αποδείχθηκε η παροχή νέων υπηρεσιών που δεν υπήρχαν προηγουμένως στην βασική έκδοση της πλατφόρμας.

Παρότι η τελική εφαρμογή εξυπηρέτησε απολύτως το σκοπό της και απέδειξε τις νέες δυνατότητες που υποστηρίζονται , ανέδειξε επίσης και μια σειρά ζητημάτων που θα μπορούσαν να μελετηθούν περαιτέρω προκειμένου να βελτιωθεί η ευχρηστία αλλά και οι δυνατότητες μιας τέτοιας εφαρμογής. Ένα πρώτο ζήτημα που θα μπορούσε να εξεταστεί στο μέλλον είναι η αποτύπωση των ονομάτων των διαχειριστών όλων των board που είναι μέλος ο χρήστης , με τέτοιο τρόπο ώστε να παρουσιάζονται στην αρχική σελίδα. Σαν δεύτερο ζήτημα που θα μπορούσε να εξεταστεί είναι μια νέα δυνατότητα επικοινωνίας μεταξύ χρηστών. Πιο συγκεκριμένα η επικοινωνία αυτή θα μπορούσε να ήταν τύπου Skype call.Ο κάθε χρήστης θα είχε την δυνατότητα να καλεί σε κλήση συνδεδεμένο χρήστη/συνεργάτη για την πραγματοποίηση συνομιλίας/διαλόγου (π.χ. για να ολοκληρώσουν ένα έργο που τους έχει ανατεθεί) . Επιπλέον , δεν θα κρινόταν απαραίτητο οι χρήστες αυτοί να έχουν κάποιο κοινό board μεταξύ τους ώστε να αξιοποιηθεί αυτή η υπηρεσία. Τέλος, τέτοιου είδους λειτουργία θα μπορούσε να υλοποιηθεί με εύκολες προγραμματιστικές τεχνικές, διότι το Realtime API παρέχει τα απαραίτητα εργαλεία όπως συνεργατικά αντικείμενα και ακροατές γεγονότων .

Συνοψίζοντας λοιπόν , αυτά που δημιουργηθήκαν έκαναν ευκολότερη την ζωή του χρήστη που χρησιμοποιεί το Trello, με επιπλέον υπηρεσίες χωρίς πολυπλοκότητα και με αξιοπιστία. Την ίδια τακτική θα ήταν σωστό να ακολουθήσει όποιος κατασκευαστής τον ενδιέφερε να αξιοποιήσει τα δύο API's που μελετήθηκαν ώστε να δημιουργήσει νέες υπηρεσίες που δεν υπήρχαν προηγουμένως στην πλατφόρμα.

8 Βιβλιογραφία

[1] Peter Johnson-Lenz, "Rhythms, Boundaries and Containers," April 1990.

[2] [Online]. https://pithos.okeanos.grnet.gr/public/7xEDw8vU20IWmbWMTUAk02

[3] Wide Web Consortium W3C web site. [Online]. https://www.w3.org/

[4] Eysenbach, "what is e-healt?".

[5] Roy Thomas Fielding, "Chapter 5: Representational State Transfer (REST)," *Architectural Styles and the Design of Network-based Software Architectures (Ph.D.)*.

[6] "Web Services Architecture," 2004.

[7] "Introducing Google Drive… yes, really," 2014.

[8] Walter S Mossberg, "Google Stores, Syncs, Edits in the Cloud," *The Wall Street Journal*, 2004.

[9] Pryor M, "Trello is Being Acquired By Atlassian," 2017.

[10] [Online]. https://reviews.financesonline.com/p/trello/#overview-benefits

[11] "A Special Announcement: Trello is now part of Trello," 2014.

[12] Daniel R, "Trello Dojo," 2014.

[13] [Online]. http://popaganda.gr/valte-programma-stis-ergasies-sas-trello/

[14] [Online]. http://gr.pcmag.com/how-to/22047/help/eukole-organose-kai-online-sunergasia-me-to-trello

[15] [Online]. https://developers.google.com/google-apps/realtime/fundamentals

[16] [Online]. https://developers.google.com/google-apps/realtime/build-model

[17] [Online]. https://developers.trello.com/v1.0/reference#actions

[18] Μπελερ, "Εφαρμογές των τεχνολογιών της πληροφορικής και των τηλεπικοινωνιών στην επεξεργασία και τη μετάδοση βιολογικών σημάτων με έμφαση στην τηλεϊατρική," 2009.

[19] [Online]. https://developers.trello.com/v1.0/reference#actions

[20] [Online]. https://www.captiga.com/tutorials/install-additional-word-reference-styles-mac-windows/