Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Σχεδιασμός και Υλοποίηση Κινητής Εφαρμογής Πληθοπορισμού για Αξιολόγηση και Ανασκόπηση Τόπων Πολιτιστικού Περιεχομένου σε Διεπαφή Χάρτη (React Native App Development)

Διπλωματικη Εργασια

του

ΕΥΘΥΜΙΟΥ ΠΕΓΚΑ

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Εργαστηριο Ευφυων Συστηματων, Περιεχομενου και Αλληλεπιδρασης

Αθήνα, Ιούνιος 2019

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Ευφυών Συστημάτων, Περιεχομένου και Αλληλεπίδρασης

Σχεδιασμός και Υλοποίηση Κινητής Εφαρμογής Πληθοπορισμού για Αξιολόγηση και Ανασκόπηση Τόπων Πολιτιστικού Περιεχομένου σε Διεπαφή Χάρτη (React Native App Development)

Διπλωματικη Εργασια

του

ΕΥΘΥΜΙΟΥ ΠΕΓΚΑ

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 12η Ιουνίου 2019.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Στάμου	Ανδρέας Γ. Σταφυλοπάτης	Νικόλαος Παπασπύρου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2019

(Υπογραφή)

.....

Ευθυμιος Πεγκας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Ευφυών Συστημάτων, Περιεχομένου και Αλληλεπίδρασης

Ευχαριστίες

Το παρόν έργο διεξήχθη στο Εργαστήριο Ευφυών Συστημάτων, Περιεχομένου και Αλληλεπίδρασης του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ του Εθνικού Μετσόβιου Πολυτεχνείου, υπό την επίβλεψη και καθοδήγηση του καθηγητή Δρ. Γεωργίου Στάμου. Η διπλωματική αυτή θεμελιώνεται στις κυριότερες αρχές που διέπουν την Ανάπτυξη Διαδικτυακών Εφαρμογών, αλλά ταυτοχρόνως διερευνά την ομορφιά και το μεγαλείο των πιο σύγχρονων διαδικτυακών τεχνολογιών του σήμερα.

Θα ήθελα να ευχαριστήσω τους καθηγητές και επιβλέποντές μου Δρ. Γ. Στάμου και Δρ. Β. Τζουβάρα για την ώθηση που μου έδωσαν ώστε να εμπνευστώ την κεντρική ιδέα του παρόντος έργου και την εμπιστοσύνη που μου έδειξαν στην πρωτοβουλία μου αυτή. Ένα μεγάλο ευχαριστώ αξίζει και σε όλους τους υπόλοιπους καθηγητές μου, για την διδασκαλία τους και τις γνώσεις που μου μετέδωσαν κατά τη διάρκεια των σπουδών μου.

Ευχαριστώ ιδιαιτέρως την μεταπτυχιακό Μαρία Ράλλη για την στήριξή της από την αρχή αυτής της προσπάθειας, την ενθαρρυντική στάση της, τις συβουλές της και τις γνώσεις της. Η καθοδήγησή της και η καλή διάθεση που μου έδειξε με βοήθησαν σημαντικά στην ολοκλήρωση αυτού του έργου.

Τέλος, αφιερώνω το παρόν έργο στην οικογένειά μου, για την αμέριστη στήριξη και εμπιστοσύνη που μου έδειξαν στις επιλογές και στις σπουδές μου. Ευχαριστώ τον αδερφό και την αδερφή μου που συνεισέφεραν με το δικό τους τρόπο ο καθένας στην προσπάθειά μου αυτή, καθώς επίσης και τους γονείς μου, οι οποίοι μου έμαθαν να ακολουθώ πάντοτε τα όνειρά μου και να επιστρατεύομαι τη γνώση ως σύμμαχο σε όλα μου τα προβλήματα. Τίποτε δεν είναι πιο πολύτιμο από τη γνώση. Σας είμαι ευγνώμων για τον άνθρωπο στον οποίο έχω εξελιχθεί σήμερα.

Περίληψη

Τα προβλήματα επίτευξης της επικοινωνίας σήμερα έχουν ελλατωθεί σε μεγάλο βαθμό. Χάρη στην πληθώρα των μέσων κοινωνικής δικτύωσης, μπορεί κανείς να αναζητήσει, να συνομιλήσει ή να γνωρίσει άλλα άτομα άμεσα. Ωστόσο, η απότομη αυτή στροφή στην διαπροσωπική επικοινωνία έχει διεισδύσει υπερβολικά στην προσωπική ζωή δημιουργώντας νέα εμπόδια και κινδύνους που περισσότερο δυσχεραίνουν, παρά καλύπτουν τις ανάγκες των χρηστών για επικοινωνία. Η παρούσα διπλωματική εργασία διερευνά τρόπους αντιμετώπισης του παραπάνω φαινομένου, μέσα από την ανάπτυξη μιας καινοτόμου εφαρμογής που στηρίζεται σε πληθοποριστικές πρακτικές. Η εφαρμογή θα εισάγει τους χρήστες της σε μια νέα εμπειρία, η οποία σκοπό έχει την ανάδειξη των δυνατοτήτων των χρηστών της, καθώς επίσης και την προβολή των σημερινών πολιτισμικών γεγονότων. Αυτό θα είναι εφικτό αξιοποιώντας την γνώση και την εμπειρία των χρηστών εντός της εφαρμογής για την άσκηση κριτικής, τον σχολιασμό και την συζήτηση με επίκεντρο εκδηλώσεις που αφορούν την πολιτισμική κοινότητα. Με αυτό τον τρόπο, αντιστρέφονται οι ρόλοι, καθώς η εφαρμογή παύει να αποτελεί τον πομπό όπως συνηθίζεται σήμερα, αλλά γίνεται ο δέκτης της πληροφορίας. Οι χρήστες με τη σειρά τους, θα αποτελούν την πηγή της πληροφορίας, χάρη στη γνώση που θα παρέχουν μέσα από τη χρήση της εφαρμογής. Αυτό θα οδηγήσει στη διαμόρφωση μιας πιο ενεργητικής στάσης απέναντι στα πολιτισμικά και κοινωνικά δρώμενα, με αποτέλεσμα την συμμετοχή του κοινωνικού συνόλου σε περισσότερα γεγονότα και εκδηλώσεις.

Η παραπληροφόρηση και η μονόπλευρη διαφήμιση στις μέρες μας, αφήνει ελάχιστα περιθώρια στους νέους που θέλουν να εξερευνήσουν την πόλη τους, να δοκιμάσουν νέες εμπειρίες και να ξεφύγουν από τον τετριμμένο τρόπο διασκέδασης. Ακόμη, η επιλογή ενός χώρου με βάση τις προτιμήσεις, μειώνει περισσότερο τις επιλογές καθώς οι ενδιαφερόμενοι επιλέγουν περισσότερο με βάση τι είναι σίγουρο και γνωστό, και λιγότερο με βάση τις προσωπικές τους επιθυμίες. Στόχος μας είναι η ανάπτυξη ενός καινοτόμου τρόπου γεφύρωσης του χάσματος που δημιουργούν οι σημερινοί ψηφιακοί κίνδυνοι, μέσα από τη χρήση σύγχρονων τεχνολογιών και πρωτοπόρων ιδεών. Η προσοχή μας θα επικεντρωθεί στη διαμόρφωση μιας εφαρμογής, της οποίας πρωταρχικός σκοπός είναι η διευκόλυνση της διαδικασίας της επικοινωνίας και της διασκέδασης με τη χρήση ενός νέου συστήματος μηνυμάτων σε τοποθεσίες επάνω σε χάρτη. Με αυτό τον τρόπο οι χρήστες θα μπορούν να πληροφορηθούν άμεσα για το τι συμβαίνει στην περιοχή ή το χώρο ενδιαφέροντος, να αλληλεπιδράσουν σε πραγματικό χρόνο με τους ευρισκόμενους την παρούσα χρονική στιγμή εκεί, να εξερευνήσουν νέα μέρη και να απολαύσουν τη διαδικασία της οργάνωσης της διασκέδασης. Πρωτίστως όμως, στόχος της εφαρμογής είναι να

άρει τα σημερινά στερεότυπα για τις κοινωνικές ομάδες μεταξύ των νέων, φέρνοντας πιο κοντά το χρήστη με τα ενδιαφέροντά του. Να μετατρέψει σε πλειονότητες τις μειονότητες εκείνων που έχουν παραγκωνιστεί από το υπόλοιπο σύνολο λόγω των διαφορετικών αντιλήψεων και ενδιαφερόντων. Να ρίξει φως σε εκείνα τα μέρη που δεν παίρνουν την αντικειμενική αξία που τους αναλογεί. Να δώσει μάτια σε εκείνα τα άτομα που επιλέγουν να ακολουθούν τυφλά την υπόλοιπη μάζα επειδή νομίζουν πως είναι μόνοι. Κάθε άτομο, οποιασδήποτε ομάδας, ηλικίας, γένους ή καταγωγής έχει δικαίωμα στον πολιτισμό και στην ενημέρωση. Δουλειά μας είναι να το κάνουμε ευρέως γνωστό.

Λέξεις Κλειδιά

Ανάπτυξη Εφαρμογών, Κινητή Εφαρμογή, Εξυπηρετητής, Βάση Δεδομένων, Πληθοπορισμός, React, React Native, JavaScript, Google Maps, Foursquare, RESTful APIs, Expo, Redux, MERN, Full Stack.

Abstract

The problems of reaching communication today are gone. Thanks to the abundance of social media, one can search, chat or meet other people directly. However, this steep shift in interpersonal communication has penetrated too much into personal life, creating new barriers and dangers that make it harder to address users' needs for communication. This diploma thesis explores ways to address this phenomenon, through the development of an innovative application based on crowdsourcing techniques. The application will introduce its users to a new experience, which aims at highlighting the capabilities of its users, as well as the cultural events that take place today. This will be feasible by leveraging the users' knowledge and experience within the app for reviewing, commenting and discussing community-centered cultural events. In this way, the roles are reversed as the application ceases to be the transmitter of information as it is used to be today, but becomes the receiver of the information. Users, on the other hand, will be the source of this information, thanks to the knowledge they will provide through the application's interface. This will lead to a more active attitude towards cultural and social events.

Disinformation and one-sided advertising nowadays leaves little room for young people to explore their city, try new experiences and escape the trivial way of having fun. In addition, choosing a preference-based part reduces ones choices, as people choose more on the basis of what is certain and familiar, and less on the basis of their personal wishes. Our goal is to present an innovative way to bridge the gap created by today's digital dangers through the use of modern technologies and pioneering ideas. Our focus will be the developing an application, whose primary purpose is to facilitate the communication and entertainment process by using a new messaging system on map locations. In this way, users can instantly learn about what's happening in their area or at the place of interest, interact in real time with those currently present at the event, explore new places, and enjoy the process of organizing their plans for having fun. Above all, however, the main goal of this application is to lift the current stereotypes for social groups among young people, bringing the user closer to their interests. Convert to minorities those who have been crowded out of the rest due to differences based on their interests. To shed light on those places that do not get the value they deserve. Give eyes to those people who choose to blindly follow the rest of the mass, because they think they are alone. Every person, of any age, gender or background, has a right into culture and information. Our job is to make it widely known.

Keywords

App Development, Mobile Application, Server, Database, Crowdsourcing, React, React Native, JavaScript, Node, Express, Mongo, Google Maps, Foursquare, RESTful APIs, Expo, Redux, MERN, Full Stack.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Παραθέσεων

Κεφάλαιο 1

Εισαγωγή

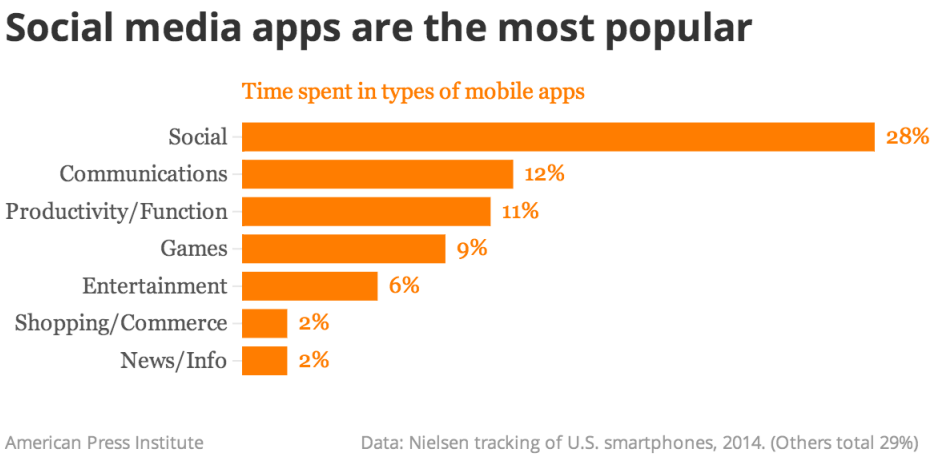
Είναι ευρεώς διαδεδομένο πως η ψηφιακή ενημέρωση βαδίζει πλέον με ραγδαίους ρυθμούς. Αμέτρητοι είναι σήμερα οι ιστότοποι και διαδικτυακές κοινότητες που σχετίζονται με όλες τις εκφάνσεις της καθημερινότητας, καθώς νέες εφαρμογές ενημέρωσης, οργάνωσης και επικοινωνίας αναδύονται στην ψηφιακή αγορά καθημερινά. Με μια αναζήτηση στο διαδίκτυο, ή με μια επισήμανση εντός της εφαρμογής, ο χρήστης μπορεί να ενημερωθεί για γεγονότα που τον ενδιαφέρουν, να δεχθεί ειδοποιήσεις για εκδηλώσεις που τον αφορούν, ή ακόμη και να προγραμματίσει το δρομολόγιό του, να υπολογίσει χρονικές και οικονομικές μεταβλητές και να σχεδιάσει το πλάνο του. Είναι λοιπόν ευνόητο το πρόβλημα που ανακύπτει από το παραπάνω φαινόμενο, σχετικά με την επικαιροποίηση και ενημέρωση της πληροφορίας που παρέχεται στο χρήστη. Παράγοντες όπως κυκλοφοριακή συμφόρηση, καιρικές αντιξοότητες, καθυστέρηση έναρξης μιας εκδήλωσης ή της άφιξης των συμμετεχόντων και άλλες αναπάντεχες εκβάσεις είναι αδύνατο να συνυπολογιστούν με κάποιον αλγόριθμο στις προαναφερθείσες εφαρμογές.

Από την άλλη, η απότομη στροφή των εφαρμογών και των μέσων κοινωνικής δικτύωσης γύρω από την ατομικότητα, έχει ως αντίκτυπο την αποδυνάμωση της συμμετοχής στο *κοινωνικό γήγνεσθαι*. Πλέον είναι προτιμότερη η απομακρυσμένη έμμεση επικοινωνία με μέσα που τροφοδοτούν το χρήστη με εγωπαθή συμπτώματα και τον απομακρύνουν από την πραγματική έννοια της επικοινωνίας. Συχνό είναι επίσης το φαινόμενο εκμετάλλευσης της κοινωνικής προβολής για λόγους επαγγελματικής ανέλιξης. Συνεπώς, γεγονότα πολιτισμικού ενδιαφέροντος περνάνε σε δεύτερη μοίρα (βλ. Σχ 1.1).

Σαν αποτέλεσμα, τόποι κοινωνικού και πολιτισμικού περιεχομένου επισκιάζονται από αυτές τις εφαρμογές '*γήγαντες*' (*facebook, instagram, snapchat, twitter, tinder* κλπ.). Εκδηλώσεις που χρήζουν προσοχής καταλήγουν να μην δέχονται την κατάλληλη προβολή. Είναι επομένως επιτακτική η ανάγκη ευαισθητοποίησης του χρήστη προκειμένου να δράσει υπερ του προσωπικού, αλλά ταυτοχρόνως και κοινωνικού οφέλους.

1.1 Αντικείμενο της διπλωματικής

Το αντικείμενο με το οποίο καταπιάνεται το παρόν έργο, επικεντρώνεται στην σχεδίαση και την υλοποίηση μιας εφαρμογής, βασισμένη στην πρακτική ενεργοποίησης ενός '*πλήθους*' ή



Σχήμα 1.1: Οι χρήστες ξοδεύουν 14 φορές περισσότερο χρόνο χρησιμοποιώντας εφαρμογές όπως το *facebook*, έναντι εφαρμογών ενημέρωσης (Πηγή: [1])

μιας ομάδας, γνωστής και ως *τακτική του πληθοπορισμού*. Η εφαρμογή που θα υλοποιηθεί θα υποστηρίζεται από iOS πλατφόρμες σε όλες τις κινητές συσκευές. Η ανάπτυξη της εφαρμογής έγινε με τη χρήση των τελευταίων τεχνολογικών εργαλείων της αγοράς και ακολουθεί τις πιο σύγχρονες τάσεις που κυριαρχούν σήμερα στον ψηφιακό κόσμο.

Πρωταρχικός στόχος της παρούσας εργασίας είναι να αναδείξει τις σύγχρονες τεχνολογίες που χρησιμοποιούνται σήμερα για την ανάπτυξη εφαρμογών, μέσα από τη σχεδίαση και υλοποίηση μιας καινοτόμου εφαρμογής. Ταυτοχρόνως, εισάγονται νέες ιδέες και τρόποι ανάδειξης τόπων πολιτισμικού περιεχομένου, μέσα από την ενεργοποίηση του κοινωνικού συνόλου. Οι χρήστες συμμετέχουν άμεσα σε πολιτισμικά δρώμενα, αναλαμβάνοντας ρόλους που έχουν άμεσο αντίκτυπο στην εικόνα του πολιτισμού. Μέσα από τη διαμόρφωση σχέσεων μεταξύ των χρηστών και της πολιτισμικής κληρονομιάς, η εφαρμογή συνδράμει τόσο στην εξέλιξη της κοινωνίας, όσο και του πολιτισμού.

1.1.1 Συνεισφορά

Η παρούσα διπλωματική εργασία σκοπό έχει να εμπλουτίσει τις δυνατότητες των χρηστών με πολιτιστικά ενδιαφέροντα. Χρησιμοποιώντας τις πιο πρόσφατες τεχνολογίες, θα προσφέρει μια ολοκληρωμένη εμπειρία μέσα από μια πλήρως λειτουργική και σύγχρονη εφαρμογή για πλατφόρμες iOS. Η εφαρμογή δίνει στους χρήστες τη δυνατότητα να ενημερώνονται σχετικά με γεγονότα και εκδηλώσεις πολιτισμικού περιεχομένου, μέσα από τεχνικές βασισμένες στον πληθοπορισμό. Πρωταρχικό μέλημα είναι η προαγωγή του πολιτισμού, αξιοποιώντας τις δυνατότητες του διαδικτύου προς όφελος των χρηστών. Η λειτουργία της εφαρμογής σέβεται και προασπίζει την ασφάλεια του χρήστη, χωρίς να εμποδίζει την αλληλεπίδρασή του με το υπόλοιπο σύνολο, μέσα από την ανταλλαγή εντυπώσεων και το σχολιασμό γεγονότων. Επίσης, οι λειτουργικότητες της εφαρμογής μπορούν να αξιοποιηθούν από πολιτισμικούς οργανισμούς και φορείς για την βελτίωση του ρόλου τους μέσα από ανατροφοδότηση των χρηστών για

μελλοντικούς σχεδιασμούς.

Είναι κοινώς παραδεκτό πως ο πολιτισμός μπορεί να λειτουργήσει ως πεδίο σεβασμού της ετερότητας. Η εφαρμογή μπορεί να διευκολύνει στο πεδίο αυτό, αναδεικνύοντας την αξία περιθωριοποιημένων ομάδων. Με τη δημιουργία μιας κοινής διεπιφάνειας χάρτη, ενοποιεί τις κοινωνικές ομάδες με κοινά ενδιαφέροντα και αναδεικνύει κοινούς τόπους. Συνδέει άτομα άγνωστα μέχρι πρότινος, άροντας όλα τα εμπόδια στην επικοινωνία και προωθώντας το διάλογο, την αμοιβαιότητα και τον σεβασμό απέναντι σε εκδηλώσεις πολιτισμικού ενδιαφέροντος. Επιπρόσθετα συμβάλλει στη δημιουργία μιας κουλτούρας του διαδυκτίου που σκοπό έχει την αναβάθμιση των χρηστών και όχι το κέρδος ή την εμπορευματοποίηση.

1.2 Οργάνωση του τόμου

Η δομή της παρούσας εργασίας είναι οργανωμένη σε έξι επί μέρους κεφάλαια:

1. Στο **2ο Κεφάλαιο** παρουσιάζονται οι σημαντικότερες έννοιες, θεωρητικές αλλά και τεχνικές, οι οποίες είναι άρρηκτα συνδεδεμένες με το έργο και θα βοηθήσουν στην διαμόρφωση μιας σφαιρικής εικόνας. Επιπλέον παρουσιάζει σχετικές προσπάθειες που έχουν γίνει στο παρελθόν για συναφή αντικείμενα και θα βοηθήσουν στην ανάπτυξη της εφαρμογής. Τέλος, γίνεται αναφορά στις βασικές τεχνολογικές έννοιες που θα αξιοποιηθούν στην εφαρμογή, όπως REST APIs, OAuth, node κλπ.

2. Στο **3ο Κεφάλαιο** αναλύονται οι λειτουργικές και μή λειτουργικές απαιτήσεις του συστήματος.Επίσης παρουσιάζεται η αρχιτεκτονική που θα ακολουθηθεί κατά την ανάπτυξη του συστήματος των διεπιφανειών του χρήστη, του εξυπηρετητή αιτημάτων και της βάσης δεδομένων.

3. Στο **4ο Κεφάλαιο** επεξηγούνται οι τεχνικές σχεδίασης των διεπιφανειών της εφαρμογής και γίνεται μια εκτεταμένη ανάλυση του τρόπου σχεδίασης της κάθε οθόνης ξεχωριτσά. Επίσης παρατίθενται η λογική σχεδίασης του εξυπηρετητή αιτημάτων και της βάσης δεδομένων.

4. Στο **5ο Κεφάλαιο** αναλύεται λεπτομερώς η υλοποίηση του συστήματος. Αρχικά, γίνεται λόγος για τις λειτουργικότητες της εφαρμογής και τις τεχνικές με τις οποίες αυτές υλοποιούνται. Στη συνέχεια, γίνεται αναφορά στο τρόπο διαχείρησης των δεδομένων εντός της εφαρμογής. Τέλος, επεξηγούνται οι λόγοι επιλογής της πλατφόρμας προγραμματισμού.

5. Στο **6ο Κεφάλαιο** γίνεται μια συνοπτική παρουσίαση του τελικού συστήματος, καθώς επίσης και μια ανάλυση των λειτουργικοτήτων που θα μπορούσαν να προστεθούν μελλοντικά στην εφαρμογή.

Κεφάλαιο 2

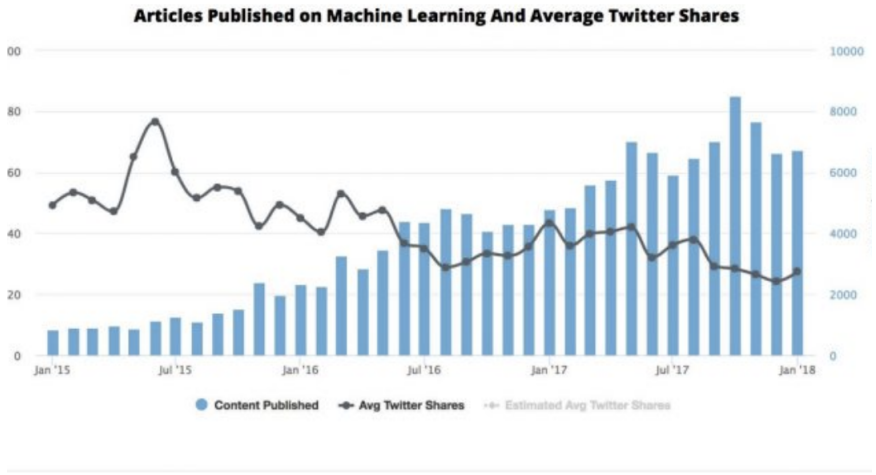
Θεωρητικό και Τεχνολογικό υπόβαθρο

Σε αυτό το κεφάλαιο θα γίνει μια εκτεταμένη ανάλυση όλων των βασικών εννοιών που θεμελιώνουν τόσο τη θεωρητική, όσο και την τεχνολογική βάση στην οποία στηρίζεται η διπλωματική εργασία. Καθώς το παρόν έργο αποτελεί συγκερασμό δύο επιστημονικών κλάδων (Κοινωνιολογία και Πληροφορική), είναι αναγκαία η διαίρεση αυτού του κεφαλαίου σε τρία μέρη. Στην ενότητα 2.1 αναλύεται το θεωρητικό μοντέλο και οι κοινωνιολογικές έννοιες που το συνιστούν. Η ενότητα 2.2 αναφέρεται σε σχετικές ερευνητικές προσπάθειες που έχουν προηγηθεί. Στο τρίτο μέρος παρατίθενται οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής (ενότητες 2.3.1-2.3.3), καθώς και άλλα σύγχρονα εργαλεία που χρησιμοποιήθηκαν (ενότητα 2.4).

2.1 Θεωρητικό Υπόβαθρο - Βασικές Έννοιες

2.1.1 Κοινωνικός Ρόλος των Σύγχρονων Εφαρμογών

Πρωταρχικό ρόλο στην επιτυχία μιας εφαρμογής παίζει το κίνητρο με το οποίο αυτή εξασφαλίζει τη διαρκή ενασχόληση του χρήστη. Βασικό στοιχείο για να επιτευχθεί αυτό είναι ο κοινωνικός ρόλος που επωμίζεται ο χρήστης εντός της εφαρμογής. Οι σημερινές εφαρμογές έχουν στρέψει την προσοχή γύρω από την κοινωνική προβολή και επιβολή, απομακρύνοντας την προσοχή από δημοσιεύσεις που αφορούν κοινωνικά δρώμενα, τεχνολογικά επιτεύγματα και γενικότερα τον συλλογικό βίο (βλ. Σχ. 2.1). Έτσι, παρατηρείται η προσκόλληση στα μέσα κοινωνικής δικτύωσης ως τρόπο άσκησης κοινωνικής επιρροής. Δημιουργούνται συνεπώς εγωκεντρικές τάσεις που τροφοδοτούν νέες ανάγκες και οδηγούν σε νέες ομοειδείς εφαρμογές. Τέτοια παραδείγματα είναι η ανάγκη για δημοτικότητα και κοινωνική αποδοχή [7] από τρίτους, η έντονη εμμονή με την προσωπική εικόνα στον ψηφιακό κόσμο και η ενίσχυση των απρόσωπων σχέσεων [8].



Σχήμα 2.1: Η δημοσιεύσεις κοινωνικών γεγονότων έχουν υποχωρήσει αισθητά. (Πηγή: [2])

Προκειμένου να επαναπροσδιοριστεί ο ρόλος των εφαρμογών, είναι απαραίτητο να αναθεωρηθούν τα κίνητρα με τα οποία αυτές κεντρίζουν το ενδιαφέρον του χρήστη. Η κοινωνική επίδειξη, να αντικατασταθεί με την κοινωνική συνεισφορά, ενώ η απομόνωση θα πρέπει να δώσει τη θέση της στην επανένταξη του ατόμου στο κοινωνικό σύνολο. Η ενημέρωση μέσω των εφαρμογών, οφείλει να έχει κοινωνικό χαρακτήρα και όχι να προβάλλει την προσωπική ζωή, ή να ωθεί σε κοινωνικά σύνδρομα τους χρήστες [3].

2.1.2 Ανάγκη της Κοινωνικής Προσφοράς

Έχοντας υπόψη τα παραπάνω, καταλήγει κανείς εύκολα στο συμπέρασμα ότι υπάρχει μεγάλη ανάγκη να επανασυνδεθεί ο ρόλος των κοινωνικών εφαρμογών με την συνεισφορά για το κοινό συμφέρον. Αν και ζούμε σε μία εποχή όπου η τεχνολογία προχωράει με αλματώδεις ρυθμούς, ελάχιστο είναι το ποσοστό του συνόλου που γνωρίζει για τα επιτεύγματα των συγχρόνων του στους διάφορους επιστημονικούς τομείς. Ακόμη κι αν το άτομο εκδηλώνει ενδιαφέρον, είναι δύσκολο να ενημερωθεί όταν όλες οι θεματικές περιστρέφονται γύρω από την προσωπική ζωή. Προκύπτει, λοιπόν μια νέα ανάγκη για κινητοποίηση του χρήστη να αλληλεπιδράσει με το κοινωνικό σύνολο. Αυτό είναι εφικτό χρησιμοποιώντας τις δυνατότητες των ήδη γνωστών εφαρμογών, αυτή τη φορά με σκοπό την ενημέρωση για γεγονότα που αφορούν την ευρύτερη πολιτισμική κοινότητα.

2.1.3 Η Έννοια του Πληθοπορισμού

Συνδυάζοντας τη δύναμη της συλλογικής προσφοράς με την τεχνολογία και τεχνογνωσία που είναι διαθέσιμη σήμερα, η ενημέρωση μπορεί να πάρει νέα διάσταση. Με στοχευμένο προσανατολισμό της κοινωνικής διάθεσης για δράση προς μία συγκεκριμένη κατεύθυνση, η κοινωνία μπορεί να στρατολογήσει τα ίδια τα μέλη της προκειμένου να προάγει τις τεχνολογικές, πολιτισμικές και κοινωνικές εκδηλώσεις, να συντονίσει την πληροφορία και να ενημερώσει το σύνολο. Σύμφωνα με τη *New York Times*, χάρις στην αυξανόμενη συνδεσιμότητα μέσω

του διαδικτύου, εκατομμύρια ανθρώπων μπορούν να συνεισφέρουν ιδέες και πληροφορίες για προβλήματα οποιασδήποτε μορφής. Η πρακτική της συμμετοχής ενός «πλήθους» ή μιας ομάδας για έναν κοινο στόχο ή επίλυση κοινών προβλημάτων, με την επιστράτευση της τεχνολογίας ως διαύλου επικοινωνίας ονομάζεται *πληθοπορισμός* (*crowdsourcing*) [4].

2.1.3.1 Εφαρμογές του Πληθοπορισμού

Τα πεδία στα οποία μπορεί να αξιοποιηθεί η τεχνική του πληθοπορισμού είναι αμέτρητα. Ο,τιδήποτε μπορεί να αποκτήσει συνεργατικό χαρακτήρα. Αναφορικά, παρατίθενται μερικοί κλάδοι όπου ανθεί η τεχνική αυτή:

- Εκπαίδευση
- Οικονομία
- Επιστήμη και Υγεία
- IT
- Διαφήμιση
- Επιχειρηματικότητα
- Κοινωνικές Εκδηλώσεις και *NGO*

2.1.3.2 Μοντέλο Πληθοπορισμού στην Εφαρμογή

Η εφαρμογή σκοπεύει να χρησιμοποιήσει την πρακτική του πληθοπορισμού για να συγκεντρώσει πληροφορίες σχετικές με πολιτισμικές και κοινωνικές εκδηλώσεις. Οι χρήστες θα έχουν τη δυνατότητα να αξιολογούν τα πάντα γύρω από ένα γεγονός. Θα είναι δυνατή η ενημέρωση για τυχόν αλλαγές της ώρας και του τόπου, για προβλήματα που μπορεί να δυσκολέψουν την διεξαγωγή της εκδήλωσης, ή ιδέες για την καλυτέρευση αυτής. Το σημαντικό στοιχείο όλων των παραπάνω είναι πως θα μπορούν να γίνουν σε πραγματικό χρόνο. Αυτό θα έχει σαν αποτέλεσμα την καλύτερη αλληλεπίδραση μεταξύ συμμετεχόντων και διοργανωτών, την αποφυγή προβλημάτων και παρανοήσεων και την καλύτερη ενημέρωση του πλήθους. Διοργανωτές και συμμετέχοντες, θα μπορούν να εκφράσουν τη γνώμη τους, όλοι ως χρήστες της εφαρμογής. Η πληροφορία θα επικαιροποιείται διαρκώς από όσους παρευρίσκονται ήδη εκεί και θα διαδίδεται σε όσους μέχρι τώρα την αγνοούσαν.Το παραπάνω μοντέλο υλοποιείται μέσω ενός συστήματος μηνυμάτων μεταξύ των χρηστών.

2.2 Σχετικές Ερευνητικές Προσπάθειες και Πρότυπα

Αν και το παρόν έργο αποτελεί προσωπική πρωτοβουλία, η ιδέα της διπλωματικής έχει κάποιες επιρροές και από άλλα παράλληλα έργα στον ίδιο τομέα. Τέτοια έργα είναι το *WITHcrowd* (https://www.withcrowd.eu) της ερευνητικής ομάδας του εργαστηρίου Ευφυών Συ-

στημάτων (*ISLAB*) του Εθνικού Μετσόβιου Πολυτεχνείου, που αποτελεί μέρος του ευρωπαϊκού προγράμματος *WITH*. Το έργο αυτό αποτέλεσε πηγή έμπνευσης για το κομμάτι του πληθοπορισμού στην εφαρμογή της παρούσας εργασίας. Χρησιμοποιήθηκαν επίσης το πρότυπο ανοιχού κώδικα *Foursquare API* (https://foursquare.com/developers/apps), και οι διεπαφές αυτού, *Foursquare Autocomplete* και *Foursquare Places*. Συνολικά, η εφαρμογή είναι μια καινοτομία η οποία προσπαθεί να στρέψει ήδη υπάρχουσες πρακτικές προς μια νέα κατεύθυνση, χρησιμοποιώντας τις σύγχρονη τεχνογνωσία για έναν πρωτοποριακό σκοπό.

2.2.1 WITH

Το WITH [5] είναι μια πλατφόρμα επανάχρησης πολιτιστικών δεδομένων. Αποτελεί πρωτοβουλία της ευρωπαϊκής κοινότητας με σκοπό τη συλλογή και ταξινόμηση δεδομένων πολιτισμικού περιεχομένου. Η πλατφόρμα αυτή εκθέτει τις διεπαφές (API) διαφόρων πυλών (portals) και ψηφιακών αποθηκών (repositories) και επιτρέπει στους χρήστες την αναζήτηση ψηφιακού περιεχομένου από μια σειρά διαφορετικών και ανεξάρτητων αποθετηρίων και βάσεων δεδομένων από ένα ενιαίο σημείο πρόσβασης. Τα αποθετήρια που μπορούν να αναζητηθούν περιλαμβάνουν μεταξύ άλλων την Europeana, την Ψηφιακή Δημόσια Βιβλιοθήκη της Αμερικής, το YouTube, το Μουσείο Rijks, την Εθνική Βιβλιοθήκη της Αυστραλίας και την Ψηφιακή Νέα Ζηλανδία. Όλη η δύναμη της πλατφόρμας συγκεντρώνεται στο γεγονός ότι ο χρήστης είναι υπεύθυνος για την επίτευξη του στόχου του προγράμματος. Αυτή την πρακτική επιχειρεί να υιοθετήσει η εφαρμογή που θα αναλυθεί στα επόμενα κεφάλαια.

2.2.2 WITHcrowd

Μια επέκταση του προγράμματος WITH είναι το WITHcrowd, το οποίο είναι μια πλατφόρμα προσανατολισμένη στον πληθοπορισμό. Περιέχει καμπάνιες (Campaigns) στις οποίες ο χρήστης μπορεί να εισέλθει είτε ως επισκέπτης, είτε δημιουργώντας προσωπικό λογαριασμό και να ταξινομήσει, να διαχωρίσει ή γενικότερα να προσφέρει τη γνώση του για τους σκοπούς της καμπάνιας.

2.2.3 Foursquare API

Το *Foursquare API* [6] είναι η διεπαφή της εφαρμογής *Foursquare* και παρέχεται στην κοινότητα της πληροφορικής δωρεάν. Δίνει τη δυνατότητα στους προγραμματιστές να χρησιμοποιήσουν δεδομένα που αφορούν τελικά σημεία (endpoints) σε Ενιαίους Εντοπιστές Πόρων (URLs), όπως τα στοιχεία μιας υπηρεσίας (πχ. τοποθεσία, πληροφορίες επικοινωνίας, διεύθυνση, όνομα, κατηγορία, ώρες λειτουργίας, παροχές κλπ).

Το *Foursquare* έφερε την επανάσταση στα μέσα κοινωνικής δικτύωσης με την καινοτομία του '*check in*'. Η λειτουργία της εφαρμογής Foursquare συνοψίζεται στην αξιολόγηση και άσκηση κριτικής σε κέντρα διασκέδασης, εστιατόρια και χώρους ψυχαγωγίας. Ο χρήστης δημοσιεύει την τοποθεσία του κάνοντας check in και ενημερώνει τους υπόλοιπους χρήστες-φίλους του. Η εφαρμογή που θα υλοποιηθεί κάνει μια απόπειρα να διοχετεύσει την πληροφορία

σε αντίστοιχα μέρη πολιτισμικού ή κοινωνικού περιεχομένου και να την αξιοποιήσει για την αξιολόγησή τους.

2.3 Τεχνολογικό υπόβαθρο - Βασικές Έννοιες

Στη συνέχεια θα γίνει μια εισαγωγή στις τεχνολογίες και τα εργαλεία προγραμματισμού. Έπειτα ακολουθεί η ανάλυση των σύγχρονων τεχνολογικών μέσων που αποτελούν τη βάση των μεθόδων και μοντέλων που χρησιμοποιήθηκαν στην υλοποίηση της εφαρμογής.

Όταν ένας προγραμματιστής αναπτύσσει μία εφαρμογή, καλείται να προσδιορίσει τρία πράγματα:

1. Την φύση της εφαρμογής - αν θα τρέχει σε φυλλομετρητή (Web Application) ή αν θα είναι μητρική (Native Application).

2. Την γλώσσα στην οποία θα αναπτύξει την εφαρμογή (Programming Language)

3. Την πλατφόρμα λογισμικού στην οποία θα υλοποιήσει την εφαρμογή (*SDK*)

2.3.1 Είδη Γλωσσών Προγραμματισμού

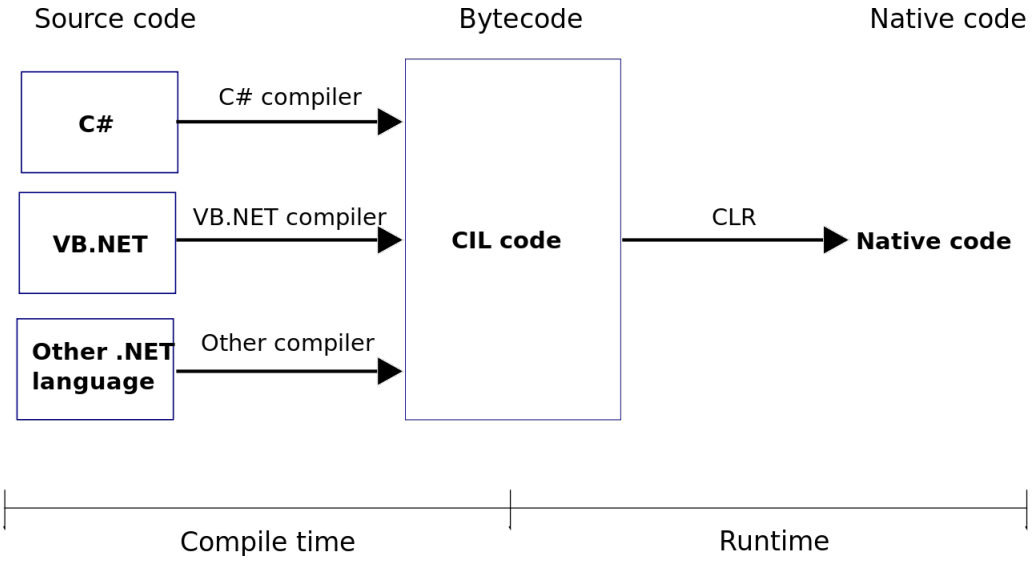
Πρωτού μιλήσουμε για τα είδη των σύγχρονων εφαρμογών, είναι απαραίτητο να αναλύσουμε τις κατηγορίες των αντίστοιχων γλωσσών και τις διακρίσεις μεταξύ αυτών. Μια γλώσσα μπορεί να ανήκει σε μία από τις παρακάτω κατηγορίες:

- Μετταγλωτισμένες ή Μητρικές γλώσσες (Compiled or Native Languages)

- Διαχειριζόμενες Γλώσσες (Managed Languages)

- Δυναμικές Γλώσσες (Dynamic Languages)

Η μητρική γλώσσα (native language) είναι μια γλώσσα που μπορεί να τρέξει στην πλατφόρμα του λειτουργικού συστήματος χωρίς να μετατραπεί σε άλλη μορφή κώδικα από τους μεταγλωττιστές (compilers). Αυτό σημαίνει πως η υλοποίησή της συνοψίζεται κυρίως στη χρήση μεταγλωττιστών, οι οποίοι είναι υπεύθυνοι για τη μετατροπή του κώδικα από γλώσσα μηχανής σε πηγαίο κώδικα (πχ. *C++, C#, Java, Swift*). Η διαχειριζόμενη γλώσσα (managed language) είναι μια γλώσσα που πρέπει να μετατραπεί ή να ερμηνευτεί πριν να εκτελεστεί στην πλατφόρμα (πχ. *.NET*). Σε αυτή την περίπτωση, ο κώδικας θα εκτελεστεί υπό τη διαχείριση μιας εικονικής μηχανής γλώσσας κοινού χρόνου εκτέλεσης ή, όπως είναι γνωστή, *CLR* [10, 11] (βλ. Σχ. 2.2). Η δυναμική γλώσσα προγραμματισμού (dynamic language) είναι μια κλάση αποτελούμενη από γλώσσες υψηλού επιπέδου (high-level programming languages), οι οποίες έχουν την ιδιότητα να εκτελούν πολλαπλές εντολές κατά το στάδιο της εκτέλεσης, σε αντίθεση με τις υπόλοιπες γλώσσες που εκτελούν εντολές στο στάδιο μεταγλώττισης (πχ. *Python, JavaScript, PHP, Ruby, MATLAB, Elixir*) [12, 13].



Σχήμα 2.2: Το CLR μετατρέπει CIL σε μητρικό κώδικα.

Οι γλώσσες υψηλού επιπέδου, όπως η Python και η Ruby, έχουν αποκτήσει μεγάλη απήχηση τα τελευταία χρόνια. Για πολλούς, αποτελούν τη νέα γενιά γλωσσών προγραμματισμού. Ωστόσο οι ρίζες τους ανατρέχουν στις αρχές της δεκαετίας του ′50, με την γέννηση της Lisp της πρώτης γλώσσας υψηλού επιπέδου. Πλέον, οι δυναμικές γλώσσες συναντόνται τόσο σε πραγματικά διαδικτυακά συστήματα, όσο και σε εφαρμογές που πριν κυριαρχούσαν οι στατικές γλώσσες προγραμματισμού [13].

Εμείς θα επικεντρωθούμε στην πρώτη κατηγορία, των μητρικών ή αλλιώς μεταφρασμένων γλωσσών προγραμματισμού. Το πλεονέκτημα χρήσης μητρικού κώδικα (native code) βρίσκεται στο γεγονός ότι προγράμματα σε τέτοιες γλώσσες είναι γρηγορότερα, χάρις τον μειωμένο χρόνο επιβάρυνσης (overhead) της διαδικασίας μετάφρασης. Αυτό οφείλεται στην ιδιότητά τους να μεταγλωττίζονται κατά το χρόνο μεταγλώττισης και όχι κατά το χρόνο εκτέλεσης, όπως συμβαίνει με άλλα προγράμματα.

Οι γλώσσες χαμηλού επιπέδου (low-level programming languages) είναι δομημένες έτσι ώστε να υφίστανται την τυπική μεταγλώττιση, ειδικά όταν η αποτελεσματικότητα είναι πρωταρχικό μέλημα, έναντι της μεταγλώττισης που υποστηρίζει πολλές πλατφόρμες (cross-platform). Για αυτές τις γλώσσες, υπάρχουν περισσότερες ένα-προς-ένα αντιστοιχίες ανάμεσα στο προγραμματισμένο κώδικα και τις λειτουργίες υλικού που εκτελούνται σε κώδικα μηχανής, καθιστώντας ευκολότερο για τους προγραμματιστές τον έλεγχο χρήσης της κεντρικής μονάδας επεξεργασίας και μνήμης με λεπτομέρεια. Με κάποια προσπάθεια, είναι πάντα δυνατή η σύνταξη μεταγλωττιστών ακόμα και για παραδοσιακά ερμηνευμένες γλώσσες. Για παράδειγμα, η Common lisp μπορεί να μεταγλωττιστεί σε Java bytecode (στη συνέχεια ερμηνεύεται από την εικονική μηχανή Java), σε κώδικα C (στη συνέχεια, μεταγλωττίζεται στον εγγενή κώδικα μηχανής) ή απευθείας σε εγγενή κώδικα. Οι γλώσσες προγραμματισμού που υποστηρίζουν πολλαπλούς στόχους σύνταξης δίνουν μεγαλύτερο έλεγχο στους προγραμματιστές για να επιλέξουν είτε ταχύτητα εκτέλεσης είτε συμβατότητα μεταξύ πολλαπλών πλατφόρμων [9].

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Παράθεση 2.1: Παράδειγμα κώδικα σε Python

2.3.2 Μητρικές Γλώσσες Προγραμματισμού (Native Programming Languages)

Έχοντας προσδιορίσει τις κατηγορίες γλωσσών προγραμματισμού, μπορεί κανείς επομένως να αντιληφθεί την ανάγκη των μητρικών ή αλλιώς μεταγλωττισμένων γλωσσών για την ανάπτυξη εφαρμογών σε συγκεκριμένες πλατφόρμες. Αυτές οι γλώσσες είναι κατά κύριο λόγο σχεδιασμένες να τρέχουν σε μια ορισμένη πλατφόρμα για καλύτερη απόδοση και ευκολότερο σχεδιασμό. Τέτοιες γλώσσες που χρησιμοποιούνται σήμερα στις εφαρμογές κινητών συσκευών είναι η Swift και η Java. Η πρώτη χρησιμοποιείται για την υλοποίηση εφαρμογών σε iOS πλατφόρμες, ενώ η δεύτερη για την ανάπτυξη εφαρμογών που τρέχουν σε Android πλατφόρμες.

2.3.2.1 Swift - iOS Development

Η *Swift* είναι μια μεταγλωττισμένη, γενικού σκοπού, πολυπαραδειγματική γλώσσα προγραμματισμού που έχει αναπτυχθεί από την *Apple Inc.* για τα προϊόντα τις ίδιας εταιρείας

(*iOS, macOS, watchOS, tvOS*). Είναι σχεδιασμένη ώστε να δουλεύει με το *XCode IDE* και χρησιμοποιείται από την κοινότητα των iOS developers για ανάπτυξη iOS εφαρμογών, υποστηριζόμενων από τις πλατφόρμες και τα προϊόντα Apple [14].

Τα πρώτα βήματα για την δημιουργία της Swift ξεκίνησαν υπό την καθοδήγηση του C. Lattner, εντός της Apple. Με επιρροές από γλώσσες όπως οι *Objective-C, Rust, Haskell, C#* και αρκετές ακόμη [15], η Swift ήρθε στο προσκήνιο επίσημα για πρώτη φορά στο Παγκόσμιο Συνέδριο Προγραμματιστών (WWDC) τον Ιούνιο του 2014 [16].

Βασικά γνωρίσματα αυτής της γλώσσας είναι η αντικειμενοστρέφεια (*OO* language), και οι απλουστευμένες δομές. Η Swift βασίζεται σε θεωρητικές έννοιες των μοντέρνων γλωσσών προγραμματισμού και προσπαθεί να παρουσιάσει μια απλούστερη συντακτική προσέγγιση [17, 18].

```
    import UIKit
    import AVFoundation

class ViewController: UIViewController {

    @IBOutlet weak var darkBlueBG: UIImageView!
    @IBOutlet weak var powerBtn: UIButton!
    @IBOutlet weak var cloudHolder: UIView!
    @IBOutlet weak var rocket: UIImageView!
    @IBOutlet weak var hustleLbl: UILabel!
    @IBOutlet weak var onLbl: UILabel!

    var player: AVAudioPlayer!

    override func viewDidLoad() {
        super.viewDidLoad()

        let path = Bundle.main.path(forResource: "hustle-on", ofType: "wav")!
        let url = URL(fileURLWithPath: path)
        do {
            player = try AVAudioPlayer(contentsOf: url)
            player.prepareToPlay()
        } catch let error as NSError {
            print(error.description)
        }
    }

    @IBAction func powerBtnPressed(_ sender: Any) {
        cloudHolder.isHidden = false
        darkBlueBG.isHidden = true
        powerBtn.isHidden = true

        player.play()

        UIView.animate(withDuration: 2.3, animations: {
            self.rocket.frame = CGRect(x: 0, y: 140, width: 375, height: 402)
        }) { (finished) in
```

```
38              self.hustleLbl.isHidden = false
39              self.onLbl.isHidden = false
40          }
41      }
42  }
```

Παράθεση 2.2: Παράδειγμα κώδικα σε Swift

2.3.2.2 Java - Android Development

Η *Java* είναι γενικού σκοπού, ταυτοχρονισμένη (concurrent), βασισμένη σε κλάσεις (class-based) και αντικειμενοστραφής (*OO*) γλώσσα [19]. Έχει σχεδιαστεί ειδικά για να επιτρέπει όσο το δυνατό μεγαλύτερη ανεξαρτησία στον προγραμματιστή και ευελιξία όσον αφορά τη συγγραφή κώδικα με συνδυασμό πολλών βιβλιοθηκών. Ακολουθεί τη νοοτροπία '*WORA*' [20], με αποτέλεσμα να μπορεί να τρέξει σε όλες τις πλατφόρμες που υποστηρίζουν Java χωρις να χρείαζεται επαναμεταγλώττιση [21]. Από το 2016 η Java έχει ανέλθει σε μία από τις δημοφιλέστερες εν ενεργεία γλώσσες προγραμματισμού [22, 23, 24].

Η Java εμφανίστηκε στο προσκήνιο το 1991, χάρις στον J. Gosling και τους συνεργάτες του. Η πρώτη επίσημη δημοσίευση έγινε το 1996 από την εταιρεία Sun Microsystems [26]. Σύντομα, ενσωματώθηκε στους σημαντικότερους φυλλομετρητές ιστού (web browsers), και σε ιστοσελίδες (web pages), πράγμα που την μετέτρεψε σε ισχυρό προγραμματιστικό εργαλείο. Η Java δέχθηκε αρκετές επιρροές από τη C++, όσον αφορά το συντακτικό κομμάτι. Όντας επίσης μιας αντικειμενοσταφής γλώσσα προγραμματισμού, η Java θεμελιώνεται σε κλάσεις (classes) και κάθε ομάδα δεδομένων αποτελεί ένα '*αντικείμενο*' (*object*). Εξαίρεση αποτελούν οι πρωτογενείς τύποι δεδομένων (πχ *integers, floating points, boolean values* κλπ)

Το 2005, η Google ανακοίνωσε ένα νέο λειτουργικό σύστημα προοριζόμενο για φορητές συσκευές με το όνομα Android. Πρόκειται για μια παραλλαγή του πυρήνα του λειτουργικού συστήματος Linux, με προσθήκες από κάποιες ακόμη βιβλιοθήκες ανοιχτού λογισμικού. Το λογισμικό Android υποστηρίζεται από συσκευές 3ης γενιάς (smartphones, tablets) καθώς και άλλα προϊόντα της Google (πχ *Android Auto, Android TV, Wear OS* κλπ). οι εφαρμογές που αναπτύσσονται με αυτό το λειτουργικό σύστημα υλοποιούνται με τη χρήση της πλατφόρμας *Android Studio SDK* [27] και της γλώσσας Java [28]. Την πλατφόρμα αυτή συνοδεύουν και άλλα χρήσιμα εργαλεία, όπως αποσφαλματωτής (debugger), βιβλιοθήκες λογισμικού και προσομοιωτής (emulator) [29].

```
1  // This is an example of a single line comment using two slashes
2
3  /* This is an example of a multiple line comment using the slash and asterisk.
4   This type of comment can be used to hold a lot of information or deactivate
5   code, but it is very important to remember to close the comment. */
6
7  package fibsandlies;
8  import java.util.HashMap;
9
10 /**
```

```java
 * This is an example of a Javadoc comment; Javadoc can compile documentation
 * from this text. Javadoc comments must immediately precede the class, method,
    or field being documented.
 */
public class FibCalculator extends Fibonacci implements Calculator {

    private static Map<Integer, Integer> memoized = new HashMap<Integer,
    Integer>();

    /*
     * The main method written as follows is used by the JVM as a starting
    point for the program.
     */
    public static void main(String[] args) {
        memoized.put(1, 1);
        memoized.put(2, 1);
        System.out.println(fibonacci(12)); //Get the 12th Fibonacci number and
    print to console
    }

    /**
     * An example of a method written in Java, wrapped in a class.
     * Given a non-negative number FIBINDEX, returns
     * the Nth Fibonacci number, where N equals FIBINDEX.
     * @param fibIndex The index of the Fibonacci number
     * @return The Fibonacci number
     */
    public static int fibonacci(int fibIndex) {
        if (memoized.containsKey(fibIndex)) {
            return memoized.get(fibIndex);
        } else {
            int answer = fibonacci(fibIndex - 1) + fibonacci(fibIndex - 2);
            memoized.put(fibIndex, answer);
            return answer;
        }
    }
}
```

Παράθεση 2.3: Παράδειγμα κώδικα σε Java

2.3.3 JavaScript in Mobile Applications

Η *JavaScript*, γνωστή και με την ακρωνυμία *JS*, είναι μία γλώσσα υψηλού επιπέδου, που ανήκει στην κατηγορία των μεταγλωττισμένων γλωσσών προγραμματισμού. Συμμορφώνεται στους διακονινισμούς που ορίζονται από το πρότυπο *ES* [30]. Χρησιμοποιείται κυρίως στον client-side προγραμματρισμό, αποτελόντας τον πυρήνα των τεχνολογιών που χρησιμοποιούνται σήμερα στην πλειοψηφία των web εφαρμογών και ιστοσελίδων [31]. Σαν πολυπαραδειγματική γλώσσα, η JavaScript υποστηρίζει όλες τις σύχρονες τεχνικές προγραμματισμού, όπως συναρτησιακό (functional), οδηγούμενο από γεγονότα (event-driven) και αντικειμενοστραφή

(object-oriented) προγραμματισμό.

Από το Μάιο του 2017, το 94.5% των δημοφιλέστερων ιστοσελίδων είναι γραμμένα σε JavaScript. Ο πιο συνηθισμένος τρόπος χρήσης της είναι για την ανάπτυξη προγραμμάτων-πελατών (clients) που είναι γραμμένα σε τυπική HTML. Το κυριότερο χαρακτηριστικό της είναι ότι προσδίδει κάποιες χρηστικές ιδιότητες στον καθιερωμένο κώδικα, που καθιστούν τα προγράμματα περισσότερο αποκρίσιμα, μέσω πληθώρας στοιχείων αλληλεπίδρασης με το *DOM*. Οι πιο κοινές περιπτώσεις αυτών είναι:

- Η φόρτωση νέου περιεχομένου σελίδας ή η υποβολή δεδομένων στον διακομιστή μέσω του συστήματος *AJAX* χωρίς επαναφόρτωση της σελίδας (για παράδειγμα, ένα κοινωνικό δίκτυο ενδέχεται να επιτρέψει στο χρήστη να δημοσιεύει ενημερώσεις κατάστασης χωρίς να εγκαταλείψει τη σελίδα).
- Κινούμενα σχέδια των στοιχείων της σελίδας, fade out/in, slide out/in, αλλαγή μεγέθους, αποκοπή, μετακίνηση, κλπ.
- Διαδραστικό περιεχόμενο, για παράδειγμα παιχνίδια, αναπαραγωγή ήχου και βίντεο.
- Επαλήθευση των τιμών εισαγωγής μιας φόρμας διαδικτύου για επιβεβαίωση ότι είναι αποδεκτές πριν υποβληθεί στο διακομιστή.
- Μετάδοση πληροφοριών σχετικά με τις συνήθειες ανάγνωσης του χρήστη και τις δραστηριότητες περιήγησης σε διάφορους ιστότοπους. Οι ιστοσελίδες το συνηθίζουν αυτό σε αναλύσεις στο Web, για παρακολούθηση διαφημίσεων, εξατομίκευση ή για άλλους σκοπούς.

Τα παραπάνω επιτυγχάνονται πρακτικά με την ενσωμάτωση τμημάτων κώδικα (script objects) γραμμένο σε JS, εντός του κυρίου μέρους (body) που είναι γραμμένο σε παραδοσιακή HTML. Ένα πρόγραμμα περιήγησης στο web είναι το πιο κοινό περιβάλλον υποδοχής για JavaScript. Εκεί δημιουργούνται συνήθως '*αντικείμενα φιλοξενίας*' (host objects) για την αναπαράσταση του DOM. Ο διακομιστής Web (server) είναι ένα άλλο κοινό περιβάλλον υποδοχής. Ένας Web JavaScript server συνήθως εκθέτει αντικείμενα φιλοξενίας που αντιπροσωπεύουν αντικείμενα αιτήματος και απόκρισης HTTP (HTTP requests), τα οποία ένα πρόγραμμα σε JavaScript θα μπορούσε στη συνέχεια να αξιολογήσει και να διαχειριστεί προκειμένου να δημιουργήσει δυναμικά ιστοσελίδες.

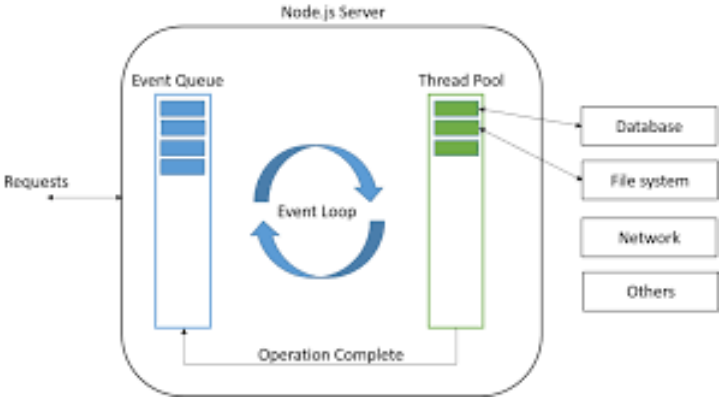
```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <button id="hellobutton">Hello</button>
    <script>
        document.getElementById('hellobutton').onclick = function() {
            alert('Hello world!');  // Show a dialog
```

```
            var myTextNode = document.createTextNode('Some new words.');
            document.body.appendChild(myTextNode);  // Append "Some new words"
    to the page
        };
    </script>
  </body>
</html>
```

Παράθεση 2.4: Μικρό παράδειγμα μιας ιστοσελίδας που συμμορφώνεται με τα πρότυπα και περιέχει JavaScript (χρησιμοποιώντας τη σύνταξη HTML5) και το DOM

2.3.3.1 Βρόχος Συμβάντων - Event Loop

Στη JavaScript, σχεδόν όλες οι λειτουργίες εισόδου/εξόδου (I/O) εκτελούνται ασύγχρονα, χωρίς μπλοκάρισμα (non-blocking execution). Σε αυτές περιλαμβάνονται HTTP αιτήσεις, ενέργειες πάνω σε βάσεις δεδομένων, εγγραφές και αναγνώσεις από το σκληρό δίσκο. Το μοναδικό νήμα ζητά από το περιβάλλον εκτέλεσης να πραγματοποιήσει μια ενέργεια, παρέχοντάς του μια συνάρτηση επανάκλησης (callback function), συνεχίζοντας με την εκτέλεση άλλων εργασιών. Όταν η ενέργεια που ζητήθηκε προηγουμένως ολοκληρωθεί, ένα μήνυμα εισάγεται σε μια ουρά μαζί με το παρεχόμενο ανάκλησης (callback content). Κάποια στιγμή στο μέλλον, το μήνυμα αυτό θα αφαιρεθεί από την ουρά και η συνάρτηση επανάκλησης θα εκτελεστεί. Αυτό το διαδραστικό, πλήρως ασύγχρονο, μοντέλο μπορεί να είναι γνώριμο στους προγραμματιστές που αναπτύσσουν λογισμικό διεπαφής χρήστη (API) – εκεί που συμβάντα όπως το πάτημα ενός κουμπιού ή η κύλιση του παραθύρου μπορούν να προκύψουν οποιαδήποτε στιγμή και πρέπει να «εξυπηρετηθούν», αλλά διαφέρει σημαντικά από το σύγχρονο μοντέλο αίτησης-απόκρισης που συναντάται σε τυπικές υλοποιήσεις εφαρμογών εξυπηρετητή. Αυτή η απεμπλοκή του καλούντος από την απάντηση που αυτός αναμένει, επιτρέπει στο περιβάλλον εκτέλεσης της JavaScript να ασχοληθεί με άλλες διεργασίες ενώ «περιμένει» την ασύγχρονη ενέργεια να διεκπεραιωθεί και να έρθει η στιγμή να εκκινήσει το callback μέρος. Αυτή, λοιπόν, η ουρά –αόρατη στον προγραμματιστή– στην οποία τα μηνύματα αποθηκεύονται προσωρινά μαζί με τα αντίστοιχα εγγεγραμμένα callback, ονομάζεται βρόχος συμβάντων (βλ. Σχ. 2.3).



Σχήμα 2.3: Βρόχος συμβάντων της *JavaScript* (Πηγή: [32])

2.3.3.2 Run to Completion Logic

Η πρακτική που ακολουθεί το περιβάλλον της JavaScript είναι η πλήρης επεξεργασία κάθε μηνύματος προτού συνεχίσει με το επόμενο. Αυτό προσφέρει κάποιες ελκυστικές ιδιότητες κατά το σχεδιασμό της λογικής των προγραμμάτων, συμπεριλαμβανομένης της εγγύησης της εγκυρότητας των δεδομένων κατά τη διάρκεια της εκτέλεσης συναρτήσεων. Παρατηρείται δηλαδή διαφορά με το μοντέλο της C, κατά το οποίο, όταν μία συνάρτηση ή κομμάτι κώδικα εκτελείται μέσα σε κάποιο νήμα, το σύστημα έχει τη δυνατότητα να διακόψει την εκτέλεσή τους και να μεταφέρει τον έλεγχο σε κάποιο άλλο νήμα. Βέβαια, υπάρχουν και μειονεκτήματα στην προσέγγιση αυτή, το σπουδαιότερο εκ των οποίων έχει να κάνει με το εάν ένα μήνυμα χρειαστεί σημαντικό χρονικό διάστημα για την επεξεργασία του, όλη η εφαρμογή καθίσταται ανίκανη να διαχειριστεί οποιαδήποτε αλληλεπίδραση με το χρήστη. Δεν είναι σπάνια, ακόμα και σήμερα, η εμφάνιση ειδοποίησης με τη μορφή ξεχωριστού παραθύρου από τον browser, σύμφωνα με την οποία «η εκτέλεση κάποιου script καθιστά την εφαρμογή μη αποκρίσιμη». Οι προγραμματιστές για να υπερκεράσουν τους περιορισμούς αυτούς προσπαθούσαν να μειώσουν όσο γίνεται το χρόνο επεξεργασίας που απαιτείτο ή, αν αυτό δεν ήταν δυνατό, να «μοιράσουν» το φόρτο σε μικρότερα μηνύματα, τα οποία τοποθετούσαν εκ νέου στο βρόχο συμβάντων χρησιμοποιώντας ειδικά API calls, όπως *setTimeout()* και *setInterval()*.

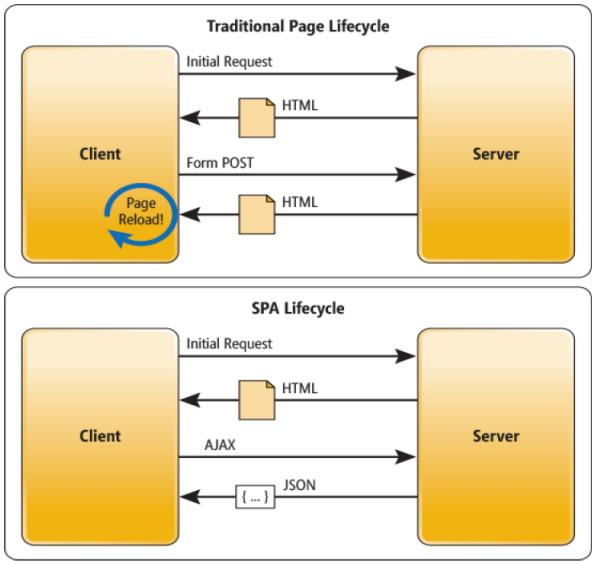
2.3.3.3 Asynchronous JavaScript and XML

Όπως έχει ήδη αναφερθεί, στο παρελθόν η HTML αποτελούσε τη μοναδική προσέγγιση στατικών ιστοσελίδων και web υπηρεσιών. Η εμφάνιση της JS προσσέγισης έφερε την πραγματική επανάσταση στην ανάπτυξη εφαρμογών και διαδικτυακών προγραμμάτων. Σε αυτό συνέβαλλε κυρίως η εμφάνιση των AJAX τεχνικών [33, 34], οι οποίες μετατόπισαν τον έλεγχο της ροής και της πληροφορίας από τους εξυπηρετητές στους πελάτες. Ο client side προγραμματισμός έχει αναβαθμιστεί και πλέον είναι εφικτή η αλληλεπίδραση μεταξύ του back-end και του frontend μιας εφαρμογής μέσω αμοιβαίων ασύγχρονων αιτημάτων. Με αυτό τον τρόπο, ο client μπορεί να αιτηθεί δεδομένα από τον server και, όταν αυτά είναι διαθέσιμα, να τα αναπαραστήσει ή να τα επεξεργαστεί κατάλληλα.

2.3.3.4 Single Page Applications

Η επόμενη καθοριστική αλλαγή ήρθε το 2002 με την είσοδο της έννοιας των εφαρμογών μονής σελίδας – *SPA*. Η λογική εδώ είναι ότι μια web εφαρμογή λαμβάνει χώρα σε μία μόνο ιστοσελίδα με στόχο να παρέχει μια πιο άμεση και ομαλή εμπειρία συγκρίσιμη με εφαρμογές επιφάνειας εργασίας (desktop applications). Η σελίδα φορτώνεται μία μόνο φορά και στη συνέχεια τα δεδομένα φορτώνονται δυναμικά μέσω αλληλεπίδρασης server – client, ενώ ταυτόχρονα εξακολουθεί να δίνει στον χρήστη την εντύπωση οτι πλοηγείται σε διαφορετικές σελίδες, εντός της εφαρμογής [35].

Αυτή η στροφή στον τρόπο διαχείρισης δεδομένων, κατέρριψε τα εμπόδια μεταξύ απόδοσης και πολυπλοκότητας.



Σχήμα 2.4: Διαφορές μεταξύ παραδοσιακής λογικής ιστοσελίδας και λογικής *SPA*.

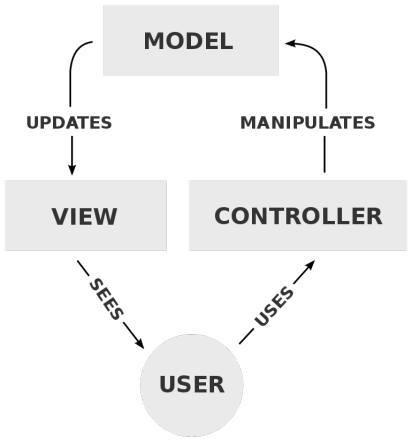
2.3.3.5 Model View Controller

Ωστόσο, πολυπλοκότητα και η ποσότητα της λογικής που δόθηκε στον πελάτη δημιούργησε νέα προβλήματα. Το μέγεθος του κώδικα αυξήθηκε δραματικά, και οι τεχνοτροπίες που δημιουργήθηκαν ποίκιλλαν πολύ. Η λύση δόθηκε ομαδοποιώντας πρακτικές και σχεδιαστικές επιλογές σε ολοκληρωμένες τεχνοτροπίες (frameworks) που μέχρι στιγμής υπήρχαν μόνο στους εξυπηρετητές. Η πιο γνωστή όλων αυτών των JavaScript Frameworks είναι η τεχνοτροπία Μοντέλου – Όψης – Ελεγκτή, γνωστή και ως *MVC*(βλ. Σχ. 2.5), η οποία αποτελείται από τα εξής τρία μέρη:

- **Μοντέλα** – ***Models*** που αναπαριστούν τις σχετικές με την εφαρμογή πληροφορίες και δεδομένα, όπως συγκεκριμένες κλάσεις-δοχεία (container classes) δεδομένων. Τα μοντέλα μπορούν να ενημερώνουν τυχόν παρατηρητές όταν η κατάστασή τους αλλάζει (π.χ. όταν κάποια πληροφορία που κρατούν ενημερώνεται ή διαγράφεται) [36].
- **Όψεις** – ***Views*** που τυπικά θεωρούνται ως η διεπαφή του χρήστη με την εφαρμογή (π.χ. ο κώδικας HTML και CSS). Πρέπει να γνωρίζουν για την ύπαρξη των μοντέλων, έτσι ώστε να τα παρατηρούν, αλλά δεν επικοινωνούν κατευθείαν μαζί τους.
- **Ελεγκτές** – ***Controllers*** που υλοποιούν τη λογική παρουσίασης (presentation logic) της εφαρμογής. Αυτοί είναι που παίρνουν τις αποφάσεις και ο συνδετικός κρίκος μεταξύ Μοντέλων και Όψεων.

2.3.4 Η Ανάγκη Μιας Νέας Τεχνολογίας - React Framework

Ωστόσο, η αύξηση της πολυπλοκότητας των mobile applications συνεπάγεται αύξηση των απαιτήσεων. Πλέον, η ανάπτυξη μιας native εφαρμογής συνοδεύεται από πολλαπλά καθήκοντα.



Σχήμα 2.5: Διάγραμμα αναπαράστασης των αλληλεπιδράσεων βάσει του μοντέλου *MVC*.

Η υλοποίηση κώδικα που μπορεί να μεταφερθεί με ελάχιστες τροποποιήσεις μεταξύ πολλαπλών πλατφορμών (cross-platform apps), η ανάγκη για εφαρμογές που υποστηρίζονται και από το web (web apps), το συνολικό κόστος ανάπτυξης, καθώς και ο χρόνος ανάπτυξης είναι μερικοί από τους βασικότερους παράγοντες που συνέδραμαν στην αναζήτηση μιας καινοτόμου λύσης.

Έτσι, το 2011 η ομάδα προγραμματιστών του Facebook διαμόρφωσε μια νέα τεχνολογία για την ανάπτυξη web και native εφαρμογών, γνωστή ως React (*React.js* ή *ReactJS*). Τα βασικότερα χαρακτηριστικά της React είναι:

- η αρχιτεκτονική διαίρεσης του κώδικα σε μικρότερα στοιχεία, τα οποία καλούνται components (component-based architecture) [37],
- η μονοσήμαντη ροή δεδομένων μέσω των '*ιδιοτήτων*' (*props*) ενός στοιχείου (one-way data binding) [37],
- η δυνατότητα αποθήκευσης της κατάστασης ενός στοιχείου (stateful component) και μετάδοσης της κατάστασης (state) σε στοιχεία-παιδιά (child-component) του στοιχείου-πατέρα (parent-component) μέσω των props,
- η JSX σύνταξη [39]
- η ανανέωση μόνο εκείνου του περιεχομένου της σελίδας που μεταββάλεται (virtual DOM) [38] και
- η δυνατότητα εκτέλεσης τμημάτων κώδικα σε δεδομένα χρονικά σημεία κατά τον κύκλο ζωής του στοιχείου με τη βοήθεια συγκεκριμένων hooks (lifecycle methods), τα βασικότερα εκ των οποίων είναι *render(), componentDidMount(), componentWillUnmount()*

Η φιλοσοφία ενιαίου κώδικα σε όλες τις πλατφόρμες απέκτησε δημοτικότητα με ραγδαίους ρυθμούς. Ως αποτέλεσμα, η κοινότητα προγραμματιστών του Facebook σηματοδότησε επισήμως το 2015 την έναρξη μιας νέας εποχής, με την ανακοίνωση της React Native.

```
class Board extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      squares: Array(6).fill(null),
    };
  }

  handleClick(i) {
    const squares = this.state.squares.slice();
    squares[i] = 'X';
    this.setState({squares: squares});
  }

  renderSquare(i) {
    return (
      <Square
        value={this.state.squares[i]}
        onClick={() => this.handleClick(i)}
      />
    );
  }

  render() {
    const status = 'Next player: X';
    return (
      <div>
        <div className="status">{status}</div>
        <div className="board-row">
          {this.renderSquare(0)}
          {this.renderSquare(1)}
          {this.renderSquare(2)}
        </div>
        <div className="board-row">
          {this.renderSquare(3)}
          {this.renderSquare(4)}
          {this.renderSquare(5)}
        </div>
      </div>
    );
  }
}
```

Παράθεση 2.5: Παράδειγμα κώδικα σε React (Πηγή:[40])

2.3.4.1 React Native

Η React Native, είναι ένα Framework για την ανάπτυξη πραγματικών, native εφαρμογών για iOS και Android πλατφόρμες. Η φιλοσοφία της React Native είναι πανονομοιότυπη με αυτή της React, και στηρίζεται στους ίδιους κανόνες και τεχνικές που διέπουν τα πλαίσια της

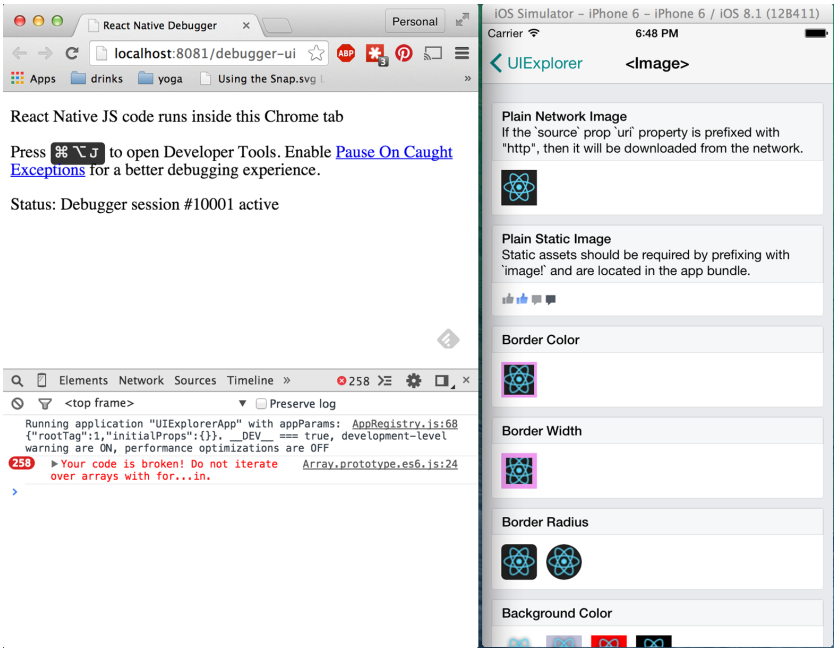
JavaScript, με τη μόνη διαφορά ότι αποσκοπεί στην σχεδίαση διεπιφανειών που απευθύνονται σε πλατφόρμες κινητών συσκευών, έναντι προγραμμάτων περιήγησης. Με άλλα λόγια, αποτελεί έναν νέο τρόπο υλοποίησης εφαρμογών που δε διαφέρουν σε τίποτα από εφαρμογές γραμμένες σε native γλώσσες προγραμματισμού (Swift, Java). Έτσι, είναι πλέον εφικτή η ανάπτυξη εφαρμογών για πολλαπλές πλατφόρμες, με τη χρήση μιας μόνο γλώσσας, η οποία υποστηρίζεται και από τις εφαρμογές web. Το χρονοδιάγραμμα υλοποίησης μειώνεται αισθητά, αφού η διαδικασία ανάπτυξης iOS και Android εφαρμογών μπορεί να γίνει τώρα παράλληλα. Εντούτοις, το κόστος ανάπτυξης της εφαρμογής δεν εξαρτάται πλέον από τη δυνατότητα ή μη υποστήριξης της εφαρμογής από περισσότερες από μια πλατφόρμες.

Τα πλεονεκτήματα που προάγουν τη React Native ως το δημοφιλέστερο cross-platform framework είναι πολλά. Όπως και η React, έτσι και η React Native χρησιμοποιεί JSX σύνταξη. Ωστόσο, στην υλοποίηση χρησιμοποιούνται οι μητρικές διεπαφές κάθε πλατφόρμας (Objective-C για iOS και Java για Android). Το γεγονός αυτό καθιστά τις εφαρμογές εξίσου αποδοτικές, με αυτές σε native κώδικα. Έτσι, οι προγραμματιστές που είναι εξοικειωμένοι με την ανάπτυξη web εφαρμογών μπορούν να εκμεταλλευτούν τη συγγένεια των δύο αυτών γλωσσών.

Προγραμματιστική Εμπειρία – Developer Experience

Για προγραμματιστές με προϋπάρχουσα εμπειρία στην ανάπτυξη mobile εφαρμογών, το περιβάλλον της React Native είναι ιδιαίτερα φιλικό. Η ομάδα της React Native προσφέρει στον προγραμματιστή ισχυρά προγραμματιστικά εργαλεία που διευκολύνουν τη διαδικασία ανάπτυξης μιας εφαρμογής και μειώνουν αισθητά το χρόνο προγραμματισμού. Για παράδειγμα, δεν απαιτείται rebuild κάθε φορά που ο προγραμματιστής επιθυμεί να προβάλλει τις αλλαγές στην εφαρμογή, αλλά αρκεί μια απλή ανανέωση της σελίδας, όπως σε μια συνηθισμένη ιστοσελίδα. Αυτό εντίνει ιδιαίτερα την αμεσότητα στη διαδικασία επικοινωνίας μεταξύ κώδικα και εφαρμογής και έτσι ο προγραμματιστής μπορεί να ελέγχει τις αλλαγές του κάθε φορά. Αυτό έχει σημαντικό αντίκτυπο στην περίπτωση σχεδιασμού (design) της εφαρμογής, όπου λεπτομερείς αλλαγές λαμβάνουν χώρα διαρκώς και η εφαρμογή μπορεί να ανανεώνεται αυτομάτως.

Επίσης, το περιβάλλον υποστήριξης της React Native συνοδεύεται από ολοκληρωμένη και επικαιροποιημένη βιβλιογραφία, ενώ σημαντική προσοχή δίνεται και στη διαδικασία αποσφαλμάτωσης του κώδικα μέσω επεξηγηματικών μηνυμάτων λάθους. Ο προγραμματιστής μπορεί να επωφεληθεί από τα έξυπνα εργαλεία εντοπισμού σφαλμάτων και την αναφορά σφαλμάτων που είναι ενσωματωμένα στους σύγχρονους περιηγητές όπως ο Chrome ή ο Safari (βλ. Σχ. 2.6). Παρομοίως, η ίδια ευελιξία χαρακτηρίζει και την επιλογή της πλατφόρμας προγραμματισμού, καθώς ο προγραμματιστής είναι ελεύθερος να επιλέξει εκείνο τον text editor με τον οποίο διαθέτει τη μέγιστη εξοικείωση. Περιορισμοί όπως ο προγραμματισμός σε συγκεκριμένο λογισμικό για εφαρμογές iOS ή Android, που χαρακτηρίζουν τις native γλώσσες προγραμματισμού (όπως αναφέρθηκε στην ενότητα 2.3.2) δεν αποτελούν εμπόδιο για την React Native. Όλα τα παραπάνω συνεισφέρουν σε μία πρωτοποριακή και εποικοδομητική εμπειρία, που επιτρέπει την επικέντρωση στο προγραμματιστικό κομμάτι και εξασφαλίζει ταχύτητα και ευκολία.



Σχήμα 2.6: Χρήση του ενσωματωμένου αποσφαλματωτή στον περιηγητή Chrome

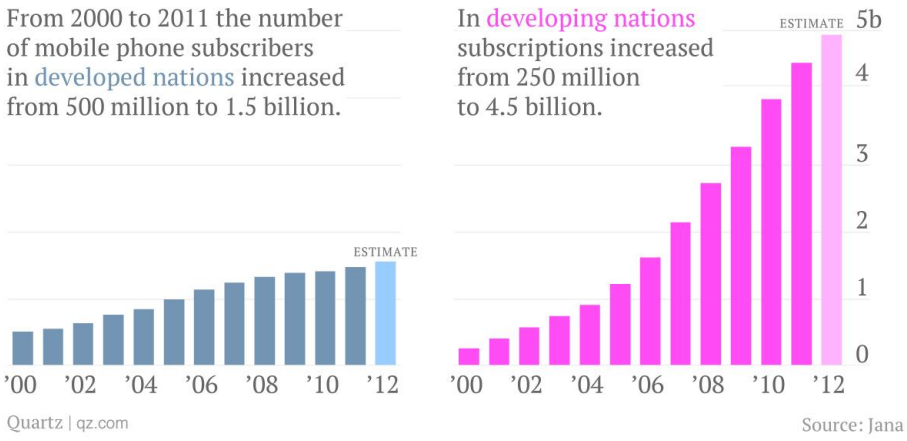
Επαναχρησιμότητα και Κοινή Χρήση – Code Reuse and Knowlegde Sharing

Η εργασία στο περιβάλλον της React Native μπορεί να ελαχιστοποιήσει δραματικά τους απαιτούμενους πόρους για την ανάπτυξη μιας mobile εφαρμογής. Οποιοσδήποτε διαθέτει γνώση της React μπορεί να στοχεύσει την ανάπτυξη, τόσο web εφαρμογών, όσο και native (iOS και Android) χωρίς περιορισμούς. Έτσι, παύει να υπάρχει διάκριση μεταξύ προγραμματιστών. Χρησιμοποιώντας το ίδιο σύνολο δεξιοτήτων και γνώσεων, κάθε ομάδα προγραμματιστών μπορεί να εργαστεί ταχύτερα, αμεσότερα και αποτελεσματικότερα, αφού η γνώση είναι κοινή και μπορεί να μοιραστεί ανά πάσα στιγμή.

Εκτός από την κοινή χρήση της γνώσης, μεγάλα τμήματα κώδικα μπορούν επίσης να επαχρησιμοποιηθούν. Προφανώς, ενδέχεται η απαίτηση περαιτέρω τροποποίησης κάποιων σημείων του κώδικα που αφορά κάθε πλατφόρμα. Παρ΄ όλα αυτά, η επαναχρησιμοποίηση κώδικα μεταξύ διαφόρων πλατφόρμων είναι ιδιαίτερα εύκολη και γρήγορη. Αρκούν κάποιες συνθήκες και η χρήση κάποιων διαφορετικών διεπαφών προκειμένου να καταστεί εφικτή η cross-platform λειτουργικότητα. Για παράδειγμα, η εφαρμογή διαχείρισης διαφημίσεων του Facebook για Android μοιράζεται το 87% της βάσης του κώδικα με την αντίστοιχη εφαρμογή για iOS [41].

Εξοικονόμηση Πόρων – Resource Saving

Στην εποχή αυτή της τεχνολογίας, συναντάται συχνά η απαίτηση υψηλής ποιότητας με χαμηλό κόστος. Η React Native θεμελιώνεται σε αυτά τα δεδομένα, προσφέροντας υψηλή απόδοση με ελαχιστοποίηση των απαιτούμενων πόρων. Οι περισσότερες ηλεκτρονικές υπηρεσίες και εταιρείες σήμερα έχουν την ανάγκη να αντιπροσωπεύονται τόσο στο διαδίκτυο μέσω



Σχήμα 2.7: Η χρήση εφαρμογών σε κινητά αυξάνεται εκθετικά.

μιας web εφαρμογής, όσο και στις φορητές συσκευές μέσω native εφαρμογών. Το γεγονός ότι η χρήση φορητών συσκευών αυξάνεται με εκθετικούς ρυθμούς την τελευταίες δεκαετίες (βλ. Σχ. 2.7), έχει στρέψει την προσοχή στην δεύτερη κατηγορία εφαρμογών. Όπως προαναφέρθηκε, είναι πλέον εφικτή η ανάπτυξη μιας εφαρμογής που ανταποκρίνεται σε πολλές πλατφόρμες, με ελάχιστες τροποποιήσεις. Ως αποτέλεσμα, οι ομάδες προγραμματιστών εντός μιας εταιρείας είναι μικρότερες και πιο ευέλικτες. Ακόμη, τα χρονικά πλαίσια ολοκλήρωσης μιας εφαρμογής είναι συντομότερα και δεν αυξάνονται ανάλογα με τον αριθμό πλατφόρμων υποστήριξης της εφαρμογής, όπως άλλοτε. Τέλος, είναι ευνόητο πως τα παραπάνω ευνοούν την εξοικονόμηση χρηματικών πόρων που απαιτούνται για την ανάπτυξη μιας εφαρμογής [42]. Αναμφισβήτητα λοιπόν, η React Native αποκτά μεγάλη δημοτικότητα και για αυτούς τους λόγους, γνωστές εφαρμογές όπως η AirBnB, το Skype, το Instagram και το Facebook την έχουν υιοθετήσει [43].

2.3.4.2 React Native Bridge

Η φιλοσοφία πίσω από τον τρόπο λειτουργίας της React Native είναι απλή. Προκειμένου να προσφέρει στον προγραμματιστή την ίδια εμπειρία με τις native εφαρμογές, χρησιμοποιεί πανομοιότυπες διεπαφές και διεπιφάνειες με τις αντίστοιχες των native γλωσσών. Πιο συγκεκριμένα, η React Native απαρτίζεται από δύο τμήματα (realms):

- το τμήμα της JavaScript (JavaScript Realm) και
- το τμήμα των Native UIs (Native Realm)

Τα δύο τμήματα επικοινωνούν μεταξύ τους χρησιμοποιώντας μια '*γέφυρα*' (bridge), η οποία αποτελεί αναμφισβήτητα τον πυρήνα της αρχιτεκτονικής της React Native (βλ. Σχ. 2.8). Η γεφύρωση αυτή των δύο παραπάνω τμημάτων, αν και υλοποιημένη σε διαφορετικές τεχνολογίες, αποτελεί ουσιαστικά την βασική έννοια που προσφέρει τη δυνατότητα της αμφίδρομης και ασύγχρονης επικοινωνίας αυτών [44]. Η επικοινωνία αυτή επιτυγχάνεται με τη χρήση δια-

Σχήμα 2.8: Η React Native χρησιμοποιεί τα ίδια native UIs, όπως η Java για Andorid και η Swift για iOS

λειτουργικών γλωσσών (interoperable languages) όπως XML, JSON κλπ. Περισσότερα για αυτές θα δούμε σε παρακάτω ενότητα (βλ. ενότητα 2.4.3).

2.4 Πρόσθετες Τεχνολογικές Έννοιες

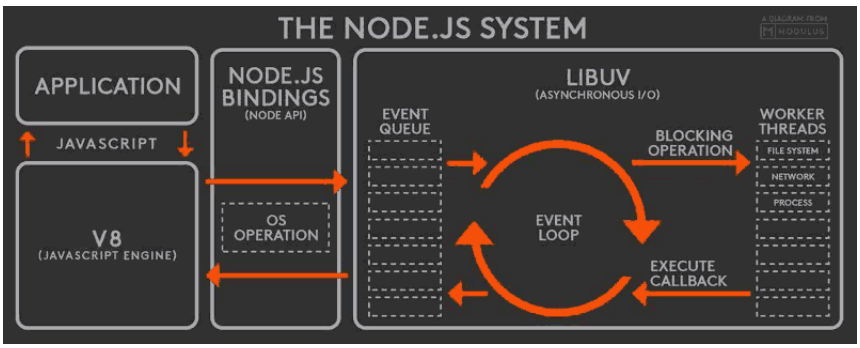
2.4.1 Node.js

Το Node.js (συχνά αναφερόμενο και ως Node) είναι ένα περιβάλλον ανοικτού κώδικα με την υποστήριξη από πολλές πλατφόρμες. Το μεγαλύτερο μέρος του είναι γραμμένο στη γλώσσα Javascript. Το Node.js χρησιμοποιεί τεχνικές εισόδου/εξόδου χωρίς αποκλεισμό (non-blocking I/O), με αποτέλεσμα να παραμένει εύχρηστο και ελαφρύ όταν έρχεται αντιμέτωπο με εφαρμογές έντονου φορτίου πραγματικού χρόνου, με υψηλές απαιτήσεις μεταφοράς δεδομένων [45]. Το non-blocking μέρος σημαίνει ότι κάποιες εντολές, όπως ανάγνωση από αρχείο, μεταβιβάζονται και εκτελούνται παράλληλα και χρησιμοποιούν μεθόδους ανάκλησης (callback functions) για να ολοκληρώσουν το μήνυμα. Σε μία γλώσσα όπως η *PHP* για παράδειγμα, μία νέα εντολή εκτελείται μόνο αφού ολοκληρωθεί η εκτέλεση της προηγούμενης εντολής [46]. Η τεχνολογία Node δεν είναι σχεδιασμένη να παρέχει σύνθετη υπολογιστική ισχύ σε πολύπλοκα συστήματα, δεδομένου ότι αυτό θα αναιρούσε όλα τα πλεονεκτήματά της.

Η Node είναι μονονηματική. Συνεπώς, υπολογισμοί που καταναλώνουν πόρους μπορούν να φράξουν το νήμα και να αποτρέψουν την επεξεργασία άλλων αιτημάτων. Χρησιμοποιείται καλύτερα για τη δημιουργία γρήγορων, κλιμακούμενων δικτυακών εφαρμογών, δεδομένου ότι το μεγαλύτερο πλεονέκτημά της είναι αποτελεσματική διαχείριση μαζικών ταυτόχρονων συνδέσεων. Ένα άλλο τεράστιο πλεονέκτημα της Node.js είναι η ενσωματωμένη διαχείριση πακέτων χρησιμοποιώντας το εργαλείο *NPM*. Το NPM παρέχει στον προγραμματιστή ένα ευρύ σύνολο επαναχρησιμοποιήσιμων προγραμματιστικών εργαλείων και πακέτων, τα οποία εγκαθίστανται εύκολα και χρησιμοποιούνται κατά την ανάπτυξη εφαρμογών. Είναι ένα από

τα μεγαλύτερα σύνολα βιβλιοθηκών ανοικτού κώδικα, με άριστη υποστήριξη και οργανωμένη βιβλιογραφία για κάθε πακέτο ανοιχτού κώδικα που διαθέτει [47].

Ο τρόπος λειτουργίας του Node.js επιδεικνύεται στο Σχ. 2.9. Η όλη δράση του Node.js συγκεντρώνεται στον βρόχο συμβάντων (βλ. ενότητα 2.3.3.1). Η αποστολή ενός αιτήματος μέσα στην εφαρμογή πυροδοτεί την έναρξη ενός συμβάντος, το οποίο τοποθετείται στην ουρά συμβάντων (event queue). Ο βρόχος συμβάντων λαμβάνει το συμβάν και το επεξεργάζεται. Σε περίπτωση που είναι αδύνατη η ολοκλήρωση ενός συμβάντος (π.χ. ανάγνωση εικόνων), τότε η επεξεργασία του ανατίθεται σε ένα από τα νήματα εργασίας (worker threads) από τη '*δεξαμενή νημάτων-εργαζομένων*' (worker thread pool). Όταν ένα νήμα εργασίας ολοκληρώση την επεξεργασία ενός συμβάντος, επιστρέφει το μήνυμα απόκρισης χρησιμοποιώντας μια μέθοδο ανάκλησης η οποία διαβιβάστηκε σε αυτήν μέσω του συμβάντος. Το αποτέλεσμα μεταβιβάζεται στην εφαρμογή μέσω μιας μορφής Node δεσμεύσεων [46].



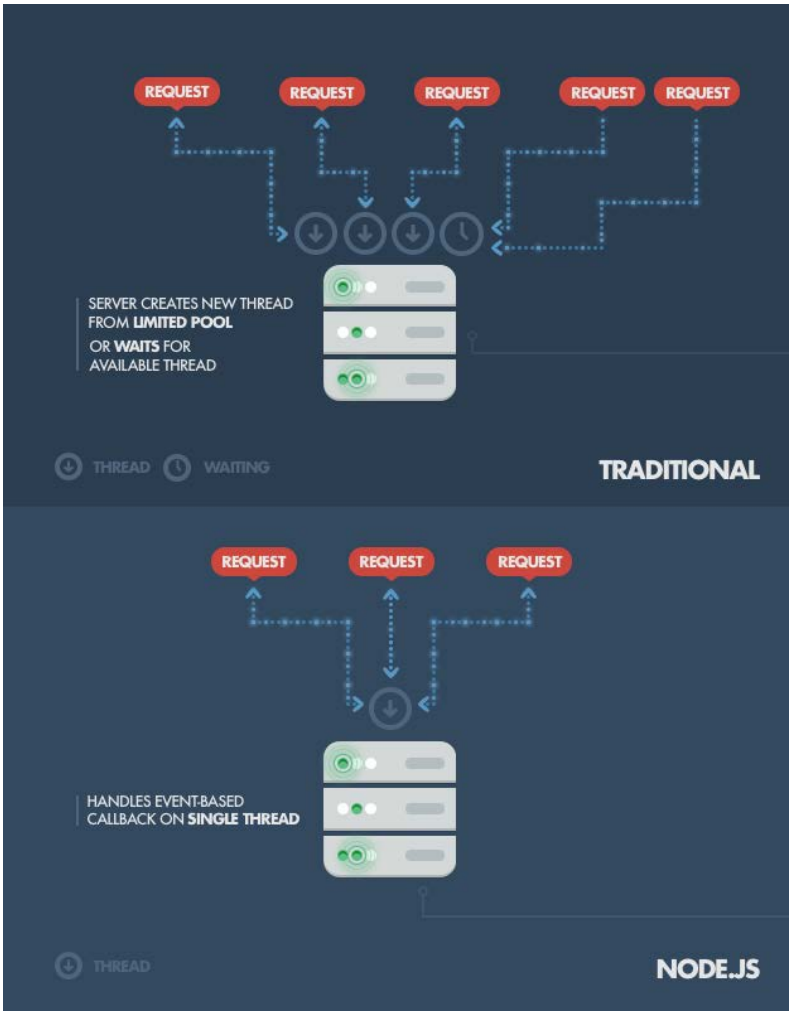
Σχήμα 2.9: Η αρχή λειτουργίας του συστήματος Node.js (Πηγή: [48])

Στη συνέχεια παρατίθενται τα κύρια πλεονεκτήματα της λειτουργίας ενός εξυπηρετητή Node.js:

- Ο Node.js server μπορεί να επεξεργάζεται ασύγχρονα εισερχόμενα αιτήματα. Επομένως ο χρόνος απόκρισης είναι μικρότερος.
- Η Node.js οδηγείται από συμβάντα (event-driven) και χρησιμοποιεί non-blocking I/O, μέσω μεθόδων επανάκλησης, του βρόχου συμβάντων και της ουράς συμβάντων. Το μοντέλο αυτό συνάδει με εκείνο των web servers.
- Δεδομένου ότι βασίζεται στη JavaScript είναι σε θέση να αξιοποιήσει το μηχανισμό V8 Javascript Engine. Ο μηχανισμός V8 Javascript Engine μεταγλωττίζει τον κώδικα από JS σε κώδικα μηχανής που αποφέρει καλύτερες επιδόσεις σε σύγκριση με συνήθεις τεχνικές όπως η μετάφραση (interpreting).
- Η Node.js είναι κατάλληλη για εφαρμογές πραγματικού χρόνου που «τρέχουν» σε πολλές συσκευές. Είναι ιδανικό για γρήγορη παράδοση δεδομένων σε πολλούς αιτούντες ταυτοχρόνως.

Στο Σχ. 2.10 γίνεται μια σύγκριση όσον αφορά στην επεξεργασία ενός αιτήματος μεταξύ ενός node server και ενός συνηθισμένου web server (όπως ο Apache web server). Στην

περίπτωση ενός web server, σε κάθε αίτημα αντιστοιχίζεται μια νηματική διεργασία από μια δεξαμενή περιοσμένου αριθμού νημάτων. Συνεπώς, αν δεν υπάρχει διαθεσιμότητα το αίτημα τίθεται σε αναμονή έως ότου ολοκληρωθεί κάποια από τις υπόλοιπες διεργασίες. Η χρήση πολυνηματικών διεργασιών καταλαμβάνει αρκετό ποσοστό μνήμης στη RAM με αποτέλεσμα την επιβράδυνση του συστήματος και άλλοτε ακόμη και τον απότομο τερματισμό αυτού. Η συμπεριφορά αυτή δεν είναι επιθυμητή. Για το λόγο αυτό, η Node.js λειτουργεί με ένα μόνο νήμα χρησιμοποιώντας non-blocking I/O κλήσεις, οι οποίες επιτρέπουν στο σύστημα την υποστήριξη πολλών ταυτόχρονων συνδέσεων.Για να γίνει αντιληπτό, ας υποθέσουμε ότι ο server έχει στη διάθεσή του μια μνήμη RAM χωρητικότητας 8GB. Επίσης, έστω ότι κάθε διεργασία καταλαμβάνει 2MB της μνήμης. Αυτό σημαίνει πως ένας παραδοσιακός web server μπορεί να επεξεργαστεί περίπου 4000 ταυτόχρονα αιτήματα. Από την άλλη πλευρά, ένας Node server δυνητικά θα μπορούσε να επεξεργαστεί 1M ταυτόχρονα αιτήματα, χάρις την ιδιότητα επεκτασιμότητάς του [45].



Σχήμα 2.10: Σύγκριση ανάμεσα σε έναν Node.js και έναν κλασσικό server

Εκτός από τα παραπάνω πλεονεκτήματα, ένα σημαντικό χαρακτηριστικό που καθιστά την Node.js μία από τις κορυφαίες σύγχρονες τεχνολογίες είναι η υλοποίησή της σε JavaScript.

Το γεγονός αυτό συνεισφέρει τα μέγιστα στην ευκολία εκμάθησής της από τους προγραμματιστές που είναι ήδη εξοικειωμένοι με τη χρήση της JS. Η προϋπόθεση της κοινής γλώσσας παίζει μείζονα ρόλο στην ανάπτυξη ολοκληρωμένων εφαρμογών (full stack development). Έτσι, ένας προγραμματιστής έχει τη δυνατότητα να υλοποιήσει τον client αλλά και τον server σε μία γλώσσα (πράγμα που παλιότερα δεν ίσχυε).

2.4.2 Representational State Transfer - REST APIs

Το REST είναι μια μορφή αρχιτεκτονικής για τη δημιουργία δικτυακών εφαρμογών που εισήχθη το 2000 απο τον Roy Fielding [50]. Δημιουργήθηκε για να υποστηρίξει τη διαλειτουργικότητα μεταξύ όλων των υπολογιστών του διαδικτύου. Ένας REST Server επιτρέπει την πρόσβαση πόρων μέσω κάποιου αναγνωριστικού (πχ. URL) και ο REST Client απλώς λαμβάνει και τροποποιεί τα δεδομένα αυτά. Οι πόροι αυτοί αναπαρίστανται με διάφορες μορφές εκ των οποίων οι πιο διαδεδομένες είναι το κείμενο, JSON objects, XML [49].

Η client–server επικοινωνία επιτυγχάνεται μέσω αιτημάτων με ροή από τον client στον server. Τα αιτήματα αυτά εκκινούν από τον client με ένα αναγνωριστικό (πχ. URL) που αναδεικνύει τη φύση του αιτήματος, το οποίο συμβαδίζει με το endpoint που έχει δημιουργηθεί από την πλευρά του server. Ο server διαθέτει δρομολογητές (routers) που αντιστοιχούν σε κάθε endpoint. Κάθε router έχει συγκεκριμένο ρόλο που εξυπηρετεί το αίτημα του client. Τέλος, ο server επιστρέφει το φορτίο (response payload) που περιέχει τα ζητούμενα δεδομένα στον client. Οι διεπαφές που βασίζονται σε αυτή την τεχνική ονομάζονται RESTful API [51]. Οι βασικότερες από τις διαθέσιμες μέθοδους με τις οποίες μπορεί να επιτευχθεί η παραπάνω διαδικασία είναι οι εξής:

- ***GET*** – λήψη δεδομένων από ένα endpoint
- ***POST*** – αποστολή δεδομένων για αποθήκευση στο προοριζόμενο endpoint
- ***PUT*** – επεξεργασία αποθηκευμένων δεδομένων στο προοριζόμενο endpoint
- ***DELETE*** – διαγραφή δεδομένων από το προοριζόμενο endpoint

Ακολουθεί επεξηγηματικός πίνακας με τις μεθόδους και την βασική αρχή λειτουργίας τους.

HTTP Methods				
URI	***GET***	***POST***	***PUT***	***DELETE***
Collection resource example *https://api.example.com/collection/*	*Retrieve* the URIs of the member resources of the collection resource in the response body.	*Create* a member resource in the collection resource using the instructions in the request body. The URI of the created member resource is automatically assigned and returned in the response *Location* header field.	*Replace* all the representations of the member resources of the collection resource with the representation in the request body, or create the collection resource if it does not exist.	*Delete* all the representations of the member resources of the collection resource.
Member resource example *https://api.example.com/collection/item1*	*Retrieve* a representation of the member resource in the response body.	*Create* a member resource in the member resource using the instructions in the request body. The URI of the created member resource is automatically assigned and returned in the response *Location* header field.	*Replace* all the representations of the member resource, or create the member resource if it does not exist, with the representation in the request body.	*Delete* all the representations of the member resource.

Πίνακας 2.1: Συνηθισμένες μέθοδοι HTTP αιτημάτων σε RESTful APIs

2.4.3 JavaScript Object Notation

Το JSON είναι μια απλή μορφή αναπαράστασης δεδομένων ως ζεύγη ιδιοτήτων/τιμών (key/value). Η σύνταξή του αποτελεί υποσύνολο της γλώσσας JavaScript, όμως κώδικας για δημιουργία και ανάλυση δομών JSON υποστηρίζεται από μεγάλο πλήθος γλωσσών προγραμματισμού. Χρησιμοποιείται ευρέως στην ασύγχρονη επικοινωνία μεταξύ browser και server και τείνει να αντικαταστήσει τη γλώσσα σήμανσης XML. Ένα αντικείμενο JSON περικλείεται σε αγκύλες { } και αποτελείται από ένα σύνολο ζευγών ιδιότητας/τιμής. Σε κάθε τέτοιο ζεύγος, η ιδιότητα είναι πάντα τύπου String, ενώ η τιμή ανήκει σε έναν από τους τύπους δεδομένων που υποστηρίζει το JSON. Οι υποστηριζόμενοι τύποι δεδομένων είναι οι αριθμοί (δε γίνεται διάκριση μεταξύ ακεραίων και δεκαδικών), Strings, Boolean, Arrays, Objects (συλλογή από ζεύγη ιδιοτήτων/τιμών) και το null, που αντιπροσωπεύει την κενή τιμή.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Παράθεση 2.6: Αναπαράσταση δεδομένων σε μορφή JSON

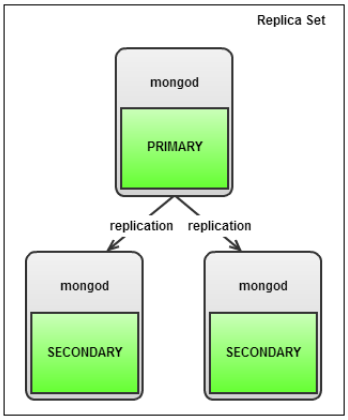
2.4.4 Βάση Δεδομένων MongoDB

Η MongoDB είναι μια μη-σχετικιστική (non-relational) βάση δεδομένων με υποστήριξη σε πολλές πλατφόρμες, που χρησιμοποιεί έγγραφα για την καταγραφή δεδομένων (document-oriented). Η αποθήκευση των δεδομένων γίνεται με ομαδοποίηση σε '*συλλογές*' (collections). Κάθε collection ακολουθεί συγκεκριμένη μοντελοποίηση ως προς την διαρρύθμιση των δεδομένων. Το μοντέλο αυτό (schema) είναι υπεύθυνο για τον προσδιορισμό της δομής των στοιχείων που θα αποθηκευτούν σε κάθε collection της βάσης και έχει μορφή JSON κειμένου.

Τα κυριότερα χαρακτηριστικά της MongoDB συνοψίζονται παρακάτω:

- **Ad Hoc queries** – Η MongoDB υποστηρίζει διάφορες μορφές αναζήτησης δεδομένων εντός της βάσης [52], όπως αναζήτηση με κριτήριο ένα συγκεκριμένο πεδίο (field search), με τήρηση συνθήκης (range search), ή με βάση μια σειρά χαρακτήρων που αποτελούν ένα μοτίβο αναζήτησης (regular expression search). Το αποτέλεσμα που επιστρέφεται από μια τέτοια αναζήτηση μπορεί να είναι ένα συγκεκριμένο πεδίο ενός εγγράφου, να περιέχει μια ομάδα πεδίων ή να αποτελείται από ένα σύνολο εγγράφων δεδομένου μεγέθους.

- **Replication** – Η MongoDB παρέχει υψηλή διαθεσιμότητα με ομάδες αναπαραγωγής (replica sets) [53]. Ένα replica set αποτελείται από δύο ή περισσότερα αντίγραφα των δεδομένων. Κάθε μέλος ομάδας αναπαραγωγής μπορεί να ενεργεί στο ρόλο πρωτογενούς ή δευτερογενούς αντιγράφου ανά πάσα στιγμή. Όλες οι εγγραφές και οι αναγνώσεις γίνονται στο κύριο αντίγραφο από προεπιλογή. Τα δευτερεύοντα αντίγραφα διατηρούν ένα αντίγραφο των δεδομένων του πρωτογενούς χρησιμοποιώντας την ενσωματωμένη αναπαραγωγή. Όταν ένα πρωτότυπο αντίγραφο αποτύχει, το σύνολο αντιτύπων διεξάγει αυτόματα μια εκλογική διαδικασία για να προσδιορίσει ποιο δευτερεύον πρέπει να γίνει το κύριο. Τα δευτερεύοντα τμήματα μπορούν προαιρετικά να προβάλλουν λειτουργίες ανάγνωσης (βλ. Σχ. 2.11).



Σχήμα 2.11: Δομή ομάδας αντιγραφής

- **Indexing** – Τα πεδία σε μια MongoDB βάση δεδομένων ακολουθούν μηδενική, πρωτοβάθμια ή δευτεροβάθμια δεικτοδότηση. Ο δείκτης αυτός αφορά την μοναδικότητα ή μη ενός πεδίου στη βάση.

- **Load balancing** – Η MongoDB κλιμακώνεται οριζοντίως με τη χρήση της ιδιότητας τεμαχιοποίησης (sharding). Ο χρήστης επιλέγει ένα '*κλειδί κοπής*' (shard key) το οποίο καθορίζει τον τρόπο κατανομής των δεδομένων μιας συλλογής. Τα δεδομένα χωρίζονται σε εύρη τιμών (με βάση το κλειδί αυτό) και κατανέμονται σε πολλαπλά shards (ένα shard είναι το κύριο έγγραφο και έχει ένα ή περισσότερα αντίγραφα). Εναλλακτικά, το κλειδί shard μπορεί να κρυπτογραφηθεί μέσω hashing για να χαρτογραφηθεί σε ένα shard – επιτρέποντας μια ομοιόμορφη κατανομή δεδομένων. Η MongoDB μπορεί να τρέξει σε πολλούς διακομιστές, εξισορροπώντας το φορτίο ή να αντιγράψει τα δεδομένα για να διατηρήσει το σύστημα σε λειτουργία και σε περίπτωση αποτυχίας υλικού [54].

- **File Storage** – Η MongoDB μπορεί να χρησιμοποιηθεί ως σύστημα αρχείων, το οποίο ονομάζεται GridFS [55], με δυνατότητες εξισορρόπησης φορτίου και αναπαραγωγής δεδομένων σε πολλαπλές μηχανές αποθήκευσης αρχείων. Το σύστημα GridFS διαιρεί ένα αρχείο σε τμήματα, ή κομμάτια, και αποθηκεύει κάθε ένα από αυτά τα κομμάτια ως ξεχωριστό έγγραφο [56].

- **Aggregation** – Η MongoDB διαθέτει τρεις μεθόδους συνάθροισης: του αγωγού συσσωμάτωσης (aggregation pipeline), της άθροισης μέσω χαρτογράφησης (map-reduce function) και τις μονού σκοπού μεθόδους συνάθροισης (single-purpose aggregation methods) [57, 58].

Συμπερασματικά, η MongoDB είναι σχεδιασμένη για να απλοποιήσει τη διαδικασία αποθήκευσης και αναζήτης δεδομένων. Δεν έχει συνθετες εντολές, αντιθέτως κάθε διαδικασία αναζήτησης είναι σαφής και σύντομη. Ο χρήστης παρέχει τις προϋποθέσεις ή τη συνθήκη που επιθυμεί και η MongoDB επιστρέφει το αποτέλεσμα από τη βάση. Στην περίπτωση εύρεσης ενός εγγράφου κάνει συγκεκριμένη αναζήτηση, ενώ στην περίπτωση εύρεσης ενός πλήθους εγγράφων κάνει ομαδική αναζήτηση με σελιδοποίηση (pagination). Η MongoDB είναι ευέλικτη και εύκολη στη χρήση, χάρις στην JSON μορφοποίηση των δεδομένων, έναντι της χρήσης πινάκων (πχ. όπως στην SQL).

2.4.5 Ταυτοποίηση Χρηστών (Open Authentication)

Η ύπαρξη πολλών χρηστών σε μια εφαρμογή γεννά δύο βασικά προβλήματα. Το πρώτο συνίσταται από την ανάγκη διαχωρισμού των χρηστών μεταξύ τους, ενώ το δεύτερο έχει να κάνει με την παροχή προσωποποιημένων υπηρεσιών σε κάθε χρήστη. Σε αυτή την ενότητα γίνεται λόγος για τον τρόπο με τον οποίο επιλύονται τα παραπάνω προβλήματα.

Οι εφαρμογές που δίνουν στους χρήστες τη δυνατότητα δημιουργίας προσωπικού λογαριασμού διαθέτουν ένα σύστημα επιβεβαίωσης και πρόσβασης εντός των υπηρεσιών τους. Το σύστημα αυτό ακολουθεί ένα πρωτόκολλο ταυτοποίησης γνωστό ως OAuth.

2.4.5.1 OAuth

Το *OAuth* είναι ένα ανοιχτό πρότυπο το οποίο αναλαμβάνει την διαχείριση πρόσβασης στις υπηρεσίες μιας εφαρμογής από τους χρήστες της. Ο μηχανισμός αυτός χρησιμοποιείται

σε όλες τις σημερινές εφαρμογές, όπως *facebook, Microsoft, Google, Twitter, Amazon* κλπ. Γενικά, το OAuth παρέχει στους πελάτες μια '*ασφαλή εξουσιοδοτημένη πρόσβαση*' σε πόρους ενός server εξ' ονόματος του ιδιοκτήτη των πόρων αυτών. Καθορίζει μια διαδικασία εξουσιοδότησης πρόσβασης στους πόρους αυτούς σε τρίτους, χωρίς να μοιράζονται τα διαπιστευτήρια τους. Σχεδιασμένο ειδικά για να λειτουργεί με το πρωτόκολλο HTTP, το OAuth ουσιαστικά επιτρέπει την έκδοση πιστοποιητικών πρόσβασης σε πελάτες τρίτων μέσω ενός server εξουσιοδότησης, με την έγκριση του ιδιοκτήτη πόρων. Το τρίτο μέρος χρησιμοποιεί το διακριτικό πρόσβασης για πρόσβαση στους προστατευόμενους πόρους που φιλοξενεί ο server πόρων.

2.4.5.2 Access Token

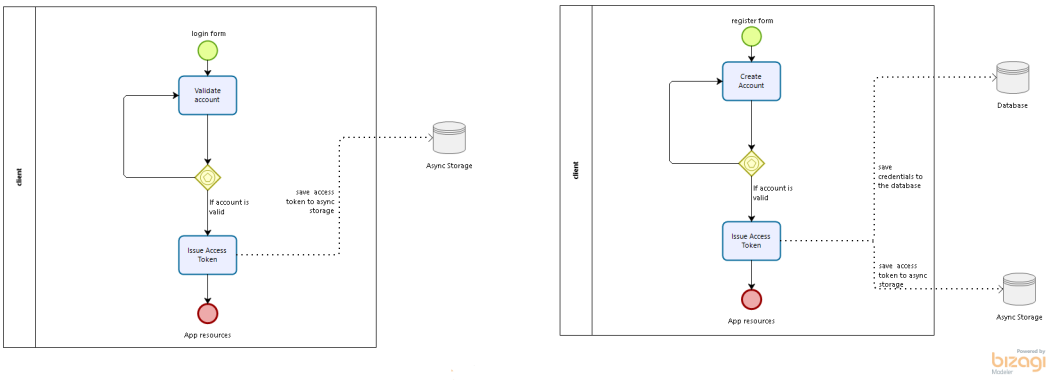
Το σύμβολο πιστοποίησης (Access Token) είναι ένα είδος πιστοποιητικού, το οποίο ένας πελάτης μπορεί να χρησιμοποιήσει για να αποκτήσει πρόσβαση σε προστατευμένους πόρους. Συγκεκριμένα είναι μία συμβολοσειρά χωρίς κάποια σημασία σε κανέναν εκτός από τον server. Πέραν του ότι είναι μικρά σε μέγεθος και δεν προσθέτουν καθυστερήσεις, εμποδίζουν να χρησιμοποιούνται τα διαπεστευτήρια του ιδιοκτήτη (username, email, password) συνεχώς και έτσι αν πέσει σε λάθος χέρια δεν τα αποκτά ο κακόβουλος χρήστης. Επιπλέον έχουν περιορισμένη πρόσβαση και συνήθως έχουν ημερομηνία λήξης, ώστε όταν αυτή παρέλθει, το σύμβολο αυτό παύει να ισχύει και δεν μπορεί να ξαναχρησιμοποιηθεί. Τότε, πρέπει να γίνει αίτηση για ανανέωση της ημερομηνίας (Refresh Token).

Η διαδικασία ταυτοποίησης και έκδοσης access token διακρίνεται σε τρία στάδια. Όταν ο χρήστης εισέρχεται για πρώτη φορά στην εφαρμογή, καλείται να δημιουργήσει λογαριασμό. Με αυτό τον τρόπο, ο χρήστης αποκτά πρόσβαση στους προστατευόμενους πόρους, καθώς επίσης και στις εξατομικευμένες εμπειρίες της εφαρμογής. Η είσοδος του χρήστη στην εφαρμογή γίνεται μέσω της συμπλήρωσης μιας φόρμας εγγραφής (register/signup form). Η διαδικασία αυτή αποτελεί το πρώτο στάδιο. Αφού ο χρήστης χορηγήσει άδεια διαχείρισης των διαπιστευτηρίων του κάνοντας register στην εφαρμογή, τότε μεταφέρεται αυτόματα εντός των υπηρεσιών της εφαρμογής. Κατά τη διάρεκεια της μετάβασης, ο server είναι υπεύθυνος για την αποθήκευση των διαπιστευτηρίων του χρήστη και την ανταλλαγή (handshake) τους με ένα μοναδικό access token που αντιστοιχεί σε αυτόν τον χρήστη που μόλις πραγματοποίησε είσοδο στην εφαρμογή. Με άλλα λόγια, το access token συμβολίζει το '*κλειδί*' με το οποίο ο χρήστης '*ανοίγει*' την εφαρμογή. Η ανταλλαγή αυτή αποτελεί το δεύτερο στάδιο.

Η διαδικασία αυτή είναι μονόδρομη, καθώς το πρώτο σταδιο αποτελεί απαραίτητη και αναγκαία συνθήκη για τη μετάβαση στο δεύτερο στάδιο. Ο χρήστης δεν μπορεί να παρακάμψει το πρώτο στάδιο καθώς η έκδοση access token είναι υποχρεωτική για όλους τους χρήστες που επιθυμούν να καταναλώσουν τις υπηρεσίες τις εφαρμογής. Ωστόσο, μετά την έκδοση access token ο χρήστης δεν χρειάζεται να επαναλάβει τη διαδικασία ταυτοποίησης. Εφόσον δεν πραγματοποιήσει έξοδο (logout) από την εφαρμογή, έχει την δυνατότητα να «κλείσει» και να «ανοίξει» την εφαρμογή χωρίς την απαίτηση των διαπιστευτηρίων του. Το κλειδί αποθηκεύεται στη μνήμη της συσκευής για μελλοντική χρήση. Αυτό είναι και το νόημα της όλης διαδικασίας. Σε αυτή την περίπτωση, ο χρήστης κάνει είσοδο στην εφαρμογή παρακάμπτοντας

το πρώτο στάδιο ταυτοποίησης, το οποίο γίνεται αυτόματα χάρις στο αποθηκευμένο κλειδί που είναι διαθέσιμο ασύγχρονα στην εφαρμογή.

Για όσο διάστημα ο χρήστης έχει στη διάθεσή του κλειδί, μπορεί να είσερχεται στην εφαρμογή ή να εξέρχεται από αυτή κατ΄ επανάληψη. Αυτό οφείλεται στο γεγονός ότι ο χρήστης εξακολουθεί να είναι συνδεδεμένος. Ωστόσο, εάν για κάποιο λόγο ο χρήστης αποφασίσει να αποσυνδεθεί από την εφαρμογή κάνοντας logout, τότε το access token αφαιρείται και καταστρέφεται και ο χρήστης επαναφέρεται αυτόματα στην σελίδα εισόδου της εφαρμογής. Αυτή η διαδικασία αποτελεί το τρίτο και τελευταίο στάδιο, με το οποίο ολοκληρώνεται ο κύκλος ταυτοποίησης ενός χρήστη. Προκειμένου να αποκτηθεί εκ νέου πρόσβαση, πρέπει να επαναληφθούν τα πρώτα δύο στάδια από την αρχή. Η μόνη διαφορά στην όλη διαδικασία είναι πως πλέον ο χρήστης είναι καταχωρημένος στη βάση της εφαρμογής και, επομένως δεν χρειάζεται να κάνει εκ νέου εγγραφή. Αρκεί να παρέχει τα διαπιστευτήριά του στην φόρμα εισόδου (login/signin form) και στην περίπτωση επιτυχούς διασταύρωσης των στοιχείων, να συνδεθεί στο λογαριασμό του και να μεταφερθεί εντός της εφαρμογής.



Σχήμα 2.12: Διαδικασία έκδοσης access token

2.4.5.3 JSON Web Token

Το JWT είναι μια μορφή access token το οποίο εκδίδεται στην πλευρά του server κατά την ταυτοποίηση ενός χρήστη. Κατά την επιτυχή είσοδο του χρήστη στην εφαρμογή, το JWT αποθηκεύεται τοπικά στη μνήμη της συσκευής. Κάθε φορά που ο χρήστης θέλει να αποκτήσει πρόσβαση σε προστατευόμενους πόρους εντός της εφαρμογής, ανταλλάσσει το JWT με τον προστατευόμενο πόρο μέσω μιας χειραψίας (handshake). Η χειραψία είναι στην ουσία ένα αίτημα το οποίο αποστέλλεται στον server και περιέχει, μεταξύ άλλων, και μια επικεφαλίδα εξουσιοδότησης (authorization header). Το περιεχόμενο της επικεφαλίδας μοιάζει με το ακόλουθο:

```
1 Authorization: Bearer eyJhbGci...<snip>...yu5CSpyHI
```

Αυτός είναι ένας μηχανισμός ελέγχου ταυτότητας χωρίς κατάσταση, αφού η κατάσταση του χρήστη δεν αποθηκεύεται ποτέ στη μνήμη του server. Οι προστατευμένες διαδρομές (routes) του server θα ελέγχουν την ύπαρξη έγκυρου JWT στην επικεφαλίδα εξουσιοδότησης και,

αν υπάρχει, ο χρήστης θα έχει πρόσβαση στους προστατευμένους πόρους. Καθώς τα JWTs είναι αυτοτελή, υπάρχουν όλες οι απαραίτητες πληροφορίες, μειώνοντας την ανάγκη συχνών προσπελάσεων της βάσης δεδομένων [59].

Τα βασικά πεδία που διαθέτει ένα JSON Web Token φαίνονται στον κάτωθι πίνακα:

Standard JWT Fields		
code	***name***	***description***
iss	Issuer	Identifies principal that issued the JWT.
sub	Subject	Identifies the subject of the JWT.
aud	Audience	Identifies the recipients that the JWT is intended for. Each principal intended to process the JWT **must** identify itself with a value in the audience claim. If the principal processing the claim does not identify itself with a value in the *aud* claim when this claim is present, then the JWT **must** be rejected.
exp	Expiration Time	Identifies the expiration time on and after which the JWT **must not** be accepted for processing. The value must be a *NumericDate*: either an integer or decimal, representing seconds past *1970-01-01 00:00:00Z.*
nbf	Not Before	Identifies the time on which the JWT will start to be accepted for processing. The value must be a *NumericDate.*
iat	Issued At	Identifies the time at which the JWT was issued. The value must be a *NumericDate.*
jti	JWT ID	Case sensitive unique identifier of the token even among different issuers.

Πίνακας 2.2: Δικαιώματα που μπορούν να αποδωθούν σε ένα JSON Web Token

2.4.5.4 Passport.js - Passport Strategies

Για την ταυτοποίηση των χρηστών στην εφαρμογή χρησιμοποιήθηκε το λογισμικό Passport.js. Το Passport είναι ένα ενδιάμεσο λογισμικό το οποίο υλοποιείται σε JS και υποστηρίζεται από τη Node.js. Είναι υπεύθυνο για την διαχείριση των χρηστών και την προστασία των πόρων της εφαρμογής. Συγκεκριμένα, πραγματοποιεί αναζήτηση στη βάση ενός χρήστη που επιχειρεί είσοδο στην εφαρμογή και σε περίπτωση επιτυχούς εύρεσης, εκδίδει token.

Η προστασία των πόρων της εφαρμογής γίνεται με την υλοποίηση στατηγικών πακέτων (Strategies) που διατίθενται από κατασκευής εντός της τεχνολογίας του Passport. Στην εφαρμογή έγινε χρήση δύο εξ' αυτών:

- **JWT Strategy**
- **Local Strategy**

Η JWT Strategy αναλαμβάνει την προστασία των πόρων της εφαρμογής από μη εγγεγραμμένους χρήστες. Αυτό επιτυγχάνεται με τον έλεγχο κάθε φορά που γίνεται αίτημα της επικεφαλίδας εξουσιοδότητης. Εάν αυτή φέρει κάποιο token τότε επιτρέπεται η πρόσβαση στους ζητούμενους πόρους. Αυτή η διαδικασία αποτελεί το αντίστροφο της χειραψίας που αναφέρθηκε νωρίτερα μεταξύ client-server.

Η Local Strategy είναι υπεύθυνη για τη διαδικασία εγγραφής ή σύνδεσης των χρηστών στην εφαρμογή. Με την υποβολή της αίτησης για δημιουργία λογαριασμού στην εφαρμογή από ένα νέο χρήστη, τα στοιχεία αποστέλλονται προς αποθήκευση στη βάση. Στην περίπτωση που η διαδικασία εγγραφής ολοκληρωθεί επιτυχώς (ο συνδυασμός των διαπιστευτηρίων του χρήστη πρέπει να είναι μοναδικός), τότε το Passport εκδίδει token για το νέο χρήστη και εκείνος αποκτά πρόσβαση στους πόρους της εφαρμογής. Αντιστοίχως, όταν ένας ήδη υπάρχων χρήστης κάνει αίτηση για σύνδεση στο λογαριασμό του, τότε γίνεται ο κατάλληλος έλεγχος στον συνδυασμό των διαπιστευτηρίων του, και σε περίπτωση επιτυχίας εκδίδεται νέο token για τον χρήστη αυτό.

```
passport.use(new LocalStrategy(
  function(username, password, done) {
    User.findOne({ username: username }, function (err, user) {
      if (err) { return done(err); }
      if (!user) { return done(null, false); }
      if (!user.verifyPassword(password)) { return done(null, false); }
      return done(null, user);
    });
  }
));
```

Παράθεση 2.7: Βασικό boilerplate της Local Strategy στην εφαρμογή

Κεφάλαιο 3

Ανάλυση Απαιτήσεων Συστήματος

Η εφαρμογή που πραγματεύεται η παρούσα διπλωματική εργασία προορίζεται για χρήση πάνω σε πλατφόρμες κινητών συσκευών που είναι συνδεδεμένες σε δίκτυο. Σκοπός της εφαρμογής είναι να βελτιστοποιήσει τη διαδικασία της ενημέρωσης για κοινωνικοπολιτισμικά δρώμενα και να προσφέρει μια ολοκληρωμένη εμπειρία στους χρήστες της. Αυτό σημαίνει ότι ο χρήστης θα μπορεί να δημιουργήσει τον προσωπικό του λογαριασμό και να ενημερώνεται για εκδηλώσεις που λαμβάνουν χώρα αυτή τη στιγμή γύρω του, όποτε το επιθυμεί. Ο χρήστης θα μπορεί να περιηγηθεί στην κεντρική διεπιφάνεια της εφαρμογής με τη βοήθεια μιας διεπαφής χάρτη, να αναρτήσει σχόλια σχετικά με κάποιο γεγονός στην τοποθεσία του και να αλληλεπιδράσει με άλλους χρήστες που βρίσκονται στην περιοχή. Η εφαρμογή θα βοηθάει στο συντονισμό όλων των ατόμων που σχετίζονται με την εκδήλωση, είτε αυτοί είναι οι διοργανωτές, είτε είναι οι συμμετέχοντες.

Το παρόν κεφάλαιο καταπιάνεται με την ανάλυση των απαιτήσεων του συστήματος. Θα γίνει μια εκτενής επεξήγηση κάθε λειτουργικότητας της εφαρμογής και θα παρουσιαστούν οι βασικές αρχές που τις διέπουν.

3.1 Γενική Περιγραφή

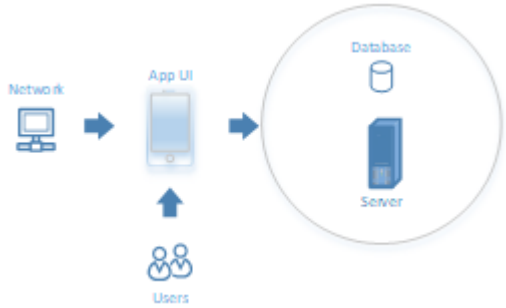
Η εφαρμογή αποτελείται από τρία μέρη: (α) τον client, (β) τον server και (γ) τη βάση δεδομένων. Ο client είναι η διεπιφάνεια όπου δρα ο χρήστης, ενώ ο server αποτελείται από το σύνολο των μεθόδων υπεύθυνων για την εκτέλεση αιτημάτων από τον client (βλ. Σχ. 3.1).

Ο client αποτελείται από ένα δρομολογητή (router) που συνίσταται από τις διεπιφάνειες της εφαρμογής με τις οποίες μπορεί να αλληλεπιδράσει ο χρήστης. Η κεντρική διεπιφάνεια αποτελείται από μια διεπαφή χάρτη (Apple Maps API) για την προβολή σημείων ενδιαφέροντος, τα οποία από εδώ και στο εξής θα αποκαλούνται *hotspots*. Ο router απαρτίζεται ακόμη από τις σελίδες *comments, hotspots, profile, new hotspot*. Καθεμία από αυτες θα αναλυθεί σε επόμενη ενότητα (βλ. ενότητα 3.3.1). Η υλοποίηση και ο έλεγχος του client έγιναν με τη βοήθεια φυσικής συσκευής με λειτουργικό iOS 12, καθώς και ειδικού SDK για αυτό το

σκοπό (περισσότερες λεπτομέρειες θα δωθούν στο Κεφ. 5).

Ο server καθορίζει τις λειτουργίες προς εκτέλεση που αντιστοιχούν στις αιτήσεις που δημιουργεί ο χρήστης μέσω του UI της εφαρμογής. Οι λειτουργίες αυτές συνοπτικά είναι η αποθήκευση των διαπιστευτηρίων και των προσωπικών πληροφοριών του χρήστη, η δημιουργία λογαριασμού (profile), η δημιουργία και επεξεργασία δημοσιεύσεων (create/update hotspot), η δημιουργία σχολίων (create comment), η λήψη και αποθήκευση φωτογραφιών κλπ. Η κατασκευή του server έχει δομή τέτοια ώστε να αποσκοπεί στη συνεργατική δράση του με τον client, με αποδοτικό και αποτελεσματικό τρόπο.

Η εφαρμογή θα διαθέτει επίσης μια βάση δεδομένων για την αποθήκευση των χρηστών, των δημοσιεύσεων, των σχολίων και άλλων στοιχείων που αφορούν το χρήστη, είτε άμεσα (στοιχεία λογαριασμού, δημοσιεύσεις χρήστη, σχολιασμοί χρήστη κλπ), είτε έμεσσα (σχόλια σε αναρτήσεις του χρήστη, σχόλια σε δημοσιεύσεις άλλων χρηστών κλπ.).



Σχήμα 3.1: Περιβάλλον συστήματος εφαρμογής

3.2 Απαιτήσεις Συστήματος

Σε αυτή την ενότητα αναφέρονται συνοπτικά οι απαιτήσεις του συστήματος που εξασφαλίζουν την ολοκληρωμένη λειτουργία της εφαρμογής. Πρόκειται για τις λειτουργικότητες της εφαρμογής οι οποίες προσφέρουν την επιθυμητή εμπειρία στο χρήστη. Στο σύνολό τους, οι λειτουργικότητες θα πρέπει να εξυπηρετούν το σκοπό για τον οποίο αναπτύχθηκε η εφαρμογή (βλ. ενότητα 3.2.1). Επίσης, προκειμένου να εξασφαλιστεί η ολοκληρωμένη και καλή εμπειρία του χρήστη, είναι απαραίτητο να ληφθούν υπόψη και κάποιες μη λειτουργικές απαιτήσεις. Οι μη λειτουργικές απαιτήσεις δεν έχουν να κάνουν με την υλοποίηση της διεπιφάνειας του χρήστη, αλλά με τη γενικότερη λειτουργικότητα της εφαρμογής. Η ανάλυση αυτών γίνεται στην ενότητα 3.2.2.

3.2.1 Λειτουργικές Απαιτήσεις Συστήματος

Η διεπιφάνεια του χρήστη θα πρέπει να φέρει τις παρακάτω λειτουργικότητες:

1. **Εγγραφή χρήστη**
   Ο χρήστης μπορεί να πραγματοποιήσει εγγραφή στην εφαρμογή, με τη συμπλήρωση κατάλληλης φόρμας.

2. **Σύνδεση χρήστη**
   Υπάρχοντες χρήστες μπορούν να πραγματοποιήσει σύνδεση στην εφαρμογή, μέσω της συμπλήρωσης κατάλληλης φόρμας με τα διαπιστευτήριά του.

3. **Ταυτοποίηση χρήστη**
   Η ταυτότητα του χρήστη πιστοποιείται μέσω της έκδοσης token με το OAuth πρωτόκολλο.

4. **Δημοσίευση κατάστασης – *hotspot***
   Ο χρήστης μπορεί να αναρτήσει μια δημοσίευση, σχετική με κάποιο εγγύς γεγονός ή εκδήλωση. Η ανάρτηση θα εμφανίζεται στην διεπιφάνεια χάρτη της κεντρικής σελίδας, στην τοποθεσία του χρήστη. Κατα τη δημιουργία hotspot είναι εφικτές οι ακόλουθες επιλογές:

   - προσθήκη περιγραφής (απαιτείται)
   - προσθήκη φωτογραγίας (προαιρετικά)
   - προσδιορισμός διάρκειας ισχύος της δημοσίευσης (απαιτείται)

5. **Ανάγνωση hotspot άλλων χρηστών**
   Ο χρήστης μπορεί να «ανοίξει» και να διαβάσει hotspots άλλων χρηστών στην περιοχή του ή σε άλλη περιοχή, μετακινόντας το χάρτη.

6. **Δημιουργία Σχολίων**
   Ο χρήστης μπορεί να σχολιάσει σε όλα τα hotspots (δικά του και άλλων χρηστών).

7. **Προβολή προσωπικών hotspots**
   Ο χρήστης μπορεί να επισκεφτεί τη λίστα με όλα τα προσωπικά του hotspots. Η λίστα είναι ταξινομημένη με το πιο πρόσφατο hotspot να φαίνεται πρώτο και το παλαιότερο hotspot να φαίνεται τελευταίο.

8. **Προβολή πρόσθετων πληροφοριών hotspot**
   Ο χρήστης μπορεί να επιβλέπει την κατάσταση όλων των hotspots (δικών του και άλλων χρηστών) μέσω δεικτών μέτρησης views και comments. Ο δείκτης views είναι υπεύθυνος για τον αριθμό νέων αναγνώσεων ενός hotspot, ενώ ο δείκτης comments δείχνει τον αριθμό των σχολίων ενός hotspot.

9. **Επιλογή φίλτρου αναζήτησης**
   Ο χρήστης μπορεί να επιλέξει μια κατηγορία ενδιαφέροντος και να πραγματοποιήσει αναζήτηση στον χάρτη σημείων που ανήκουν σε αυτή. Με το σύρσιμο της οθόνης νέα σημεία εμφανίζονται στο χάρτη.

10. **Αναίρεση φίλτρου αναζήτησης**
    Ο χρήστης μπορεί να καταργήσει τα επιλεγμένο φίλτρο για να επιλέξει νέο, ή να «καθαρίσει» το χάρτη.

11. **Μπάρα Αναζήτησης**
    Ο χρήστης μπορεί να πληκτρολογήσει κάποιο σημείο ενδιαφέροντος (point of interest) και να πραγματοποιήσει συγκεκριμένη αναζήτηση του επιλεγμένου σημείου.

12. **Άνοιγμα κεντρικού μενού πλοήγησης**
    Ο χρήστης μπορεί να πατήσει στο κεντρικό μενού (*FAB*) και να επιλέξει τη σελίδα στην οποία θέλει να μεταβεί. Ο χρήστης μπορεί να πλοηγηθεί στις παρακάτω σελίδες:

    - λίστα προσωπικών hotspots
    - φόρμα δημοσίευσης νέου hotspot
    - προσωπικό λογαριασμό (profile) χρήστη

13. **Προβολή προσωπικών στοιχείων**
    Ο χρήστης μπορεί να μεταβεί στο profile του και να δει τα προσωπικά του στοιχεία.

14. **Επεξεργασία προσωπικών στοιχείων**
    Ο χρήστης μπορεί να επεξεργαστεί τα προσωπικά του στοιχεία και να ανανεώσει το προσωπικό του λογαριασμό μέσω κατάλληλης φόρμας. Μπορεί να ανανεώσει τα διαπιστευτήριά του, την εικόνα profile, το όνομά του, τον τόπο διαμονής, την ημερομηνία γέννησης και το γένος.

15. **Προβολή στατιστικών μετρήσεων**
    Ο χρήστης μπορεί να δει στατιστικά δεδομένα σχετικά με την δράση του στην εφαρμογή, όπως αριθμό hotspots, αριθμό σχολίων, αριθμό ατόμων που επισκέφτηκαν τα hotspots του χρήστη, ποσοστό αγοριών/κοριτσιών, δημοτικότητα κλπ.

16. **Αποσύνδεση**
    Ο χρήστης μπορεί να αποσυνδεθεί από το λογαριασμό του κάνοντας logout (από τη σελίδα του profile του).

3.2.2 Μη Λειτουργικές Απαιτήσεις Συστήματος

Η εφαρμογή θα πληροί και ορισμένες μη λειτουργικές απαιτήσεις για μια ολοκληρωμένη εμπειρία χρήστη. Οι απαιτήσεις αυτές απαριθμούνται στη συνέχεια.

1. **Απόδοση – Performance**
   Ο χρόνος απόκρισης του συστήματος στην εξυπηρέτηση αιτημάτων από το server θα πρέπει να είναι μικρός. Το σύστημα θα πρέπει να μπορεί να ανταποκρίθεί σε μεγάλο αριθμό αιτημάτων.

2. **Ταχύτητα ανάκτησης – Recoverability**
   Το σύστημα θα πρέπει να μπορεί να ανακτήσει την επιθυμητή κατάσταση λειτουργίας γρήγορα, σε περίπτωση αποτυχίας ή απότομου τερματισμού.

3. **Υποστήριξη – Documentation**
   Η εφαρμογή θα πρέπει να συνοδεύεται από έγγραφη τεκμηρίωση. Η τεκμηρίωση θα πρέπει να είναι επεξηγηματική και εκτενής προκειμένου να καλύπτει όλα τα πιθανά ζητήματα που μπορεί να προκύψουν.

4. **Ασφάλεια – Security**
   Η εφαρμογή θα πρέπει να εξασφαλίζει την ασφάλεια των προσωπικών δεδομένων των χρηστών της. Ευαίσθητα δεδομένα όπως κωδικοί ασφαλείας θα πρέπει να κρυπτογραφούνται πριν την αποθήκευσή τους στη βάση, για την αποφυγή κατάχρησης από κακόβουλους χρήστες.

5. **Επεκτασιμότητα – Extensibility/Scalability**
   Η υλοποίηση της εφαρμογής θα πρέπει να στηρίζεται σε τεχνολογίες που επιτρέπουν την μελλοντική επέκτασή της με νέα χαρακτηριστικά και πρόσθετες εφαρμογές. Επιπλέον, η εφαρμογή θα πρέπει να μπορεί να διαχειριστεί μεγάλο αριθμό χρηστών.

3.3 Αρχιτεκτονική Εφαρμογής

Όπως είδαμε στο σχ. 3.1, το περιβάλλον της εφαρμογής διαμορφώνεται από τρεις βασικούς συντελεστές: τον client, τον server και τη βάση δεδομένων. Ο client θα πρέπει να είναι συνδεδεμένος σε δίκτυο προκειμένου να εδραιωθεί η επικοινωνία με τον server. Ο server της εφαρμογής στην τελική φάση θα φιλοξενείται από έναν κεντρικό εξυπηρετητή μέσω μιας διαδικασίας γνωστής ως hosting. Κατά την ανάπτυξη της εφαρμογής, ο εξυπηρετητής ενεργοποιούνταν σε τοπικό δίκτυο. Η διεπικοινωνία μεταξύ client-server είναι υπεύθυνη για την σωστή λειτουγία της εφαρμογής και την διασφάλιση μιας ολοκληρωμένης εμπειρίας από την πλευρά του χρήστη.

Ο client είναι υπεύθυνος για την διαμόρφωση ενός διαδραστικού περιβάλλοντος για τον χρήστη. Το περιβάλλον αυτό αποτελείται από έναν router ο οποίος είναι υπεύθυνος για την πλοήγηση του χρήστη στις διάφορες σελίδες της εφαρμογής. Σε κάθε σελίδα, υπάρχουν διαδραστικά στοιχεία όπως γραφικές διεπιφάνειες, διεπαφές, πλήκτρα ενεργειών και στοιχεία εισόδου και διάφορα άλλα στοιχεία αλληλεπίδρασης, όλα με σκοπό την παροχή μιας πλήρους εμπειρίας στο χρήστη.

Ο server αναλαμβάνει να εξυπηρετήσει τα αιτήματα που δημιουργεί ο χρήστης από την πλευρά του client. Κάθε φορά που ο χρήστης πραγματοποιεί μια ενέργεια που απαιτεί δεδομένα από τον server, τότε δημιουργείται ένα αίτημα το οποίο καλείται να εξυπηρετήσει ο server. Η προβολή κάποιου hotspot, η δημιουργία ενός σχολίου, η επεξεργασία του λογαριασμού του χρήστη, αποτελούν μερικά παραδείγματα τέτοιων αιτημάτων.

Η βάση δεδομένων αποτελεί τον αποθηκευτικό χώρο της εφαρμογής. Όλα τα δεδομένα, από το λογαριασμό του χρήστη μέχρι και τον αριθμό προβολών ενός hotspot, αποθηκεύονται σε συλλογές εντός της βάσης. Έτσι, κάθε φορά που ζητούνται δεδομένα, ο server πραγματοποιεί αναζήτηση κατάλληλης μορφής στη βάση και επιστρέφει τα απαραίτητα δεδομένα στον client.

3.3.1 Υποδομή Client και Σενάρια Χρήσης (*Frontend*)

Στις επόμενες ενότητες ακολουθεί μια επεξηγηματική ανάλυση της αρχιτεκτονικής του παραπάνω συστήματος. Θα παρουσιαστούν οι υποδομές της εφαρμογής και οι οθόνες που τις θεμελιώνουν. Έπειτα, θα αναλυθούν όλα τα πιθανά σενάρια χρήσης (use cases) της εφαρμογής.

3.3.1.1 Οθόνες Εφαρμογής

Η εφαρμογή δομείται σε τέσσερα κύρια μέρη: ταυτοποίηση, κεντρική σελίδα χάρτη, hotspots και profile. Οι αντίστοιχες οθόνες (screens) αυτών είναι:

(α) οθόνη έναρξης (*WelcomeScreen*)

Εκεί μεταφέρεται ο χρήστης την πρώτη φορά που ανοίγει την εφαρμογή, ή όταν αποσυνδεθεί από το λογαριασμό του.

(β) οθόνη ταυτοποίσης χρήστη (*SignUpScreen & LoginScreen*)

Η οθόνη ταυτοποίησης χρήστη αποτελείται από δυο tabs (Register, Login), καθένας εκ των οποίων περιέχει φόρμα κατάλληλης μορφής την οποία καλείται να συμπληρώσει ο χρήστης.

(γ) κεντρική οθόνη χάρτη (*HomeScreen*)

Η κεντρική οθόνη αποτελείται κυρίως από τη διεπαφή χάρτη (Apple Map API). Στον χάρτη φαίνονται όλα τα hotspots και ο χρήστης μπορεί να ανανεώνει το περιεχόμενο, αλλάζοντας τη θέση του χάρτη.

(ε) οθόνη δημιουργίας νέου hotspot (*CreateHotspotScreen*)

Η οθόνη αυτή συνίσταται από μια φόρμα όπου ο χρήστης εισάγει τα στοιχεία του hotspot προς δημιουργία.

(δ) οθόνη επεξεργασίας hotspot (*EditHotspotScreen*)

Εάν ο χρήστης το επιθυμεί, μπορεί να επεξεργαστεί ένα από τα δικά του hotspots εδώ.

(στ) οθόνη προσωπικών hotspot (*HotspotListScreen*)

Εδώ μπορεί να πλοηγηθεί ο χρήστης όταν επιθυμεί να προβάλλει όλα τα hotspots που έχει δημιουργήσει ο ίδιος.

(ζ) οθόνη προσωπικού profile (*ProfileScreen*)

Σε αυτή την οθόνη, ο χρήστης μπορεί να προβάλλει τα στοιχεία του λογαριασμού του.

(η) οθόνη επεξεργασίας profile (*EditProfileScreen*)

Εάν ο χρήστης επιθυμεί να αλλάξει τα στοιχεία του προσωπικού του λογαριασμού ή να επεξεργαστεί κάποια από αυτά, μπορεί να το κάνει σε αυτή τη σελίδα. Η οθόνη αποτελείται από μια φόρμα η οποία περιέχει τα αρχικά στοιχεία του χρήστη πριν από την αλλαγή. Για την ολοκλήρωση της διαδικασίας απαιτείται ο κωδικός του χρήστη για επιβεβαίωση.

(θ) οθόνη στατιστικών δεδομένων (*StatisticsScreen*)

Στατιστικά δεδομένα σχετικά με τη χρήση της εφαρμογής και την αλληλεπίδραση με άλλους χρήστες εμφανίζονται στην οθόνη στατιστικών δεδομένων.

3.3.1.2 Σενάρια Χρήσης Εφαρμογής

Εγγραφή Χρήστη στην Εφαρμογή (*Register*)

1. Ο χρήστης ανοίγει την εφαρμογή και πατάει το πλήκτρο '*Get Started*' στην οθόνη έναρξης (*WelcomeScreen*).

2. Ο χρήστης συμπληρώνει τη φόρμα εγγραφής που βρίσκεται στο tab με την ονομασία register. Στη φόρμα ο χρήστης πρέπει να εισάγει όλες τις απαιτούμενες πληροφορίες. Τα απαιτούμενα πεδία είναι τα εξής:

   - πλήρες όνομα (*fullname*)
   - όνομα χρήστη (*username*)
   - ηλεκτρονική διεύθυνση (*email*)
   - κωδικός (*password*)
   - επαλήθευση κωδικού (*confirm password*)
   - ημερομηνία γέννησης (*birthdate*)
   - πόλη (*city*)
   - γένος χρήστη (*gender*)

   Εάν ο χρήστης το επιθυμεί, μπορεί να επιλέξει μια φωτογραφία profile από τη συλλογή φωτογραφιών του ή να βγάλει καινούρια μέσω της κάμερας της συσκευής.

3. Στην περίπτωση που δεν υπάρχουν σφάλματα κατά την υποβολή της φόρμας εγγραφής, ο χρήστης εγγράφεται στην εφαρμογή με επιτυχία. Η εφαρμογή τον πηγαίνει αυτόματα στην κεντρική σελίδα της διεπαφής χάρτη (*HomeScreen*).

Σύνδεση Χρήστη στην Εφαρμογή (*Login*)

1. Ο χρήστης ανοίγει την εφαρμογή και πατάει το πλήκτρο '*Get Started*' στην οθόνη έναρξης (*WelcomeScreen*).

2. Ο χρήστης συμπλήρώνει τη φόρμα σύνδεσης που βρίσκεται στο tab με την ονομασία login. Στη φόρμα ο χρήστης πρέπει να εισάγει όλες τις απαιτούμενες πληροφορίες. Τα απαιτούμενα πεδία είναι τα εξής:

   - ηλεκτρονική διεύθυνση (*email*)
   - κωδικός πρόσβασης (*password*)

3. Στην περίπτωση που δεν υπάρχουν σφάλματα κατά την υποβολή της φόρμας σύνδεσης, ο χρήστης επαναφέρεται αυτόματα στην αρχική σελίδα (*HomeScreen*).

Δημιουργία Νέου Hotspot

1. Ο χρήστης πατάει πάνω στο FAB στο κάτω δεξία μέρος της οθόνης στην HomeScreen.

2. Ο χρήστης πατάει επάνω στο εικονίδιο με το σύμβολο '*plus*' από το κεντρικό μενού πλοήγησης που εμφανίζεται.

3. Ο χρήστης μεταφέρεται στη σελίδα δημιουργίας νέου Hotspot (*CreateHotspotScreen*).

4. Ο χρήστης συμπλήρώνει τη φόρμα δημιουργίας νέου Hotspot. Τα απαιτούμενα πεδία είναι τα εξής:

   - περιγραφή (*description*)
   - χρονική διάρκεια ισχύος σε λεπτά (*validity*)

   Εάν ο χρήστης το επιθυμεί, μπορεί να προσθέσει μια φωτογραφία από τη συλλογή φωτογραφιών του ή να βγάλει καινούρια μέσω της κάμερας της συσκευής.

5. Στην περίπτωση που δεν υπάρχουν σφάλματα κατά τη δημιουργία νέου Hotspot, η διαδικασία ολοκληρώνεται επιτυχώς και ο χρήστης μεταφέρεται αυτόματα στην αρχική σελίδα. Το νέο Hotspot θα εμφανίζεται στο χάρτη στην τοποθεσία του χρήστη.

Γρήγορη Προβολή Hotspot

1. Στο χάρτη της αρχικής οθόνης, ο χρήστης πατάει πάνω στο εικονίδιο ενός hotspot (*hotspot marker*).

2. Οι λεπτομέρειες του hotspot εμφανίζονται σε ένα '*παράθυρο*' ενσωματωμένο στο χάρτη (*callout*). ο χρήστης μπορεί να δει την περιγραφή, τον αριθμό σχολίων και προβολών, και αν υπάρχει, την εικόνα του επιλεγμένου hotspot.

Προβολή Λεπτομερειών Hotspot

1. Στο χάρτη της αρχικής οθόνης, ο χρήστης πατάει πάνω στο εικονίδιο ενός hotspot (*hotspot marker*).

2. Οι λεπτομέρειες του hotspot εμφανίζονται σε ένα '*παράθυρο*' ενσωματωμένο στο χάρτη (*callout*). ο χρήστης μπορεί να δει την περιγραφή, τον αριθμό σχολίων και προβολών, και αν υπάρχει, την εικόνα του επιλεγμένου hotspot.

3. Ο χρήστης πατάει πάνω στο εικονίδιο '*comments*' και μεταφέρεται αυτόματα στην οθόνη λεπτομερειών του συγκεκριμένου hotspot.

Δημιουργία Σχολίου

1. Ο χρήστης πηγαίνει στη σελίδα λεπτομερειών ενός hotspot (*CommentsScreen*) ακολουθώντας τη διαδικασία που περιγράφεται στην παράγραφο *Προβολή Λεπτομερειών Hotspot*.

2. Ο χρήστης πλοηγείται στο κάτω μέρος της σελίδας.

   Αλλιώς,

3. Ο χρήστης σύρει προς τα αριστερά ένα από τα υπάρχοντα σχόλια και πατάει το εικονίδιο '*reply*'.

4. Ο χρήστης εισάγει το σχόλιό του στο *comment box* με επιγραφή '*Add a comment...*'.

5. Μόλις ο χρήστης ολοκληρώσει το σχόλιό του, πατάει στο εικονίδιο '*send*'.

6. Σε περίπτωση που το σχόλιο δεν έχει σφάλματα, η διαδικασία ολοκληρώνεται επιτυχώς και το νέο σχόλιο εμφανίζεται τελευταίο στο τμήμα σχολίων της οθόνης.

Προβολή Περισσότερων Σχολίων

1. Ο χρήστης πηγαίνει στη σελίδα λεπτομερειών ενός hotspot (*CommentsScreen*) ακολουθώντας τη διαδικασία που περιγράφεται στην παράγραφο *Προβολή Λεπτομερειών Hotspot*.

2. Ο χρήστης πλοηγείται στο κάτω μέρος της σελίδας.

3. Ο χρήστης πατάει το πλήκτρο με επιγραφή '*more*'. Πέντε νέα σχόλια εμφανίζονται στο τμήμα σχολίων της οθόνης.

Συγκεκριμένη Αναζήτηση Σημείου Ενδιαφέροντος

1. Στη σελίδα χάρτη (*HomeScreen*) ο χρήστης πατάει εντός της μπάρας αναζήτησης με επιγραφή '*Search a hotspot...*' που βρίσκεται στο πάνω μέρος της οθόνης.

2. Ο χρήστης πληκτρολογεί το όνομα ή κάποια άλλη λέξη-κλειδί που σχετίζεται με το σημείο ενδιαφέροντος.

3. Ο χρήστης πατάει σε μία από τις προτεινόμενες επιλογές από τη λίστα που εμφανίζεται.

4. Το επιλεγμένο σημείο ενδιαφέροντος εμφανίζεται στο χάρτη.

Γενικευμένη Αναζήτηση Σημείου Ενδιαφέροντος

1. Στη σελίδα χάρτη (*HomeScreen*) ο χρήστης πατάει εντός της μπάρας αναζήτησης με επιγραφή '*Search a hotspot...*' που βρίσκεται στο πάνω μέρος της οθόνης.

2. Ο χρήστης πληκτρολογεί το όνομα ή κάποια άλλη λέξι-κλειδί που σχετίζεται με το σημείο ενδιαφέροντος.
3. Ο χρήστης πατάει το εικονίδιο '*search*'.
4. Τα αποτελέσματα εμφανίζονται στο χάρτη.

Αναζήτηση Σημείου Ενδιαφέροντος με Φίλτρο

1. Στη σελίδα χάρτη (*HomeScreen*) ο χρήστης πατάει το εικονίδιο '*menu*' που βρίσκεται στο πάνω αριστερά μέρος της οθόνης.
2. Στο '*drawer menu*' που εμφανίζεται, ο χρήστης επιλέγει μία από τις κατηγορίες πατώντας επάνω σε αυτή. Οι διαθέσιμες κατηγορίες είναι:
   - *Coffee*
   - *Food*
   - *Drinks*
   - *Sights*
   - *Arts*
3. Τα αποτελέσματα εμφανίζονται στο χάρτη. Κάθε φορά που ο χρήστης αλλάζει τη θέση του χάρτη, νέα αποτελέσματα φορτώνονται στη νέα θέση.

Αναίρεση Αναζήτησης

1. Στη σελίδα χάρτη (*HomeScreen*) ο χρήστης πατάει το εικονίδιο '*menu*' που βρίσκεται στο πάνω αριστερά μέρος της οθόνης.
2. Στο '*drawer menu*' που εμφανίζεται, ο χρήστης πατάει την επιλογή '*Clear*'.
3. Τα αποτελέσματα αφαιρούνται από το χάρτη.

Επαναφορά Θέσης Χάρτη (*Find My Location*)

1. Στη σελίδα χάρτη (*HomeScreen*) ο χρήστης πατάει το εικονίδιο '*location*' που βρίσκεται στο κάτω αριστερά μέρος της οθόνης.
2. Η θέση του χάρτη επαναφέρεται στην τοποθεσία του χρήστη.

Προβολή Λίστας Προσωπικών Hotspots

1. Ο χρήστης πατάει πάνω στο FAB στο κάτω δεξιά μέρος της οθόνης στην HomeScreen.
2. Ο χρήστης πατάει επάνω στο εικονίδιο με το σύμβολο '*hotspots*' από το κεντρικό μενού πλοήγησης που εμφανίζεται.
3. ο χρήστης μεταφέρεται στη σελίδα προσωπικών Hotspots (*HotspotListScreen*). Στη λίστα εμφανίζονται λεπτομέρειες των hotspots του χρήστη. Ο χρήστης μπορεί να δει την περιγραφή, τον αριθμό σχολίων και προβολών του hotspot.

Επεξεργασία Προσωπικού Hotspot

1. ο χρήστης μεταφέρεται στη σελίδα προσωπικών Hotspots (*HotspotListScreen*) ακολουθώντας τη διαδικασία που περιγράφεται στην παράγραφο *Προβολή Λίστας Προσωπικών Hotspots*.

2. Ο χρήστης σύρει προς τα δεξιά το hotspot που θέλει να επεξεργαστεί. Στη συνέχεια πατάει στο εικονίδιο '*edit*' που εμφανίζεται εντός του κίτρινου πλαισίου.

3. Ο χρήστης μεταφέρεται αυτόματα στη σελίδα επεξεργασίας hotspot (*EditHotspotScreen*).

4. Ο χρήστης επεξεργάζεται τα επιθυμητά στοιχεία του hotspot και όταν τελειώσει πατάει '*Save*' στο πάνω δεξιά μέρος της οθόνης.

5. Σε περίπτωση που δεν υπάρχουν σφάλματα, η διαδικασία ολοκληρώνεται επιτυχώς και οι αλλαγές αποθηκεύονται. Το hotspot εμφανίζεται πλέον με τις νέες αλλαγές.

Διαγραφή Προσωπικού Hotspot

1. ο χρήστης μεταφέρεται στη σελίδα προσωπικών Hotspots (*HotspotListScreen*) ακολουθώντας τη διαδικασία που περιγράφεται στην παράγραφο *Προβολή Λίστας Προσωπικών Hotspots*.

2. Ο χρήστης σύρει προς τα αριστερά το hotspot που θέλει να διαγράψει. Στη συνέχεια πατάει στο εικονίδιο '*delete*' που εμφανίζεται εντός του κόκκινου πλαισίου.

3. Το hotspot διαγάφεται από τη λίστα προσωπικών hotspots.

Προβολή Προσωπικού Λογαριασμού (*Profile*)

1. Ο χρήστης πατάει πάνω στο FAB στο κάτω δεξία μέρος της οθόνης στην HomeScreen.

2. Ο χρήστης πατάει επάνω στο εικονίδιο με το σύμβολο '*account*' (ή την εικόνα profile, αν έχει επιλέξει μία) από το κεντρικό μενού πλοήγησης που εμφανίζεται.

3. ο χρήστης μεταφέρεται στη σελίδα προσωπικού λογαριασμού (*ProfileScreen*). Σε αυτή την οθόνη ο χρήστης μπορεί να προβάλλει τα προσωπικά του στοιχεία.

Επεξεργασία Προσωπικού Λογαριασμού (*Profile*)

1. ο χρήστης μεταφέρεται στη σελίδα προσωπικού λογαριασμού (*ProfileScreen*) ακολουθώντας τη διαδικασία που περιγράφεται στην παράγραφο '*Προβολή Προσωπικού Λογαριασμού*'.

2. ο χρήστης πατάει στο πλήκτρο '*Edit*' που βρίσκεται στο επάνω δεξιά μέρος της οθόνης.

3. ο χρήστης μεταφέρεται στη σελίδα επεξεργασίας του προσωπικού λογαριασμού (*EditProfileScreen*). Η οθόνη αυτή αποτελείται από μια φόρμα με τα αρχικά στοιχεία του χρήστη. Ο χρήστης μπορεί να επεξεργαστεί τα προσωπικα του στοιχεία.

4. Σε περίπτωση επιτυχούς υποβολής της φόρμας επεξεργασίας των στοιχείων του προσωπικού λογαριασμού, τα νέα στοιχεία αποθηκεύονται στο σύστημα και ο χρήστης επαναφέρεται στο *profile* του.

Προβολή Στατιστικών Δεδομένων Λογαριασμού

1. ο χρήστης μεταφέρεται στη σελίδα προσωπικού λογαριασμού (*ProfileScreen*) ακολουθώντας τη διαδικασία που περιγράφεται στην παράγραφο '*Προβολή Προσωπικού Λογαριασμού*'.

2. ο χρήστης πατάει στο πλήκτρο βέλους με την επιγραφή '*Stats for nerds*' στο τμήμα '*Settings*'.

3. Ο χρήστης μεταφέρεται στη σελίδα στατιστικών δεδομένων του προσωπικού του λογαριασμού (*StatisticsScreen*). Η οθόνη διαιρείται σε τρία τμήματα:

   (α) *account stats*

   Στατιστικά δεδομένα που αφορούν τις κινήσεις του χρήστη εντός της εφαρμογής. Τα δεδομένα αυτά είναι ο συνολικός αριθμός των hotspots του χρήστη, ο συνολικός αριθμός σχολίων του χρήστη και ο συνολικός αιρθμός προβολών των hotspots του χρήστη από άλλους χρήστες.

   (β) *insights*

   Ποσοστιαίες τιμές που αφορούν την δράση και εξέλιξη του χρήστη εντός της εφαρμογής. Οι τιμές αυτές είναι η δημοτικότητα (popularity) και η αλληλεπίδραση (engagement) του χρήστη.

   (γ) *audience*

   Διάκριση του κοινού του χρήστη με βάση το γένος (θηλυκό ή αρσενικό).

3.3.2 Υποδομή Server και Βάσης Δεδομένων (*Backend*)

Στην προηγούμενη ενότητα, παρουσιάστηκαν οι βασικές λειτουργικότητες που προσφέρει η εφαρμογή στους χρήστες της. Αναλύθηκαν οι υπηρεσίες της εφαρμογής καθώς επίσης και τα σενάρια με τα οποία ο χρήστης μπορεί να αποκτήσει πρόσβαση σε αυτές. Έτσι, όταν ο χρήστης επιθυμεί για παράδειγμα να δημιουργήσει ένα νέο hotspot, αρκεί να πλοηγηθεί στην οθόνη δημιουργίας νέου hotspot, να συμπληρώσει την αντίστοιχη φόρμα και να την υποβάλλει. Αυτές είναι οι ενέργειες που πρέπει να γίνουν από την πλευρά του client. Αυτή η ενότητα ασχολείται με τις ενέργειες που απαιτούνται από την πλευρά του server και τον τρόπο με τον οποίο αυτές είναι σχεδιασμένες να αλληλεπιδρούν μεταξύ τους αλλά και με το υπόλοιπο σύστημα.

Είναι προφανής από τα παραπάνω η ανάγκη αποσύμπλεξης των αιτημάτων προς εξυπηρέτηση από το server. Κάθε λειτουργία της εφαρμογής χαρακτηρίζεται από ένα σύνολο ενεργειών που πρέπει να εκτελεστούν κάθε φορά προκειμένου να προκληθεί το επιθυμητό αποτέλεσμα. Για παράδειγμα, η δημιουργία ενός hotspot πυροδοτεί μια σειρά αιτημάτων προς τον server όπως η συσχέτιση του hotspot με το δημιουργό του, η αποθήκευση των δεδομένων του hotspot στην βάση, η καταγραφή των σχολίων του hotspot κλπ. Έτσι, η σχεδίαση θα πρέπει να βρίσκεται σε συμφωνία με τις απαιτήσεις που πρέπει να πληροί το σύστημα. Ανακύπτει λοιπόν το συμπέρασμα πως είναι αδύνατη η σειριακή εξυπηρέτηση όλων των αιτημάτων από τον ίδιο πόρο, καθώς αυτό θα καθιστούσε το σύστημα υπερβολικά αργό και περίπλοκο.

Συνεπώς, η πρακτική που ακολουθεί η δομή του server αποσκοπεί στην αποσυσχέτιση διαφορετικών αιτημάτων. Αυτό επιτυγχάνεται με την ιεράρχηση των λειτουργιών που αφορούν τα αιτήματα, με βάση τη φύση τους. Αιτήματα που σχετίζονται με τους χρήστες θα ανατίθενται αποκλειστικά σε εξυπηρετητές ενεργειών των χρηστών, αιτήματα που αφορούν τα hotspots θα ανατίθενται αποκλειστικά σε εξυπηρετητές ενεργειών των hotspots κοκ. Η αποθήκευση δεδομένων στη βάση βρίσκεται σε συνοχή με την παραπάνω τακτική, αφού τα δεδομένα μοντελοποιούνται σε collections βάσει της φύση τους. Με αυτό τον τρόπο, επιτυγχάνεται η παραλληλοποίηση των λειτουργιών του server, η διαίρεση του φορτίου και η διακριτοποίηση των μεθόδων. Το τελευταίο έχει σημαντική συνεισφορά στην διευκόλυνση της σχεδίασης του συστήματος από τον προγραμματιστή, χάρις τον μεγάλο βαθμό επαναληψιμότητας των περισσότερων τμημάτων.

3.3.3 Διεπικοινωνία Μεταξύ Υποδομών

Το σύστημα client-server επικοινωνεί μέσω αιτημάτων συγκεκριμένου σκοπού. Κάθε φορά που ο χρήστης πραγματοποιεί μια ενέργεια εντός της εφαρμογής, ο client δημιουργεί ένα αίτημα προς στον server. Το αίτημα αυτό δρομολογείται στον κατάλληλο εξυπηρετητή από την πλευρά του server. Έπειτα, ακολουθεί η ανάθεση του αιτήματος στον αντίστοιχο πόρο και η εκτέλεση των απαραίτητων λειτουργιών. Το αποτέλεσμα επηρεάζει τα δεδομένα στη βάση και επιστρέφει με κατάλληλη μορφή πίσω στον client. Η παραπάνω διαδικασία γίνεται βέλτιστα, μέσω ασύγχρονων μεθόδων διαχείρισης δεδομένων από και προς τη βάση δεδομένων. Η διαδικασία εξυπηρέτησης αιτημάτων είναι παραλληλοποιημένη ώστε πολλά αιτήματα να εξυπηρετούνται από διαφορετικούς πόρους του συστήματος ταυτόχρονα.

Κεφάλαιο 4

Σχεδίαση Συστήματος

Στο προηγούμενο κεφάλαιο έγινε μια εισαγωγική αναφορά στα δομικά μέρη του συστήματος της εφαρμογής, δηλάδή του frontend (client) και του backend (server και βάση δεδομένων). Αναπτύχθηκαν οι λειτουργικότητες της εφαρμογής και οι αντίστοιχες οθόνες που θα τις καλύπτουν. Επίσης, έγινε μια αναφορά στα συστατικά του server και πως αυτά αλληλεπιδρούν με τη βάση δεδομένων για την λήψη και αποστολή δεδομένων από και προς τον client.

Αυτό το κεφάλαιο θα επικεντρωθεί στην σχεδίαση των υποδομών της εφαρμογής. Συγκεκριμένα, θα γίνει αναφορά στις διεπιφάνειες που θα συνιστούν κάθε οθόνη της εφαρμογής και θα παρουσιαστούν τα διαγράμματα ροής για τα σενάρια χρήσης της εφαρμογής (βλ. ενότητα 4.1). Στην ενότητα 4.2, θα γίνει αναφορά στην πλευρά του server. Θα παρουσιαστεί η διαδικασία σχεδίασης των εξυπηρετητών αιτημάτων και θα μελετηθούν οι τακτικές και οι τεχνολογίες που χρησιμοποιήθηκαν. Τέλος, στην ενότητα 4.3, θα γίνει μια εκτενής ανάλυση του τρόπου με τον οποίο σχεδιάστηκε η βάση δεδομένων για να ανταποκρίνεται βέλτιστα στις απαιτήσεις της εφαρμογής.

4.1 Σχεδίαση Μοντέλων των Διεπιφανειών της Εφαρμογής

Όπως έχει ήδη αναφερθεί, η εφαρμογή αποτελείται από οθόνες, καθεμία εκ των οποίων συνιστά και μια διεπαφή χρήστη. Σε αυτή την ενότητα θα αναλυθούν οι τεχνικές σχεδίασης των διεπαφών χρήστη και θα παρουσιαστούν παραστατικά διαγράμματα των συστατικών μερών τους.

4.1.1 Αρχιτεκτονική Σχεδίασης Οθονών

Κατα την σχεδίαση μιας mobile εφαρμογής, ο προγραμματιστής καλείται να αποφανθεί σχετικά με τον τρόπο διαμερισμού του εσωτερικού της εφαρμογής (compartmentalization). Τα δομικά μέρη της εφαρμογής θεωρούνται ως εξαρτήματα, τα οποία στο σύνολό τους συνθέτουν μεγαλύτερα τμήματα της εφαρμογής. Τέτοια εξαρτήματα είναι τα πλήκτρα ενεργειών,

τα μενού, οι διεπαφές εισόδου και όλα τα διαδραστικά στοιχεία της εφαρμογής. Η κατάλληλη διασύνδεση μεταξύ πολλών διαδραστικών στοιχείων διαμορφώνει μια διεπιφάνεια χρήστη. Η διεπιφάνειες χρήστη ή αλλιώς κλάσεις, αποτελούν το κύριο συνθετικό στοιχείο (component) μιας οθόνης της εφαρμογής. Κάθε οθόνη περιλαμβάνει διαφορετικές κλάσεις, οι οποίες εξυπηρετούν διαφορετικούς σκοπούς. Για παράδειγμα, η οθόνη εγγραφής/σύνδεσης του χρήστη στην εφαρμογή αποτελείται από μια διεπαφή ταυτοποίησης, η οποία περιέχει μια φόρμα με πολλαπλά στοιχεία εισόδου κειμένου, πλήκτρα και μηνύματα. η κεντρική οθόνη της εφαρμογής συνίσταται από μια διεπαφή χάρτη, η οποία αποτελεί τη διεπιφάνεια προβολής όλων των hotspots και περιέχει markers για καθένα από τα hotspots.

Είναι προφανές λοιπόν πως, αν και η οθόνη ταυτοποίησης χρήστη διαφέρει ως προς το σκοπό της από την οθόνη χάρτη, η δομή τους ακολουθεί μια ομοιόμορφη αρχιτεκτονική σχεδίασης. Στη συνέχεια, θα γίνει μια αναλυτική αναφορά στις μεθόδους που χρησιμοποιήθηκαν κατά τη σχεδίαση των οθονών της εφαρμογής και θα παρουσιαστούν τα μοντέλα κάθε διεπιφάνειας.

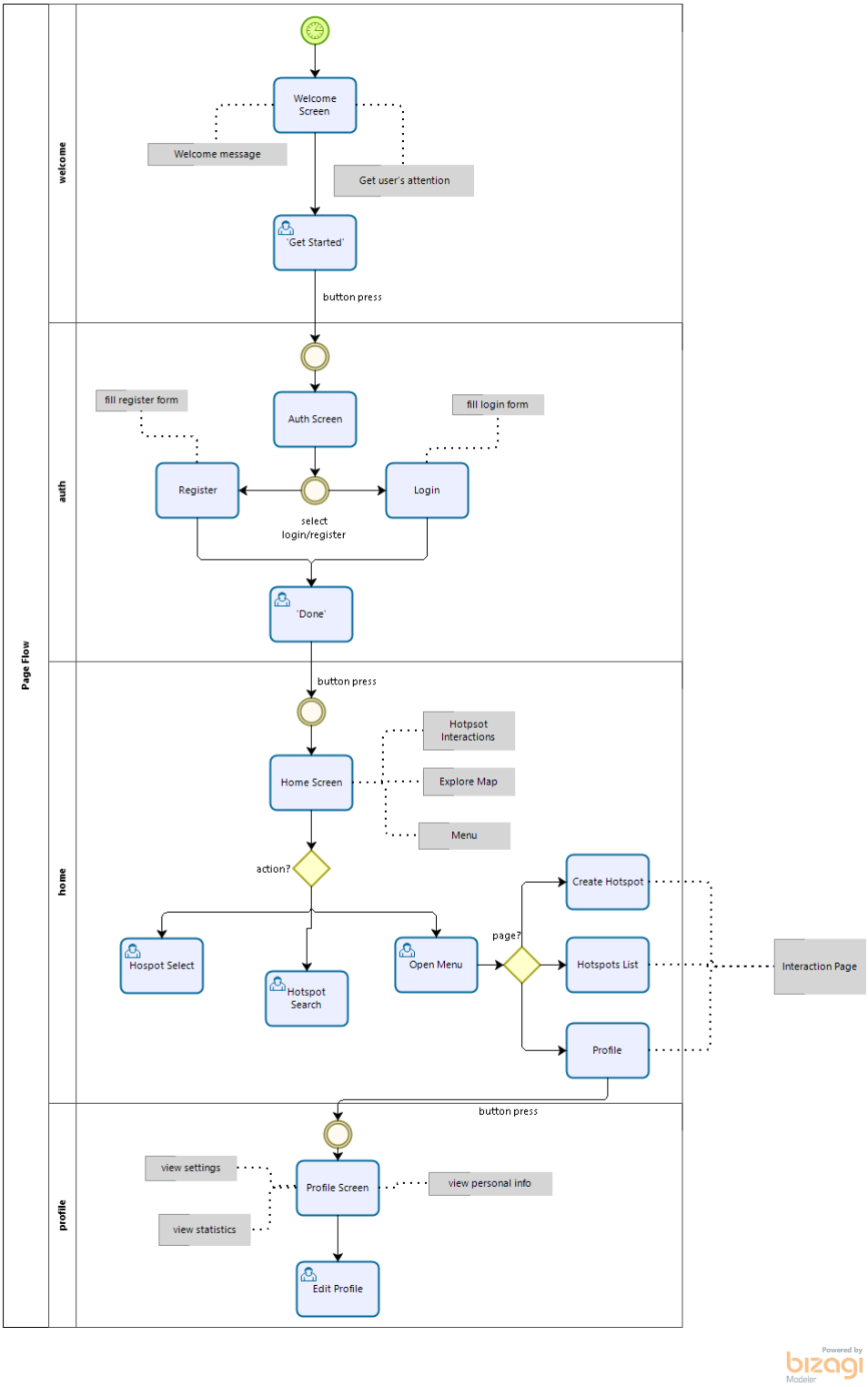
4.1.2 Λογική Σχεδίασης και Ροή Πληροφορίας Μεταξύ οθονών

Η λογική σχεδίασης που ακολουθεί η εφαρμογή διακρίνεται σε 5 στάδια, τα οποια σειριοποιούνται ως εξής:

1. οθόνη έναρξης
2. ταυτοποίηση χρήστη
3. κεντρική οθόνη
4. οθόνες μηνυμάτων και αλληλεπίδρασης
5. προσωπικός λογαριασμός και ρυθμίσεις

Το μοντέλο αυτό ροής της πληροφορίας (βλ. Σχ. 4.1) ακολουθείται από την πλειοψηφία των σημερινών εφαρμογών. Κατά τη διαδικασία εξερεύνησης της εφαρμογής, ο χρήστης εισάγεται στον τρόπο λειτουργίας της εφαρμογής σταδιακά. Κάθε στάδιο προσφέρει στο χρήστη εμπειρία συγκεκριμένου τύπου, χωρίς να τον επιβαρρύνει με αχρείαστες υπηρεσίες. Ο χρήστης θα πρέπει να μπορεί να αποκτήσει εμπειρία άμεσα. Η εκμάθηση της εφαρμογής θα πρέπει να είναι εύκολη και αβίαστη και όχι να δημιουργεί την αίσθηση ανικανότητας ή δυσκολίας χειρισμού.

Επίσης, κάθε σημείο της εφαρμογής θα πρέπει να είναι εύκολα προσβάσιμο από κάθε οθόνη της εφαρομογής. Για το λόγο αυτό κάθε οθόνη πρέπει να περιλαμβάνει μια επικεφαλίδα με πλήκτρα πλοήγησης και χαρακτηριστικά γνωρίσματα που ενημερώνουν το χρήστη για την πορεία του εντός της εφαρμογής.



Σχήμα 4.1: Ροή πληροφορίας μεταξύ οθονών της εφαρμογής.

Στο πρώτο στάδιο της εφαρμογής, ο χρήστης εισάγεται με άμεσο και σύντομο τρόπο στο νόημα της εφαρμογής. Ένα φιλικό μήνυμα καλωσορίσματος εξυπηρετεί στη διαμόρφωση

μιας θετικής πρώτης εντύπωσης στο χρήστη. Το μόνο απαραίτητο διαδραστικό μέσο είναι το πλήκτρο που θα μεταφέρει το χρήστη στο επόμενο στάδιο.

Το στάδιο ταυτοποίησης του χρήστη απαρτίζεται από δύο σελίδες. Η πρώτη σελίδα αποτελείται από τη φόρμα εγγραφής, ενώ η δεύτερη διαθέτει μια φόρμα σύνδεσης. Και οι δύο σελίδες συνοδεύονται από ένα πλήκρο υποβολής. Κάθε φόρμα θα πρέπει να είναι σαφής, τόσο ως προς τα στοιχεία που απαιτεί από το χρήστη, όσο και ως προς τον τρόπο συμπλήρωσής της. Για το λόγο αυτό γίνεται χρήση διευκρυνιστικών τίτλων, παραδειγμάτων εισόδου και άλλων μηνυμάτων τέτοιου σκοπού. Επίσης, θα πρέπει να ληφθούν υπόψη πιθανά σφάλματα που μπορεί να προκύψουν, όπως η αγνόηση ενός ή περισσότερων σημείων εισόδου, η λανθασμένη συμπλήρωση ενός ή περισσότερων κελιών, ο λανθασμένος συνδυασμός διαπιστευτηρίων του χρήστη κλπ. Η εφαρμογή θα διαθέτει μια πλήρως καταρτισμένη και εξυπηρετική διεπαφή εμφάνισης μηνυμάτων λάθους στο χρήστη, με σκοπό την εύκολη διόρθωση σφαλμάτων και τη γρήγορη σύνδεση/εγγραφή στην εφαρμογή.

Η κεντρική οθόνη είναι το πιο σύνθετο στάδιο της εφαρμογής. Αποτελείται από πολλά διαδραστικά στοιχεία και διεπαφές χρήστη, οι κυριότερες εκ των οποίων είναι ο χάρτης των hotspots, η μπάρα αναζήτησης, το μενού πλοήγησης και το μενού επιλογής φίλτρων. Αν και η πολυπλοκότητα είναι υψηλή, οι αντίστοιχες διεπιφάνειες είναι σχεδιασμένες με τέτοιο τρόπο, ώστε να απλοποιούν τις ενέργειες που απαιτούνται από το χρήστη. Εικονίδια, τίτλοι και βοηθητικές περιγραφές χρησιμοποιούνται καθ΄ όλη την έκταση της κεντρικής σελίδας. Τα μηνύματα είναι σύντομα, χρησιμοποιώντας όσο το δυνατόν λιγότερους χαρακτήρες για να διευκολύνουν το χρήστη και να δημιουργούν μια εύχρηστη και ευανάγνωστη διεπιφάνεια. Άλλωστε, ο χώρος στην οθόνη μιας κινητής συσκευής είναι εξαρχής περιορισμένος και δεν ευνοεί το συνωστισμό πληροφοριών σε ένα σημείο ή την υπερβολική λεπτομέρεια στο σχεδιασμό.

Το ενδιάμεσο στάδιο ορίζεται από τις οθόνες αλληλεπίδρασης του χρήστη με τις εξειδικευμένες υπηρεσίες που προσφέρει η εφαρμογή. Η οθόνη γρήγορης προβολής ενός hotspot, η οθόνη προβολής λεπτομερειών ενός hotspot, η οθόνη δημιουργίας ενός hotspot και η οθόνη λίστας προσωπικών hotspots του χρήστη αποτελούν το πεδίο δράσης του χρήστη. Οι οθόνες αυτές διαμορφώνουν τη φυσιογνωμία της εφαρμογής και τη διακρίνουν από τις υπόλοιπες εφαρμογές, με τις μοναδικές λειτουργικότητες που προσφέρουν.

Τέλος, το στάδιο του προσωπικού λογαριασμού αποτελείται από τις οθόνες που αφορούν τα προσωπικά στοιχεία του χρήστη. Οι οθόνες αυτές είναι σχεδιασμένες έτσι ώστε να προσφέρουν με άμεσο και σύντομο αλλά περιεκτικό τρόπο μια εικόνα του λογαριασμού του εκάστοτε χρήστη. Επιπλέον, ο χρήστης έχει τη δυνατότητα να προβάλλει επιπρόσθετα στοιχεία που αφορούν τη δράση του εντός της εφαρμογής. Στην οθόνη του προσωπικού λοαγιασμού, ο χρήστης μπορεί να τροποποιήσει τις προσωπικές του ρυθμίσεις, καθώς και να επεξεργαστεί τα προσωπικά του στοιχεία.

4.1.3 Διεπαφές Μηνυμάτων Προς το Χρήστη

Η εφαρμογή είναι σχεδιασμένη ώστε να επιτρέπει τη μέγιστη δυνατή ευελιξία στο χρήστη εντός της εφαρμογής. Ωστόσο, προκειμένου να αποφευχθούν σφάλματα και σενάρια μη επιτρε-

πτής λειτουργίας, είναι απαραίτητο να ληφθούν μέτρα που προστατεύουν το χρήστη εντός της εφαρμογής, αλλά και την ίδια την εφαρμογή. Είναι χαρακτηριστικό της εφαρμογής να δέχεται είσοδο από το χρήστη σε πολλαπλά σημεία, με αποτέλεσμα να είναι απαραίτητος κάθε φορά ο έλεγχος της παρεχόμενης εισόδου του χρήστη, για να εξασφαλίζεται κάθε φορά η εύρυθμη λειτουργία της εφαρμογής και ταυτοχρόνως να προλαμβάνονται ανεπιθύμητες ενέργειες από την πλευρά του χρήστη με τον καλύτερο δυνατό τρόπο.

Ο ρόλος του ελεγκτή εισόδου του χρήστη αποδίδεται σε μια διεπαφή μηνυμάτων που σχεδιάστηκε για αυτό το σκοπό. Η διεπαφή μηνυμάτων είναι υπεύθυνη για τον έλεγχο της εισόδου που δίνεται από την πλευρά του χρήστη, την κατάλληλη τροποποίηση και την αποδοχή ή απόρριψή της. Στη συνέχεια, εμφανίζει μήνυμα λάθους ή επιτυχίας ανάλογα με την έκβαση τις εκάστοτε ενέργειας. Σε κάθε οθόνη που διαθέτει ένα ή και περισσότερα στοιχεία εισόδου, είναι πιθανό αυτά να διαφέρουν μεταξύ τους ως προς τις απαιτούμενες προϋποθέσεις που πρέπει να ικανοποιούν. Για το λόγο αυτό η διεπαφή μηνυμάτων λάθους συνίσταται από ένα σύστημα, στο οποίο κάθε διεπαφή εισόδου αντιστοιχίζεται σε έναν ελεγκτή σφάλματος. Στην περίπτωση όπου η είσοδος του χρήστη δεν ανήκει μεταξύ των επιθυμητών, εμφανίζεται στην οθόνη κατάλληλο μήνυμα σφάλματος. Εάν η είσοδος που δωθεί απότ ο χρήστη ανήκει στις επιτρεπτές τιμές, τότε η διεπαφή είτε εμφανίζει μήνυμα επιτυχίας, είτε ανακατευθύνει το χρήστη στην ανανεωμένη οθόνη.

Τα είδη των μηνυμάτων προς το χρήστη μπορούν να ταξινομηθούν σε τέσσερεις κατηγορίες:

- μηνύματα σφάλματος
- μηνύματα επιβεβαίωσης
- μηνύματα επιτυχίας
- μηνύματα δικαιωμάτων

Τα μηνύματα σφάλματος εμφανίζονται στην περίπτωση που ο χρήστης δώσει είσοδο που δεν ανήκει στις επιτρεπτές τιμές. Κάθε είσοδος του χρήστη πρέπει να πληροί κάποιες προϋποθέσεις. Για παράδειγμα, σε μια φόρμα εγγραφής, το πεδίο ηλεκτρονικής διεύθυνσης θα πρέπει να περιέχει μια σειρά χαρακτήρων που πληρούν τα κριτήρια (πχ. example@domain.com). Στην περίπτωση που η είσοδος του χρήστη δεν ικανοποιεί την παραπάνω συνθήκη, ένα κατάλληλο μήνυμα λάθους εμφανίζεται στην οθόνη. Όταν η είσοδος του χρήστη είναι μεταξύ των επιθυμητών, το μήνυμα λάθους εξαφανίζεται.

Σε κάποια σημεία της εφαρμογής είναι πιθανό ο χρήστης να αλλάξει γνώμη για μια ενέργεια. Τα κρίσιμα αυτά σημεία συνοδεύονται απο ένα στάδιο επιβεβαίωσης πριν την υποβολή των δεδομέων στη βάση και την πραγματοποίηση της αντίστοιχων λειτουργιών. Το στάδιο αυτό αποτελείται από ένα μήνυμα στο οποίο διευκρινίζονται στο χρήστη οι ενέργειες που θα επακολουθήσουν. Εάν ο χρήστης είναι σύμφωνος με αυτές, τότε προχωράει στη διαδικασία αποδοχής, ενώ αν δεν επιθυμεί να πραγματοποιηθούν οι παραπάνω ενέργειες ακυρώνει τη διαδικασία. Με τον τρόπο αυτό, η εφαρμογή προσθέτει ένα ακόμη στάδιο προστασίας του χρήστη,

από πιθανά δικά του σφάλματα. Για παράδειγμα, στην οθόνη δημιουργίας νέου hotspot, μόλις ο χρήστης συμπληρώσει σωστά την αντίστοιχη φόρμα και πατήσει το πλήκτρο υποβολής, ένα μήνυμα επιβεβαίωσης εμφανίζεται στην οθόνη.

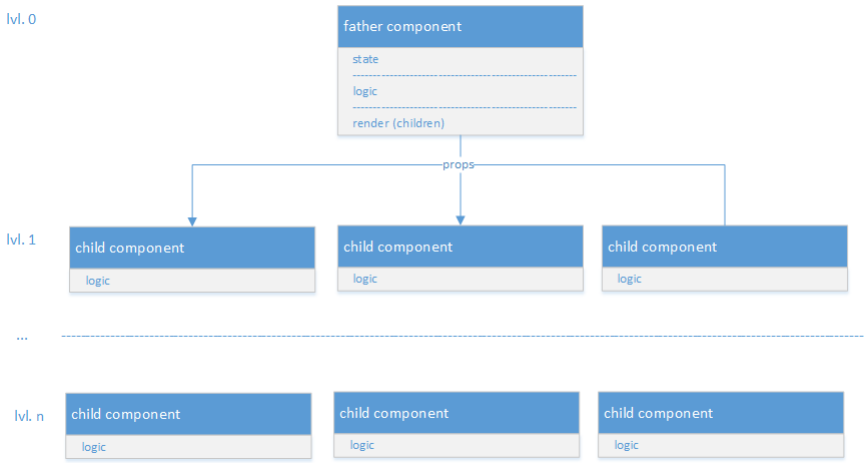
Κατά τη διαδικασία δημιουργίας νέου hotspot, ο χρήστης επιλέγει ένα χρονικό διάστημα ισχύος του hotspot. Μετά την επιλογή, ένα μήνυμα επιτυχίας εμφανίζεται στην οθόνη, διευκρινίζοντας στο χρήστη το αποτέλεσμα της ενέγειας που μόλις πραγματοποιήθηκε. Στις περισσότερες περιπτώσεις επιτυχίας, ο χρήστης ενημερώνεται έμμεσα, από την ανανέωση της οθόνης με τη συμπερίληψη των καινούριων δεδομένων που παρείχε ο ίδιος.

Τέλος, όπως έχει ήδη αναφερθεί, η εφαρμογή κάνει χρήση εξωτερικών ρυθμίσεων της συσκευής σε αρκετά σημεία. Για το λόγο αυτό είναι απαραίτητη η εμφάνιση κατάλληλων μηνυμάτων στα σημεία αυτά, όπου ζητείται από το χρήστη η παροχή του δικαιώματος στην εφαρμογή να πραγματοποιήσει αλλαγές στις ρυθμίσεις της συσκευής για την ολοκληρωμένη και εύρυθμη λειτουργία της. Ο χρήστης ενημερώνεται κάθε φορά με κατάλληλο μήνυμα για το είδος των ζητούμενων δικαιωμάτων και μπορεί να αποδεχθεί ή να απορρίψει την παροχή αυτών προς την εφαρμογή.

4.1.4 Screen Containers

Κάθε οθόνη αποτελείται από ένα σύνολο κλάσεων και στοιχείων που συνθέτουν την διεπιφάνεια στην οποία δρα ο χρήστης. Το σύνολο αυτό συνιστά ένα container, ενώ τα στοιχεία που αποτελούν εξαρτήματα της διεπιφάνειας αποκαλούνται components. Κάθε screen έχει δικά της components, τα οποία επικοινωνούν μεταξύ τους μέσω του state. Η επικοινωνία μεταξύ components ακολουθεί το μοτίβο πατέρα-παιδιού, σύμφωνα με το οποίο η πληροφορία μεταδίδεται από το υψηλότερο (πατέρας) στα χαμηλότερα επίπεδα (παιδιά). Η σχέση αυτή είναι μονοσήμαντη. Αυτό σημαίνει πως η πληροφορία ρέει προς μία μόνο κατεύθυνση, από τον πατέρα προς τα παιδιά.

Τα components χωρίζονται σε δύο είδη. Αυτά που αναπαριστούν δεδομένα και αυτά που διαχειρίζονται την κατάσταση των δεδομένων στις διάφορες χρονικές στιγμές κατά τον κύκλο ζωής της εφαρμογής. Η πρώτη κατηγορία, αποτελείται από functional ή αλλιώς representational components. Τέτοιας μορφής είναι συνήθως τα παιδιά. Δεν διαθέτουν state και μπορούν μόνο να αναπαραστήσουν λογική. Από την άλλη, τα stateful components μπορούν να πραγματοποιήσουν αλλαγές στο state και να διαχειριστούν λογική. Κάθε οθόνη διαθέτει τουλάχιστον ένα stateful component το οποίο αποτελεί και τον πατέρα. Ο αριθμός των παιδιών ποικίλλει και εξαρτάται από τις λειτουργίες που καλείται να εξυπηρετήσει κάθε οθόνη. Είναι προτιμότερη η τακτική διάσπασης των λειτουργιών μιας οθόνης σε επιμέρους components, καθώς έτσι μειώνεται ο όγκος του κώδικα και αποσυμπλέκονται οι λειτουργίες. Κάθε παιδί αποτελείται από ένα ή και περισσότερα επίπεδα με functional components. Κάθε stateful component διαθέτει μία μέθοδο αναπαράστασης δεδομένων (render method). Ο πατέρας χρησιμοποιεί τη render ως μέσο αναπαράστασης των παιδιών του. Ο συγχρονισμός της πληροφορίας επιτυγχάνεται χάρις την ιδιότητα του πατέρα να μεταφέρει το state στα παιδιά του μέσω των props.



Σχήμα 4.2: Επικοινωνία μεταξύ πατέρα-παιδιών.

4.1.4.1 WelcomeScreen

Σκοπός της οθόνης έναρξης της εφαρμογής είναι να καλωσορίσει το χρήστη με ένα σύντομο και φιλικό μήνυμα. Η WelcomeScreen παίζει καθοριστικό ρόλο στην πρώτη εντύπωση του χρήστη. Για το λόγο αυτό, θα πρέπει να δωθεί ιδιαίτερη σημασία στα στοιχεία που θα χρησιμοποιηθούν σε αυτό το σημείο της εφαρμογής. Χωρίς πολλές πληροφορίες, ο χρήστης εισάγεται στο νόημα της εφαρμογής και παροτρύνεται να ξεκινήσει να τη χρησιμοποιεί.

Ένα εξίσου σημαντικό στοιχείο που πρέπει να ληφθεί υπόψη είναι η χρονική απόσταση μεταξύ WelcomeScreen και HomeScreen. Στην περίπτωση όπου η έναρξη αποτελεί μια πολυσέλιδη εισαγωγική διαδικασία, με αρκετά κείμενα, είναι πολύ πιθανό να αποθαρρύνει το χρήστη. Ο χρόνος που απαιτείται για να μεταβεί ο χρήστης από την οθόνη έναρξης στην κεντρική οθόνη της εφαρμογής πρέπει να είναι ο ελάχιστος δυνατός. Οι μόνοι παράγοντες από τους οποίους θα εξαρτηθεί αυτό, είναι η διαδικασία εγγραφής του χρήστη στην εφαρμογή και οι απαραίτητες ειδοποιήσεις που σχετίζονται με εξωτερικές ρυθμίσεις της συσκευής για την εύρυθμη και ολοκληρωμένη λειτουργία της εφαρμογής.

Είναι πολύ συχνό το φαινόμενο διαγραφής μιας εφαρμογής προτού ο χρήστης φτάσει στην κεντρική οθόνη. Αυτό οφείλεται κατά κύριο λόγο στα πολλαπλά μηνύματα που εμφανίζονται κατά τη διαδιασία εισόδου στην εφαρμογή. Συνεχόμενες ειδοποιήσεις σχετικές με ρυθμίσεις που απαιτούνται εντός της εφαρμογής κουράζουν το χρήστη και απομακρύνουν την εφαρμογή από το στόχο της, να κερδίσει δηλαδή το ενδιαφέρον του χρήστη. Οι ειδοποιήσεις θα πρέπει να βρίσκονται σε λογικά σημεία στην εφαρμογή και να εμφανίζονται μόνο όταν ο χρήστης πρόκειται να χρησιμοποιήσει μια υπηρεσία που απαιτεί εξωτερικούς πόρους και πρόκειται να αλλάξει τις ρυθμίσεις της συσκευής.

4.1.4.2 RegistrerScreen

Η οθόνη εγγραφής του χρήστη στην εφαρμογή είναι ένα ιδιαίτερα κρίσιμο σημείο της εφαρμογής. Ο χρήστης σχηματίζει την πρώτη του εντύπωση και για το λόγο αυτό θα πρέπει η οθόνη εγγραφής να μην εμποδίζει τη διαμόρφωση μιας θετικής εικόνας. Αυτό σημαίνει πως θα

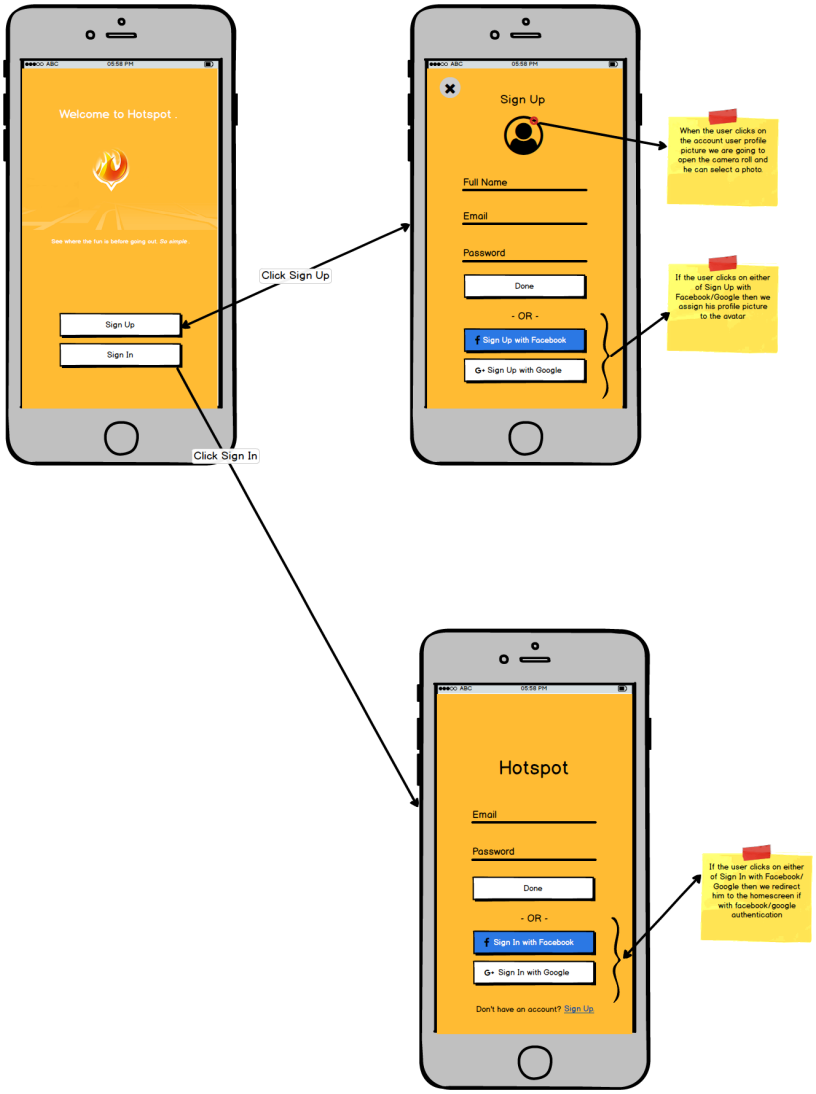
πρέπει να είναι φιλική όσον αφορά στη χρήση της, να μην αποτελείται από κουραστικά στοιχεία και περιττές λεπτομέρειες και να είναι σύντομη. Στο σχεδιασμό της φόρμας εγγραφής θα πρέπει να ληφθεί υπόψη το γεγονός ότι ο χρήστης δεν αντιδρά θετικά στις καθυστερήσεις και τις πρόσθετες υποχρεώσεις που δημιουργεί η διαδικσία εγγραφής στην εφαρμογή. Έτσι, ο χρόνος που απαιτείται για τη συμπλήρωση της φόρμας και την ολοκλήρωση της διαδικασίας εγγραφής θα πρέπει να είναι ο ελάχιστος δυνατός.

Συνεπώς, η οθόνη θα περιέχει μόνο τα απαραίτητα στοιχεία για την σωστή εγγραφή του χρήστη στην εφαρμογή. Η φόρμα εγγραφής θα είναι σύντομη αλλά επεξηγηματική, τα στοιχεία εισόδου θα είναι λιτά αλλά ευδιάκριτα και το πλήκτρο υποβολής θα είναι κατάλληλου μεγέθους και εύκολο στη χρήση. Κάθε στοιχείο εισόδου θα αντιστοιχίζεται με την κατάλληλη μορφή του πληκτρολογίου, ανάλογα με τη φύση του. Δηλαδή, εάν η είσοδος απαιτεί ηλεκτρονική διεύθυνση, τότε το πληκτρολόγιο θα παρέχει γρήγορη πρόσβαση στα απαραίτητα σύμβολα και αλφαριθμητικούς χαρακτήρες που απαιτούνται στην περίπτωση αυτή. Ο κωδικός πρόσβασης θα προστατεύεται αυτόματα και για λόγους αποφυγής προβλημάτων θα ζητείται η επιβεβαίωσή του.

Στην περίπτωση όπου υπάρχουν σφάλματα, μια διεπαφή σφαλμάτων θα εμφανίζει κατάλληλα μηνύματα λάθους στο χρήστη. Τα μηνύματα θα είναι σύντομα και σαφή και θα στοχεύουν κάθε λάθος ξεχωριστά για την καλύτερη αντιμετώπισή τους από την πλευρά του χρήστη. Με αυτό τον τρόπο, επιδιώκεται η γρήγορη και αποτελεσματική συμπλήρωση της φόρμας και η μετάβαση στο κύριο τμήμα της εφαρμογής.

4.1.4.3 LoginScreen

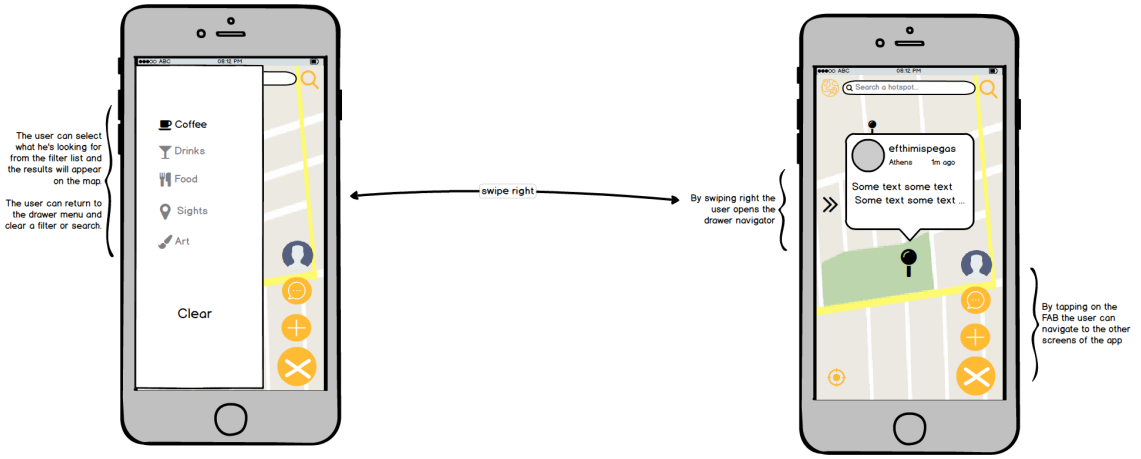
Η οθόνη σύνδεσης είναι πολύ πιο απλή από την οθόνη εγγραφής και έχει σκοπό την γρήγορη είσοδο του χρήστη στην εφαρμογή. Για το λόγο αυτό, περιέχει μόνο τις εισόδους για τα διαπιστευτήρια του χρήστη. Ο χρήστης στη φόρμα σύνδεσης καλείται να παραθέσει το συνδυασμό ηλεκτρονικής διεύθυνσης και κωδικού πρόσβασης. Στην περίπτωση που ο παραπάνω συνδυασμός δεν είναι σωστός, η φόρμα ανταποκρίνεται με κατάλληλο μήνυμα λάθους, μέσω μιας διεπαφής σφαλμάτων παρόμοια με αυτή της φόρμας εγγραφής.



Σχήμα 4.3: Αρχικό σχέδιο της οθόνης έναρξης και των οθονών εγγραφής/σύνδεσης.

4.1.4.4 HomeScreen

Η κεντρική οθόνη αποτελεί τον πυρήνα της εφαρμογής. Για το λόγο αυτό, επιβάλλεται η σχεδίασή της να είναι σφαιρική. Αυτό σημαίνει πως ο χρήστης θα πρέπει να μπορεί να αποκτήσει πρόσβαση σε όλες τις υπόλοιπες οθόνες από το σημείο αυτό. Οι περισσότερες λειτουργικότητες ξεκινούν από την οθόνη αυτή, πράγμα που καθιστά την απόκτηση εμπειρίας από την πλευρά του χρήστη εύκολη και άμεση.



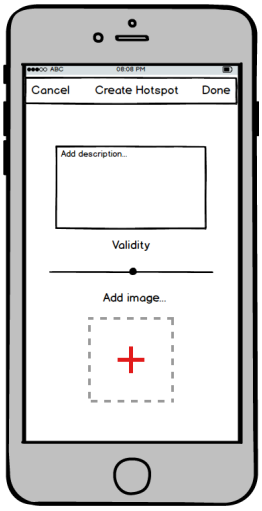
Σχήμα 4.4: Αρχικό σχέδιο της κεντρικής οθόνης.

Η HomeScreen περιέχει περισσότερες λειτουργικότητες από κάθε άλλη οθόνη. Προκειμένου ο χρήστης να μπορεί να πλοηγηθεί στις υπόλοιπες οθόνες, θα διαθέτει ένα διακριτικό αλλά κομψό πλήκτρο στο κάτω δεξιά μέρος της οθόνης. Με το πάτημά του θα αναδύονται επιμέρους πλήκτρα με εικονίδια που αντιστοιχούν στις υπόλοιπες οθόνες της εφαρμογής. Για την αναζήτηση συγκεκριμένων σημείων ενδιαφέροντος, ο χρήστης θα εισάγει τις λέξεις-κλειδιά στην μπάρα αναζήτησης στο πάνω μέρος της οθόνης και έπειτα θα μπορεί είτε να επιλέξει ένα από τα προτεινόμενα αποτελέσματα, είτε να κάνει γενικευμένη αναζήτηση μέσω του πλήκτρου αναζήτησης. Επιπρόσθετα φίλτρα αναζήτησης θα εμφανίζονται όταν ο χρήστης σύρει το αριστερό άκρο της οθόνης προς τα δεξιά, ή πατάει στο εικονίδιο που βρίσκεται στο επάνω αριστερά μέρος της οθόνης. Προηγούμενες αναζητήσεις ή εφαρμοσμένα φίλτρα θα αναιρούνται επιλέγοντας το πλήκτρο καθαρισμού από το ίδιο μενού. Τέλος, ο χρήστης θα μπορεί να επαναφέρει τη θέση του χάρτη στην τοποθεσία του μέσω πλήκτρου για το σκοπό αυτό που θα βρίσκεται στο κάτω αριστερά μέρος της οθόνης.

Από τα παραπάνω γίνεται αντιληπτή η αυξημένη πολυπλοκότητα που συνοδεύει τη σχεδίαση της HomeScreen. Ωστόσο, είναι κρίσιμο ο χρήστης να μην οδηγείται σε αυτό το συμπέρασμα κατά τη χρήση. Κάθε βήμα στη σχεδίαση καθεμιάς από τις προαναφερθείσες λειτουργικότητες θα πρέπει λοιπόν να χαρακτηρίζεται από απλότητα και να είναι βέλτιστη. Αυτό επιτυγχάνεται με τη χρήση απλών και ευδιάκριτων εικονιδίων, την προκαταβολική εκτέλεση πολλαπλών λειτουργιών σε προηγούμενο χρονικό σημείο και την υλοποίηση κάθε διεπαφής με εξελιγμένες βιβλιοθήκες και εργαλεία που χρησιμοποιούνται από τις πιο καταξιωμένες εφαρμογές σήμερα. Με αυτό τον τρόπο, δίνεται στο χρήστη η εντύπωση της αποδοτικής και ομαλής λειτουργίας της εφαρμογής, πράγμα που αποτελεί και τον τελικό στόχο.

4.1.4.5 CreateHotspotScreen

Όταν ο χρήστης θέλει να δημιουργήσει ένα νέο hotspot, μπορεί να το κάνει πηγαίνοντας στην CreateHotspotScreen. Η οθόνη δημιουργίας νέου hotspot εξυπηρετεί έναν αποκλειστικό σκοπό. Για το λόγο αυτό, στη σχεδιασή της συμπεριλήφθησαν στοιχεία που αφορούν στις ανάγκες της συγκεκριμένης οθόνης.



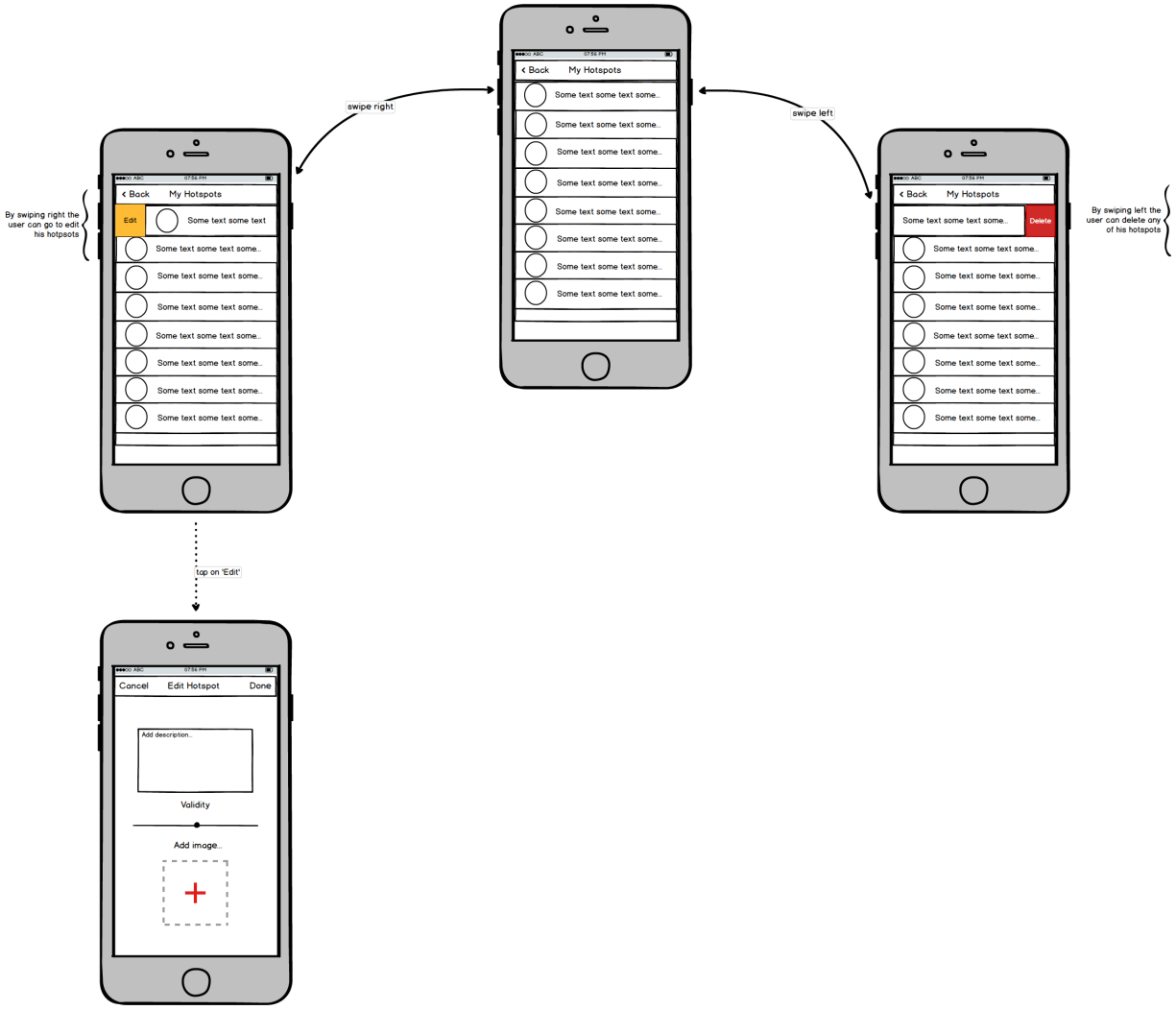
Σχήμα 4.5: Αρχικό σχέδιο της οθόνης δημιουργίας νέου hotspot.

Οι προϋποθέσεις που πρέπει να πληροί η οθόνη για να είναι πλήρως λειτουργική και να συναντάει όλες τις απαιτήσεις, είναι η φόρμα δημιουργίας νέου hotspot, η διεπαφή για την ενσωμάτωση αρχείου εικόνας και τα βασικά πλήκτρα ενεργειών και πλοήγησης. Η φόρμα βρίσκεται σε απλουστευμένη μορφή και αποτελείται από ένα στοιχείο εισόδου που αντιστοιχεί στην περιγραφή του hotspot προς δημιουργία, και ένα slider που αντιστοιχεί στο χρονικό διαστημα για το οποίο θα είναι έγκυρο το hotspot.

Όπως και κάθε οθόνη που δέχεται είσοδο από το χρήστη, έτσι και εδώ, απαιτείται κατάλληλος έλεγχος. Η είσοδος του χρήστη θα συνοδεύεται από διεπαφή μηνυμάτων κατάλληλης μορφής. Στην περίπτωση λάθους, η διεπαφή μηνυμάτων θα εμφανίζει στο χρήστη επεξηγηματικό μήνυμα λάθους, που θα προσδιορίζει το σημείο και το είδος του σφάλματος που έχει δημιουργηθεί. Εάν δεν υπάρχουν σφάλματα, τότε η υποβολή του νέου hotspot γίνεται με επιτυχία, και ο χρήστης επαναφέρεται στο χάρτη. Το καινούριο hotspot φαίνεται στο χάρτη, στην τοποθεσία του χρήστη.

4.1.4.6 HotspotListScreen

Προκειμένου να γίνει περισσότερο λειτουργική, η εφαρμογή ταξινομεί τα προσωπικά hotspots του χρήστη. Πηγαίνοντας στη HotspotListScreen, ο χρήστης μπορεί να εξευρευνήσει τη λίστα με όλες τις δημοσιεύσεις του. Σε αυτή την οθόνη ο χρήστης μπορεί να διαχειριστεί τα προσωπικά του hotspots σε μόλις λίγα βήματα. Σύροντας την οθόνη προς τα επάνω, ο χρήστης μπορεί να περιηγηθεί στη σελίδα και να δει περισσότερα hotspots, καθώς και σχετικές πληροφορίες. Η λίστα είναι ταξινομημένη με βάση την ημερομηνία δημιουργίας. Έτσι, το πιο πρόσφατο hotspot βρίσκται στην κορυφή, ενώ το παλαιότερο βρίσκεται στο τέλος της λίστας. Επίσης, για την γρηγορότερη φόρτωση και εμφάνιση των δεδομένων της οθόνης, η εφαρμογή φορτώνει τα πρώτα επτά hotspots του χρήστη. Έπειτα, ο χρήστης μπορεί να σύρει την οθόνη προς τα επάνω και να πατήσει το πλήκτρο εμφάνισης περισσότερων αποτελεμάτων με την επιγραφή '*show more*'.



Σχήμα 4.6: Αρχικό σχέδιο της οθόνης προσωπικών hotspots του χρήστη.

Στην οθόνη προσωπικών hotspots, Ο χρήστης έχει τη δυνατότητα να πραγματοποιήσει αλλαγές στα δικά του hotspots. Η λίστα είναι εφοδιασμένη με κρυφές λειτουργικότητες τις οποίες μπορεί να αξιοποιήσει ο χρήστης σύροντας ένα στοιχείο της λίστας προς μια οριζόντια κατεύθυνση. Για την επεξεργασία ενός hotspot, ο χρήστης σύρει ένα hotspot προς τα δεξιά. Τότε, πατώντας το πλήκτρο επεξεργασίας με την επιγραφή '*edit*', η εφαρμογή μεταφέρει το χρήστη στη σελίδα επεξεργασίας του επιλεγμένου hotspot. Εκεί, ο χρήστης μπορεί να πραγματοποιήσει τις επιθυμητές αλλαγές στην περιγραφή ή την εικόνα του hotspot. Για την διαγραφή ενός hotspot, ο χρήστης σύρει ένα hotspot προς τα αριστερά. Τότε, πατώντας το πλήκτρο διαγραφής με το εικονίδιο '*delete*', η εφαρμογή διαγράφει το επιλεγμένο hotspot από τη λίστα.

Η HotspotListScreen εξυπηρετεί την ανάγκη του χρήστη να ελέγχει την κατάσταση των προσωπικών του hotspots. Είναι σχεδιασμένη έτσι ώστε να παρέχει στο χρήστη τις επιθυμητές πληροφορίες για τα hotspots του, χωρίς να χρειάζεται να μεταβεί σε άλλη σελίδα. Για παράδειγμα, μπορεί γρήγορα να δει τον αριθμό των views και των comments ενός hotspot. Εάν

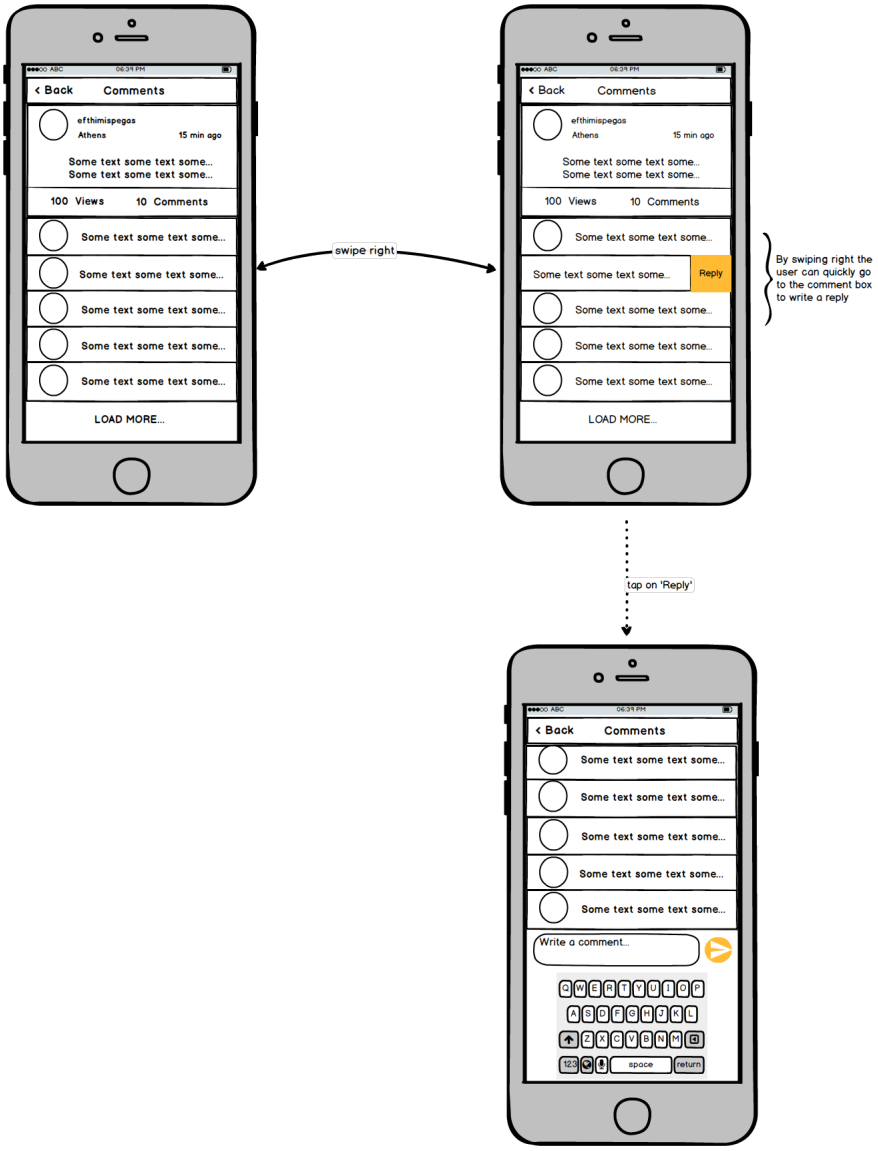
ο χρήστης επιθυμεί να δεί αναλυτικές πληροφορίες για ένα συγκεκριμένο hotspot, τότε μπορεί να πατήσει στο εικονίδιο σχολίων με την επιγραφή '*comments*'. Στη συνέχεια, η εφαρμογή μεταβαίνει στην οθόνη σχολίων CommentScreen του επιλεγμένου hotspot. Περισσότερα για την οθόνη αυτή αναφέρονται στην επόμενη παράγραφο.

4.1.4.7 CommentScreen

Για μια πιο λεπτομερή περιήγηση στις πληροφορίες σχετικά με κάποιο hotspot, ο χρήστης μπορεί να μεταβεί στην CommentScreen. Εκεί βρίσκονται όλα τα δεδομένα που αφορούν ένα συγκεκριμένο hotspot, όπως ο αριθμός των comments και των views, σχόλια άλλων χρηστών στο συγκεκριμένο hotspot, απαντήσεις πάνω σε σχόλια και φωτογραφία του hotspot, αν υπάρχει.

Η οθόνη αυτή συμπεριλαμβάνει όλα εκείνα τα στοιχεία που έχουν να κάνουν με την αλληλεπίδραση των χρηστών με ένα συγκεκριμένο hotspot. Εάν ο χρήστης επιθυμεί να γράψει ένα σχολιασμό ή να απαντήσει σε κάποιο σχόλιο, μπορεί να το κάνει εδώ. Σύροντας την οθόνη προς τα επάνω, ο χρήστης μπορεί χειροκίνητα να μεταφερθεί στο τέλος της λίστας με τα σχόλια, όπου βρίσκεται η διεπαφή δημιουργίας νέου σχολίου. Ένας άλλος τρόπος να επιτευχθεί αυτό είναι σύροντας κάποιο σχόλιο προς τα αριστερά και πατώντας στο πλήκτρο '*reply*'. Η εφαρμογή τότε θα μεταφέρει αυτομάτως το χρήστη στη διεπαφή δημιουργίας νέου σχολίου. Η διεπαφή είναι σχεδιασμένη ώστε να είναι ευδιάκριτη και μοντέρνα, αλλά ταυτοχρόνως απλή. Το τμήμα δημιουργίας νέου σχολίου συνοδεύεται και από το πλήκτρο υποβολής του νέου σχολίου, το οποίο φέρει το χαρακτηριστικό εικονίδιο '*send*'.

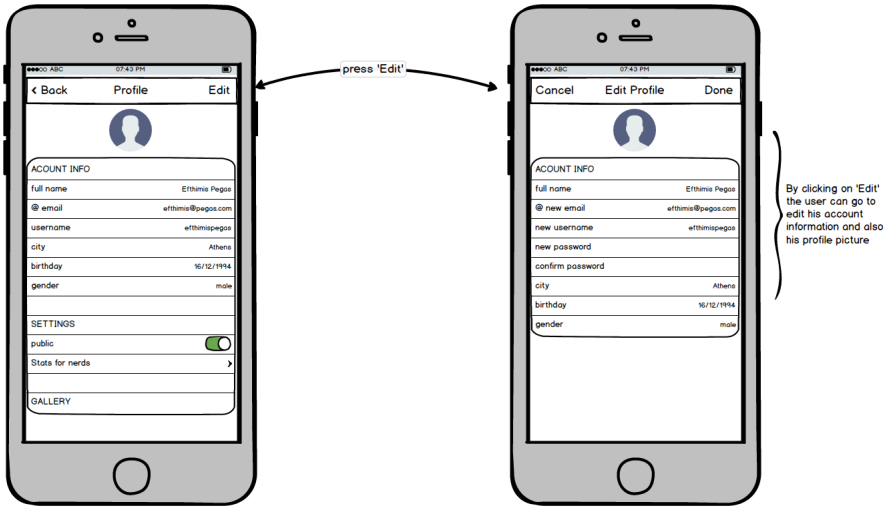
Η οθόνη είναι σχεδιασμένη ώστε να είναι αποδοτική και γρήγορη. Η λίστα με τα σχόλια θα πρέπει να φορτώνεται άμεσα. Η εμπειρία που προσφέρεται στο χρήστη θα πρέπει να είναι η καλύτερη δυνατή. Συνεπώς, η καθυστέρηση φόρτωσης θα πρέπει να είναι η ελάχιστη δυνατή. Η οθόνη είναι σχεδιαμένη με τέτοιο τρόπο, ώστε αν η λίστα με τα σχόλια είναι αρκετά μεγάλη, τότε να φορτώνεται μόνο ένας ορισμένος αριθμός σχολίων την πρώτη φορά. Εάν ο χρήστης επιθυμεί αν φορτώσει περισσότερα σχόλια, τότε μπορεί να περιηγηθεί στο τέλος της λίστας σύροντας την οθόνη προς τα επάνω και να πατήσει το πλήκτρο με την επιγραφή '*load more*'. Με την τμηματοποίηση της λίστας σχολίων, αποφεύγεται η περίπτωση μεγάλης καθυστέρησης φόρτωσης λόγω τεράστιου όγκου δεδομένων.



Σχήμα 4.7: Αρχικό σχέδιο της οθόνης σχολιασμών επάνω σε κάποιο hotspot.

4.1.4.8 ProfileScreen

Ο προσωπικός λογαριασμός του χρήστη είναι ένα κρίσιμο σημείο της εφαρμογής και έχει δοθεί ιδιαίτερη προσοχή στη σχεδίασή του. Ο χρήστης θα πρέπει να μπορεί, από τη μία να προβάλλει τα προσωπικά του στοιχεία και από την άλλη να επεξεργαστεί τις προσωπικές του ρυθμίσεις ή να αλλάξει το profile του. Εδώ ο χρήστης μπορεί να δει επίσης δεδομένα χρήσης της εφαρμογής, όπως ποσοστά αλληλεπίδρασης με άλλους χρήστες, συνολικό αριθμό σχολίων και προβολών, καθώς επίσης και το είδος του κοινού με το οποίο ο χρήστης αλληλεπιδρά περισσότερο. Στόχος της οθόνης profile είναι να ολοκληρώσει τη θετική εικόνα του χρήστη για την εφαρμογή και να αυξήσει το ενδιαφέρον του για αυτή.



Σχήμα 4.8: Αρχικό σχέδιο της οθόνης προσωπικού λογαριασμού του χρήστη.

Οι πληροφορίες που προβάλλονται στην ProfileScreen διακρίνονται σε τρεις κατηγορίες. Ταξινομημένα πρώτα είναι τα προσωπικά στοιχεία του χρήστη, με τα οποία συπλήρωσε τη φόρμα εγγραφής την πρώτη φορά που εισήλθε στην εφαρμογή. Έπειτα ακολουθούνε οι προσωπικές ρυθμίσεις του χρήστη, όπως το είδος του λογαριασμού που έχει δημιουργήσει (public ή private). Εάν ο χρήστης το επιθυμεί, μπορεί να προβάλλει τα στατιστικά δεδομένα που βρίσκονται σε ξεχωριστή σελίδα, πατώντας το βέλος που βρίσκεται στα δεξιά της επιγραφής '*Stats for nerds*'. Τέλος, στο κάτω μέρος της οθόνης βρίσκεται η συλλογή φωτογραφιών του χρήστη από τα προσωπικά hotspots που έχει δημιουργήσει ο ίδιος στο παρελθόν.

Για την επεξεργασία των προσωπικών του στοιχείων, ο χρήστης μπορεί να μεταβεί στην σελίδα επεξεργασίας profile. Πατώντας το πλήκτρο με την επιγραφή '*Edit*' που βρίσκεται στο δεξί μέρος της επικεφαλίδας, ο χρήστης μεταφέρεται στην EditProfileScreen. Εκεί, ο χρήστης μπορεί να επεξεργαστεί τα προσωπικά του στοιχεία, όπως το όνοματεπώνυμό του, την ηλεκτρονική του διεύθυνση, το όνομα χρήστη του, την εικόνα προφίλ του κλπ. Ακόμη, είναι εφικτή η αλλαγή των ρυθμίσεων του χρήστη, όπως το είδος του λογαριασμού. Για την επιτυχή αλλαγή των στοιχείων του ο χρήστης πρέπει να συμπληρώσει τον κωδικό πρόσβασης. Η φόρμα επεξεργασίας των προσωπικών δεδομένων του χρήστη συνοδεύεται από μια διεπαφή μηνυμάτων λάθους, όπως και κάθε οθόνη που απαιτεί είσοδο από το χρήστη. Στην περίπτωση όπου ένα ή περισσότερα πεδία είναι εσφαλμένα, κατάλληλο μήνυμα λάθους εμφανίζεται κάτω από το αντίστοιχο πεδίο. Στην περίπτωση όπου δεν υπάρχουν σφάλματα, η υποβολή της φόρμας ολοκληρώνεται με επιτυχία και ο χρήστης μεταφέρεται πίσω στον προσωπικό του λογαριασμό. Τα νέα στοιχεία του χρήστη φαίνονται στην οθόνη.

Χαρακτηριστικό της εφαρμογής αποτελεί το γεγονός ότι δεν είναι εύκολη η αποσύνδεση του χρήστη από την εφαρμογή. Αυτό οφείλεται στο γεγονός ότι το πλήκτρο αποσύνδεσης δεν υπάρχει σε καμία από τις προηγούμενες οθόνες. Έτσι, ο χρήστης δεν μπορεί να αποσυνδεθεί παρά μόνο όταν έχει ολοκληρώσει την περιήγησή του εντός της εφαρμογής και έχει αποκτήσει μια πλήρη εικόνα των υπηρεσιών που προσφέρει. Για την αποσύνδεση από την εφαρμογή, ο χρήστης μεταβαίνει στον προσωπικό του λογαριασμό και πατάει το πλήκτρο αποσύνδεσης

που φέρει την επιγραφή '*Log out*'. Έπειτα, μεταφέρεται αυτόματα στην οθόνη έναρξης της εφαρμογής.

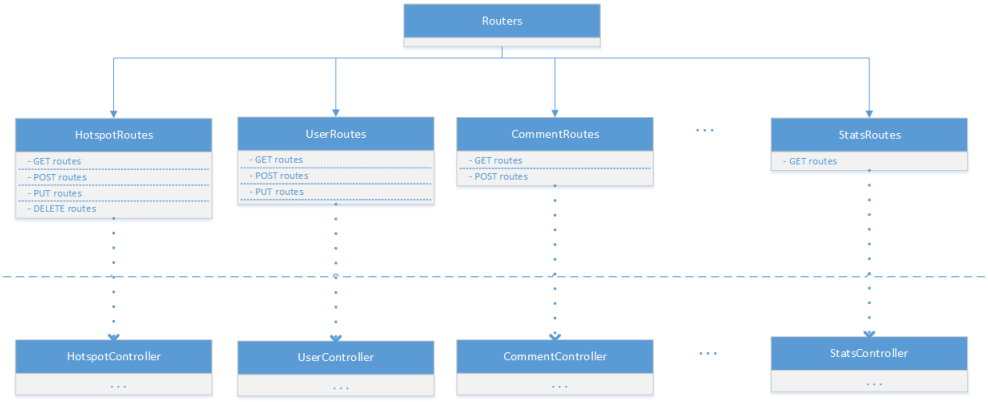
Είναι καθήκον της λογικής σχεδίασης της εφαρμογής να ενθαρρύνει συνεχώς την αλληλεπίδραση του χρήστη με τις υπηρεσίες της. Όπως φαίνεται από τα παραπάνω, αυτό επιτυγχάνεται τόσο μέσα από την παροχή πληθώρας λειτουργικοτήτων, όσο και μέσω της παρουσίασης στατιστικών δεδομένων που αφορούν τη σχέση του χρήστη με το υπόλοιπο σύνολο και δημιουργούν μια εικόνα της θέσης του χρήστη εντός του οικοσυστήματος της εφαρμογής. Αυτό συνεισφέρει στη συνεχή δράση του χρήστη και την ανάπτυξη στενών σχέσεων με την εφαρμογή.

4.2 Σχεδίαση Συστήματος Εξυπηρέτησης Αιτημάτων

Στην προηγούμενη ενότητα καλύφθηκε η λογική που ακολουθήθηκε στη σχεδίαση του client της εφαρμογής. Αναλύθηκαν οι δομές που απαρτίζουν τις διάφορες διεπιφάνειες χρήστη, και παρουσιάστηκαν τα αρχικά σχέδια αυτών. Σε αυτή την ενότητα θα αναπτυχθούν οι τεχνικές σχεδίασης του server και των υποσυστημάτων του.

Στην σχεδίαση του εξυπηρετητή αιτημάτων της εφαρμογής, πρωταρχικός σκοπός είναι η αποσύμπλεξη των λειτουργιών που δημιουργεί ένα αίτημα. Κάθε ενέργεια που λαμβάνει χώρα σε μια από τις διεπιφάνειες χρήστη, πυροδοτεί μια σειρά κλήσεων που απαιτούν τη μεταφορά δεδομένων από και προς τον server. Ο server έχει σχεδιαστεί με τέτοιο τρόπο, ώστε κάθε αίτημα να αποσυναρμολογείται στις επιμέρους διακριτές λειτουργίες που πρέπει να εκτελεστούν. Οι λειτουργίες αυτές δρομολογούνται με τη σειρά τους στα κατάλληλα υποσυστήματα για την εξυπηρέτησή τους.

4.2.1 Λογική Σχεδίασης Εξυπηρετητή και Υποσυστήματα



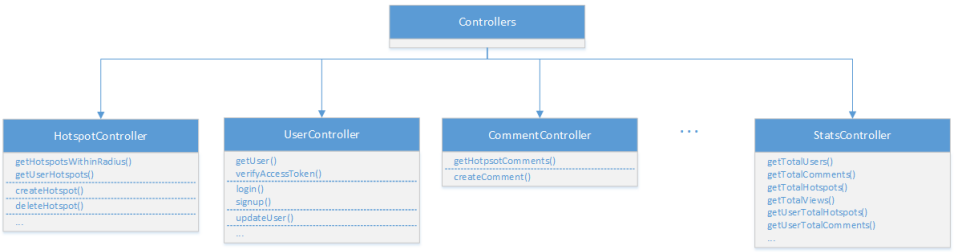
Σχήμα 4.9: Λογική σχεδίασης του συστήματος δρομολόγησης αιτημάτων.

Ο server της εφαρμογής θα συνίσταται από ένα σύστημα ελεγκτών (controllers) και δρομολογητών (routers), με σκοπό την αποτελεσματική και γρήγορη ικανοποίηση των αιτημάτων που δημιουργεί ο client. Κάθε δρομολογητής θα είναι υπεύθυνος για την σωστή ταξινόμηση

των αιτημάτων με βάση τη φύση τους και την δρομολόγηση στον κατάλληλο ελεγκτή. Έπειτα, ο ελεγκτής θα εκτελεί μια σειρά λειτουργιών που αντιστοιχούν στο συγκεκριμένο αίτημα που δημιουργήθηκε. Μόλις ολοκληρωθεί η παραπάνω διαδικασία, το αποτέλεσμα επιστρέφεται σε κατάλληλη μορφή πίσω στον client.

Το σύστημα δρομολόγησης των αιτημάτων διαιρείται σε επιμέρους routers (βλ. Σχ. 4.9). Για την καλύτερο προγραμματισμό του συστήματος και την απλοποίηση της σχεδίασης, επιλέχθηκε ως κριτήριο διάκρισης η φύση των λειτουργιών που εξυπηρετεί κάθε route (περισσότερα στην ενότητα 5.2.1).

Το σύστημα εξυπηρέτησης των αιτημάτων υλοποιείται από controllers που καθορίζουν τις εντολές που θα εκτελεστούν για την ολοκλήρωση κάθε αιτήματος (βλ. Σχ. 4.10). Οι εντολές συνοψίζονται σε μεθόδους, γνωστές ως ελεγκτές. Κάθε ελεγκτής αντιστοιχίζεται στην εξυπηρέτηση ενός αποκλειστικού αιτήματος. Το σύστημα των ελεγκτών είναι σχεδιασμένο ώστε να γίνεται κατανοητό από τον προγραμματιστή και ευέλικτο όσον αφορά στην επεξεργασία του. Όπως και στο σύστημα των δρομολογητών, έτσι και εδώ, κριτήριο στην διακριτοποίηση των ελεγκτών αποτελεί η φύση του αιτήματος (περισσότερα στην ενότητα 5.2.2).



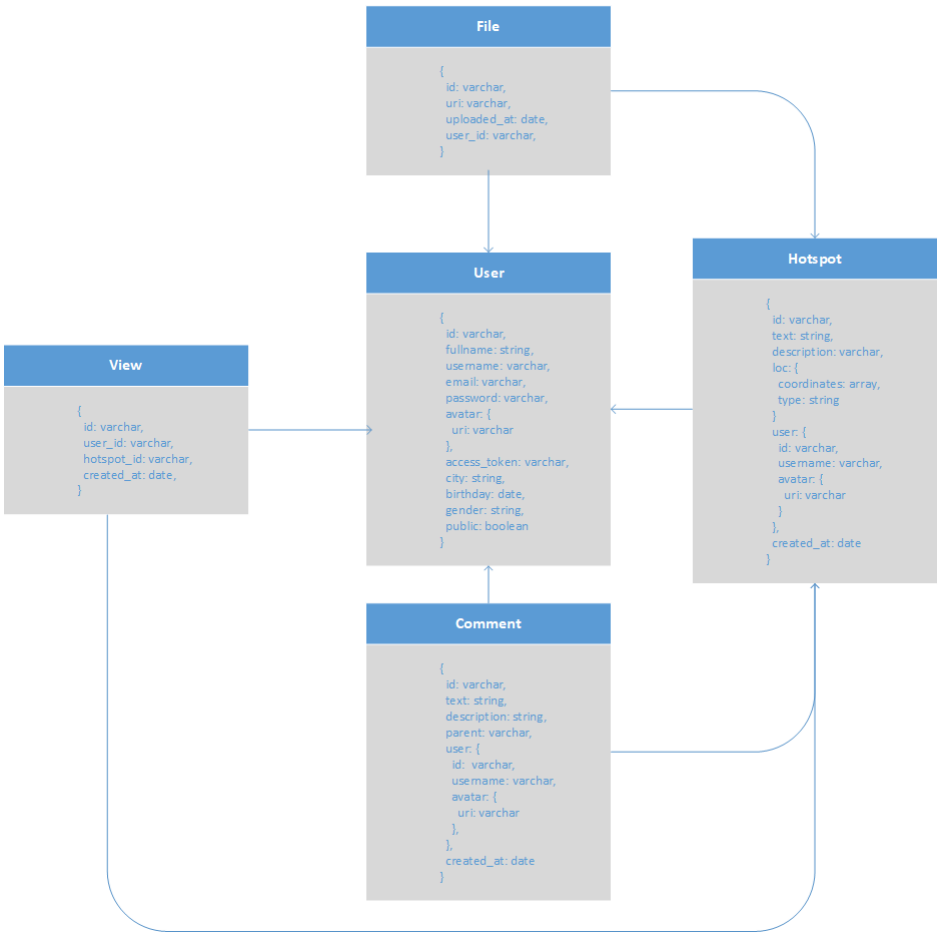
Σχήμα 4.10: Λογική σχεδίασης του συστήματος ελεγκτών στον εξυπηρετητή αιτημάτων της εφαρμογής.

Από τα παραπάνω λοιπόν, γίνεται αντιληπτή η λογική πορεία εξυπηρέτησης αιτημάτων στην οποία βασίζεται ο server. Ωστόσο, κατά την πορεία αυτή γίνεται πρόσβαση σε δεδομένα τα οποία είναι αποθηκευμένα σε έναν ειδικά διαμορφωμένο χώρο. Ο αποθηκευτικός αυτός χώρος αποτελεί τη βάση δεδομένων της εφαρμογής και αποτελεί ίσως το σημαντικότερο κομμάτι του backend. Στην ενότητα που ακολουθεί θα αναλυθούν οι βασικές σχεδιαστικές τεχνικές που διέπουν τη βάση δεδομένων της εφαρμογής, καθώς επίσης και τα μοντέλα στα οποία οργανώνονται τα δεδομένα της εφαρμογής.

4.3 Σχεδίαση Μοντέλου Βάσης Δεδομένων

Τα πολλά αιτήματα μεταξύ client-server δημιουργούν την ανάγκη για συνεχείς προσπελάσεις της βάσης δεδομένων. Συνεπώς, η δομή που θα ακολουθεί η βάση θα πρέπει να είναι σχεδιασμένη με τέτοιο τρόπο, ώστε να ευνοεί τη συνεχή πρόσβαση σε αυτή. Για να επιτευχθεί αυτό, θα πρέπει η μοντελοποίηση των δεδομένων να γίνεται με όσο το δυνατόν λιγότερες εξαρτήσεις και με τη χρήση απλών δομών για υψηλή απόδοση.

4.3.1 Μοντελοποίηση Δεδομένων



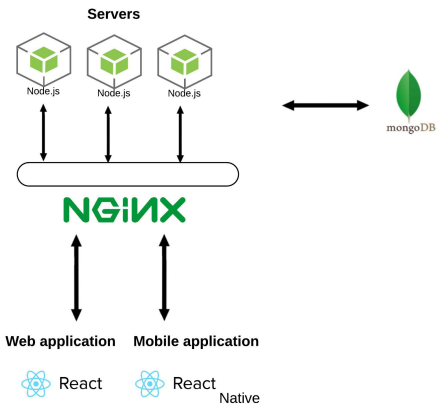
Σχήμα 4.11: Σχέδιο μοντελοποίσης των δεδομένων της εφαρμογής.

Όπως αναφέρθηκε στην ενότητα 3.3.2, τα δεδομένα της εφαρμογής μοντελοποιούνται σε collections. Η εσωτερική διάρθρωση κάθε τέτοιου μοντέλου περιλαμβάνει ορισμένες ιδιότητες και κάθε μοντέλο που δημιουργείται θα πρέπει να τις ακολουθεί αυστηρά. Κάθε μοντέλο διακρίνεται από τα υπόλοιπα της ίδιας συλλογής μέσω του αναγνωριστικού ID το οποίο προσδίδεται αυτόματα σε αυτό. Επίσης, υπάρχει η πιθανότητα κάποια ιδιότητα να είναι μοναδική για κάθε μοντέλο (πχ. η ηλεκτρονική διεύθυνση στο μοντέλο χρήστη). Αυτό επιτυγχάνεται με τον ορισμό ενός δείκτη μοναδικότητας (uniqueness index) στην συγκεκριμένη ιδιότητα του μοντέλου. Τα νέα μοντέλα αποθηκεύονται στην αντίστοιχη συλλογή για μετέπειτα χρήση από την εφαρμογή. Στο σχ. 4.11 φαίνεται η δομή των μοντέλων της εφαρμογής, καθώς και οι χαρακτηριστικές ιδιότητες του κάθε μοντέλου.

Η αναπαράσταση των δεδομένων με αυτό τον τρόπο, δίνει στον server τη δυνατότητα να προσπελαύνει τη βάση δεδομένων με μεγάλη ταχύτητα. Έτσι, η αναζήτηση δεδομένων γίνεται σε μικρό χρονικό διάστημα και η απόδοση της εφαρμογής είναι η βέλτιστη δυνατή. Το ζητούμενο αυτό είναι απότα κυριότερα όταν πρόκειται για την σχεδίαση μιας εφαρμογής με πολλούς χρήστες, μιας και δημιουργούνται συνεχώς αιτήματα για πρόσβαση στη βάση δεδομένων. Η

ύπαρξη απλοποιημένων μοντέλων βοηθάει στην γρήγορη εξυπηρέτηση των παραπάνω αιτημάτων και δίνει τη δυνατότητα παραλληλοποίησης των διαδικασιών.

Καθοριστικό ρόλο στην ταχύτητα πρόσβασης στη βάση δεδομένων παίζει και η επιλογή της τεχνολογίας. Για το λόγο αυτό έχει επιλεχθεί η MongoDB ως το είδος της βάσης που θα χρησιμοποιηθεί στην υλοποίηση. Η MongoDB επικεντρώνεται στην εξυπηρέτηση πολλαπλών αιτημάτων που απαιτούν ταυτόχρονη πρόσβαση σε δεδομένα της βάσης, αποτελόντας συνεπώς και την καλύτερη λύση (βλ. σχ. 4.12).



Σχήμα 4.12: Φιλοσοφία πίσω από τη σχεδίαση της MongoDB.

Κεφάλαιο 5

Υλοποίηση Συστήματος

Μέχρι στιγμής, έχει γίνει αναφορά στις τεχνολογίες και τις τεχνικές σχεδίασης της εφαρμογής. Το κεφάλαιο αυτό επεκτείνεται στην υλοποίηση της εφαρμογής, αναλύοντας τις μεθόδους που χρησιμοποίηθηκαν για τον προγραμματισμό τόσο του frontend, όσο και του backend. Η ενότητα 5.1 αφορά το frontend κομμάτι, δηλαδή με τον προγραμματισμό των διεπιφανειών χρήστη. Η ενότητα 5.2 καταπιάνεται με την υλοποίηση του backend, το οποίο απαρτίζεται από τον εξυπηρετητή αιτημάτων που δημιουργούνται από τις διεπιφάνειες χρήστη και τη βάση δεδομένων στην οποία αποθηκεύονται όλες οι πληροφορίες της εφαρμογής. Στην ενότητα 5.3 παρουσιάζονται τα τεχνολογικά εργαλεία τα οποία χρειάστηκαν στην υλοποίηση. Τέλος, στην ενότητα 5.4 γίνεται μια συνοπτική αναφορά στην πλοτφόρμα προγραμματισμού που επιλέχθηκε για την υλοποίηση της εφαρμογής.

5.1 Λεπτομέρειες Υλοποίησης Διεπιφανειών Χρήστη

Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι τεχνικές που χρησιμοποιήθηκαν στην υλοποίηση του συστήματος διεπιφανειών χρήστη. Η εφαρμογή αφορά κινητές συσκευές με λειτουργικό iOS, επομένως δόθηκε μεγάλη βαρύτητα σε μεθόδους που υιοθετούν οι περισσότερες native εφαρμογές της ίδιας κατηγορίας. Κατά τον προγραμματισμό, σχεδόν όλες οι διεπαφές που χρησιμοποιούνται αφορούν αποκλειστικά iOS πλατφόρμες προκειμένου να προσφέρουν την καλύτερη δυνατή εμπειρία στο χρήστη.

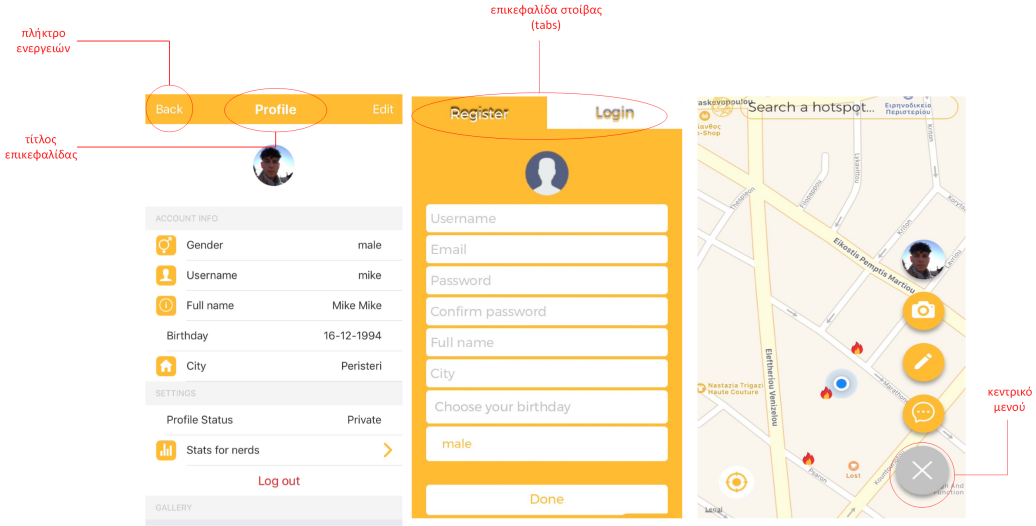
Η εφαρμογή αναφέρεται σε κινητά τελευταίας γενιάς. Αυτό εισάγει κάποιους περιορισμούς στον προγραμματισμό, αλλά έχει ταυτοχρόνως και κάποια πλεονεκτήματα. Για παράδειγμα, οι διαστάσεις της οθόνης στις κινητές συσκευές είναι πολύ μικρότερες από αυτές ενός σταθερού υπολογιστή. Διαδραστικά στοιχεία όπως πλήκτρα, πεδία εισόδου, εικονίδια και άλλα γραφικά θα πρέπει να λαμβάνουν υπόψη τους τον παραπάνω περιορισμό. Συνεπώς, ο προγραμματισμός για mobile apps είναι σαφώς πιο δύσκολος από τον προγραμματισμό για desktop apps.

Το κυριότερο πλεονέκτημα του προγραμματισμού για να mobile apps είναι το γεγονός ότι δεν χρειάζεται να δοθεί ιδιαίτερη βάση σε λεπτομέρειες, αφού οι οθόνες των κινητών δεν επιτρέπουν την σχολαστική ενασχόληση με κάθε στοιχείο ή την υπερβολική λεπτομέρεια των διαδραστικών στοιχείων, αφού αυτό θα καθιστούσε την εμπειρία χρήσης της εφαρμογής

κουραστική για το χρήστη. Έτσι, ο προγραμματισμός επικεντρώνεται κυρίως γύρω από το κομμάτι με το οποίο θα αλληλιπιδρά ο χρήστης και τις τεχνικές λεπτομέρειες που συμπεριλαμβάνει. Για να εξασφαλιστεί μια ολοκληρωμένη, αλλά συγχρόνως απλή εμπειρία εντός της εφαρμογής, όλες οι διεπιφάνειες ακολουθούν αυτή την τακτική, όπου το διαδραστικό περιβάλλον αποτελείται αποκλειστικά από διεπαφές που είναι ζωτικές για την σωστή λειτουργία της κάθε οθόνης.

5.1.1 Βασικά Προγραμματιστικά Χαρακτηριστικά Οθονών

Βασική μέριμνα του προγραμματισμού των οθονών της εφαρμογής αποτέλεσε η χρήση οικουμενικών διεπαφών. Η δυνατότητα επαναληψιμότητας ορισμένων προγραμματιστικών τεχνικών, συνέβαλλε στην απλοποίηση της διαδικασίας υλοποίησης και την ευκολία χειρισμού των διεπιφανειών. Τα κυριότερα χαρακτηριστικά των διεπαφών αυτών είναι η χρήση ενός κεντρικού πλοηγητή για την περιήγηση στις οθόνες της εφαρμογής, η επικεφαλίδες με τίτλο για την κάθε οθόνη, η ύπαρξη ενός κεντρικού μενού πλοήγησης σελίδων και τα βασικά πλήκτρα ενεργειών στην επικεφαλίδα κάθε οθόνης (βλ. Σχ. 5.1). Καθένα από τα παραπάνω προγραμματιστικά χαρακτηριστικά αναλύεται στη συνέχεια.



Σχήμα 5.1: Παράδειγμα χαρακτηριστικών που επαναλαμβάνονται σε όλη την εφαρμογή.

5.1.1.1 Τεχνική Πλοήγησης στην Εφαρμογή

Ένα από τα βασικότερα σημεία της υλοποίησης αποτελεί η πλοήγηση εντός της εφαρμογής. Η τεχνική που ακολουθήθηκε βασίζεται στην ύπαρξη ενός ενιαίου router, ο οποίος είναι υπεύθυνος για την πλόγηση κάθε φορά στην επιλεγμένη σελίδα, αλλά και την αποθήκευση της σελίδας σε μια στοίβα. Με αυτό τον τρόπο, είναι δυνατή η εκτεταμένη περιήγηση σε περισσότερες από μια σελίδες της εφαρμογής και η επιστροφή σε οποιαδήποτε από τις προηγούμενες σελίδες χάρις την δυνατότητα διευθυνσιοδότητησης καθεμιάς από τις σελίδες αυτές.

Πρακτικά, αυτό επιτυγχάνεται με την τεχνική των εμφωλευμένων σελίδων. Σύμφωνα με

αυτή, οι σελίδες που πρόκειται να έχουν τη δυνατότητα επιστροφής σε κάποια προηγούμενη σελίδα θα εμφωλιάζονται ομαδικά υπό μια υποτιθέμενη '*σκηνή*' (scene). Κάθε σελίδα αποτελεί από μόνη της μια σκηνή και πολλές σκηνές μαζί δημιουργούν μια '*στοίβα*' από σκηνές (stack). Όσες σκηνές βρίσκονται εντός της στοίβας, μπορούν να διαδέχονται η μία την αμέσως επόμενη, έχοντας ταυτόχρονα την δυνατότητα επιστροφής στην προηγούμενη σελίδα-σκηνή. Επίσης, υιοθετούν μία σειρά από ιδιότητες στις οποίες θα γίνει αναφορά στις ενότητες που ακολουθούν.

Όλες οι στοίβες που θα δημιουργηθούν τελικά, είναι εμφωλευμένες στον router. Αν το τελευταίο δεν ισχύει, τότε οι παραπάνω ιδιότητες δεν εφαρμόζονται στις επιμέρους στοίβες.

5.1.1.2 Επικεφαλίδα Οθόνης

Μία από τις βασικότερες ίσως ιδιότητες που μεταδίδεται στις σκηνές που απαρτίζουν μια στοίβα είναι η επικεφαλίδα. Κάθε οθόνη που ανήκει στην ομάδα εμφωλευμένων σκηνών έχει την δυνατότητα να φέρει ή όχι κάποια χαρακτηριστικά στην επικεφαλίδα. Αυτά είναι η επιπλέον επικεφαλίδα της στοίβας, αποκλειστικές σχεδιαστικές επεμβάσεις (όπως πχ. χρώμα, είδος και μέγεθος γραμματοσειράς, εικονίδια κλπ.), ο τίτλος της παρούσας σελίδας, διαρρύθμιση πλήκτρων που θα έχει η επικεφαλίδα, αλλαγή θέσης των στοιχείων και πολλά άλλα. Η επικεφαλίδα αποτελεί ξεχωριστό component, το οποίο ανήκει σε μια ομάδα επαναχρησιμοποιήσιμων components (commons) και χρησιμοποιείται σε κάθε σελίδα με διαφορετικές διαρρυθμίσεις.

Οι παραπάνω ιδιότητες είναι ιδιαίτερα χρήσιμες στον προγραμματισμό κάθε οθόνης. Δίνουν πολλές επιλογές στον προγραμματιστή όσον αφορά στην διαμόρφωση της επικεφαλίδας της κάθε οθόνης και την ανεξαρτητοποίηση των οθονών μεταξύ τους από σχεδιαστικής απόψεως. Από την άλλη, μπορούν να αυτοματοποιηθούν πολλές προγραμματιστικές ενέργειες με την χρήση τμημάτων κώδικα που μπορεί να επαναχρησιμοποιηθεί σε όλες σχεδόν τις οθόνες. Αυτό καθιστά τη διαδικασία προγραμματσιμού αισθητά πιο εύκολη και βέλτιση.

5.1.1.3 Κεντρικό Μενού Οθονών

Στη δεξιότερη εικόνα του σχ. 5.1 φαίνεται το κεντρικό μενού για την πλοήγηση μεταξύ των οθονών της εφαρμογής. Η τεχνική χρήσης ενός κεντρικού πλήκτρου για όλες τις βασικές οθόνες της εφαρμογής επιλύει πολλά προβλήματα και καθιστά το κομμάτι του προγραμματισμού αρκετά πιο απλό. Το FAB εξυπηρετεί επίσης από άποψη χώρου, αφού δεν καταλαμβάνει μεγάλο τμήμα της κεντρικής οθόνης, πράγμα που το κάνει την καταλληλότερη προγραμματιστική τεχνική. Μια άλλη προσέγγιση θα ήταν η ύπαρξη μιας μπάρας εικονιδίων με τις κύριες οθόνες στο κάτω μέρος της οθόνης, αλλά αυτό θα μείωνε αρκετά τις διαστάσεις της διεπαφής χάρτη, που είναι άλλωστε και το κύριο χαρακτηριστικό της οθόνης.

Το FAB αποτελεί ξεχωριστό component της κεντρικής οθόνης και συνίσταται από το κύριο πλήκτρο επέκτασης του μενού και τρία επιμέρους εικονίδια. Κάθε εικονίδιο αντιστοιχεί και σε μία από τις βασικές οθόνες της εφαρμογής και για τον λόγο αυτό του έχει αποδοθεί κατάλληλο εικονίδιο.

5.1.1.4 Βασικά Πλήκτρα Ενεργειών

Όπως έχει ήδη αναφερθεί, κάθε οθόνη διαθέτει μια εξατομικευμένη επικεφαλίδα. Η επικεφαλίδα, εκτός των άλλων φέρει τον τίτλο της σελίδας στον οποίο βρίσκεται ο χρήστης και κάποια βασικά πλήκτρα ενεργειών. Τα πλήκτρα αυτά αποσκοπούν στην δυνατότητα πλοήγησης προς τα εμπρός ή προς τα πίσω σε μια στοίβα σελίδων. Για παράδειγμα, όταν ο χρήστης επισκέπτεται την οθόνη του προσωπικού του profile (ProfileScreen), μπορεί είτε να επιστρέψει στην προηγούμενη σελίδα πατώντας το πλήκτρο επιστροφής με την επιγραφή '*back*', είτε να πλοηγηθεί στην σελίδα επεξεργασίας των προσωπικών του στοιχείων (EditProfileScreen) πατώντας το πλήκτρο επιστροφής με την επιγραφή '*edit*'.

ο προγραμματισμός της λειτουργικότητας του κάθε πλήκτρου ενεργειών εξαρτάται από το είδος της σελίδας. Το πλήκτρο στα αριστερά μιας σελίδας έχει πάντοτε το ρόλο επιστροφής στην προηγούμενη σελίδα. Ανάλογα με τις ενέργειες που θα παρέχονται στον χρήστη, το πλήκτρο ενεργειών στα δεξιά της επικεφαλίδας θα εκτελεί είτε την λειτουργία επεξεργασίας δεδομένων, είτε τη λειτουργία υποβολής φόρμας. Στο παραπάνω παράδειγμα, η οθόνη στην οποία αναφέρονται τα πλήκτρα ενεργειών είναι αυτή του profile του χρήστη, επομένως το πλήκτρο στα αριστερά παίζει το ρόλο της επιστροφής στην προηγούμενη σελίδα, ενώ το πλήκτρο στα δεξιά μεταφέρει το χρήστη στην οθόνη επεξεργασίας. Εκεί έπειτα, τα πλήκτρα αναφέρονται σε μια οθόνη που περιέχει φόρμα, συνεπώς το πλήκτρο στα δεξιά έχει τώρα τον ρόλο υποβολής της φόρμας.

5.1.2 Λειτουργικότητα Οθονών Εφαρμογής

Στην ενότητα αυτή θα αναλυθούν οι μεθοδολογίες που ακολουθήθηκαν στην υλοποίηση των λειτουργικοτήτων της εφαρμογής. Όπως αναφέρθηκε ήδη στο Κεφ. 3, η εφαρμογή θα δίνει στο χρήστη τη δυνατότητα εγγραφής και σύνδεσης στην εφαρμογή, ως μέσο ταυτοποίησης. Ακόμη, ο χρήστης θα μπορεί να πλοηγείται σε ένα χάρτη στα διάφορα σημεία ενδιαφέροντος. Εκεί, θα μπορεί να επιλέξει από μια πληθώρα ενεργειών για να αλληλεπιδράσει με τα στοιχεία του χάρτη. Η εφαρμογή θα δίνει στο χρήστη τη δυνατότητα να εξατομικεύσει τον προσωπικό του λογαριασμό, μέσα από την ταξιθέτηση των προσωπικών του δημοσιεύσεων αλλά και από την επεξεργασία του προσωπικού του profile. Τέλος, ο χρήστης θα μπορεί να αποσυνδεθεί απο την εφαρμογή όποτε το επιθυμεί.

5.1.2.1 RegisterScreen & LoginScreen

Η οθόνη αποτελείται από δύο επιμέρους tabs, ένα για εγγραφή και ένα για σύνδεση των χρηστών. Ο χρήστης μπορεί να επιλέγει κάθε φορά το tab που αρμόζει στην στην κάθε περίπτωση. Εάν ο χρήστης επιθυμεί να πραγματοποιήσει εγγραφή στην εφαρμογή, μπορεί να επιλέξει το tab με τίτλο '*Register*', κάνοντας swipe προς τα δεξιά. Από την άλλη, εάν ο χρήστης θέλει να πραγματοποιήσει σύνδεση στον λογαριασμό του, μπορεί να επιλέξει το tab με τίτλο '*Login*', κάνοντας swipe προς τα αριστερά.

Κάθε οθόνη περιέχει μια φόρμα με πεδία τα οποία καλείται να συμπληρώσει ο χρήστης. Στην οθόνη εγγραφής, η φόρμα περιέχει πεδία για το πλήρες όνομα, το όνομα χρήστη, την ηλε-

κτρονική διεύθυνση, τον κωδικό πρόσβασης, την πόλη, το γένος και την ημερομηνία γέννησης του χρήστη. Επίσης, ο χρήστης μπορεί να επιλέξει μία από τις φωτογραφίες στη συλλογή του ή να βγάλει μια νέα φωτογραφία και να την θέσει ως εικόνα προφίλ στο λογαριασμό του. Η οθόνη σύνδεσης είναι πιο σύντομη και αποτελείται από δύο πεδία, τα οποία ο χρήστης συμπληρώνει με τα διαπιστευτήριά του. Ο συνδυασμός διαπιστευτηρίων χρήστη αποτελείται από την ηλεκτρονική διεύθυνση και των κωδικό πρόσβασης του χρήστη. Σε καθεμία εκ των ανωτέρω περιπτώσεων, ο χρήστης μεταφέρεται στο χάρτη της κεντρικής οθόνης της εφαρμογής μετά από επιτυχή υποβολή της φόρμας.

Κάθε φόρμα έχει υλοποιηθεί ώστε να προσφέρει την καλύτερη δυνατή εμπειρία χρήστη. Το είδος της εισόδου που απαιτείται στην συμπλήρωση κάθε φόρμας διαφέρει από πεδίο σε πεδίο. Για παράδειγμα, το πεδίο κωδικού πρόσβασης απαιτεί είσοδο αποτελούμενη από αλφαριθμητικούς χαρακτήρες, ενώ στο πεδίο της ηλεκτρονικής διεύθυνσης χρησιμοποιούνται συγκεκριμένοι ειδικοί χαρακτήρες και σύμβολα. Για το λόγο αυτό, έχει προσδιοριστεί το είδος κάθε πεδίου, με αποτέλεσμα την εμφάνιση του κατάλληλου είδους πληκτρολογίου κάθε φορά. Επίσης, το πλήκτρα υποβολής κάθε φόρμας αντικαθίστανται από spinners που δείχνουν στο χρήστη ότι η εφαρμογή φορτώνει.

Σε περίπτωση ανεπιτυχούς υποβολής κάποιας από τις παραπάνω φόρμες, η εφαρμογή ενημερώνει το χρήστη σχετικά με τη φύση του σφάλματος. Σφάλματα μπορούν να προκύψουν από την λανθασμένη συμπλήρωση ενός ή περισσότερων πεδίων εισόδου. Σε κάθε περίπτωση, ένα κατάλληλο μήνυμα λάθους εμφανίζεται κάτω από το αντίστοιχο πεδίο, το οποίο αναφέρει συνοπτικά τις μετατροπές που πρέπει να γίνουν ώστε να είναι αποδεκτή η είσοδος του χρήστη.

5.1.2.2 HomeScreen

Σε αυτό το σημείο της εφαρμογής εμφανίζονται και οι κατάλληλες ειδοποιήσεις στο χρήστη. Η εφαρμογή χρειάζεται την τοποθεσία του χρήστη για να την προσδιορίσει στο χάρτη και στη συνέχεια να φορτώσει τα hotspots. Δίνοντας την συγκατάνευσή του, ο χρήστης επιτρέπει στην εφαρμογή να πραγματοποιήσει τις αναγκαίες ρυθμίσεις για την ορθή λειτουργία της. Επίσης, ο χρήστης θα πρέπει να πραγματοποιήσει σύνδεση σε κάποιο δίκτυο προκειμένου να μπορέσει να χρησιμοποιήσει την εφαρμογή.

Μετά την είσοδό του στην εφαρμογή, ο χρήστης ανακατευθύνεται στον χάρτη. Ο χάρτης φορτώνει με τα hotspots που βρίσκονται εντός εμβέλειας της τοποθεσίας του χρήστη. Η εμβέλεια είναι προγραμματισμένη να έχει μία ακτίνα ίση με 5km. Τα hotspots που φαίνονται στο χάρτη και έχουν δημιουργηθεί από άλλους χρήστες συμβολίζονται με markers που έχουν τη μορφή φωτιάς συγκεκριμένου μεγέθους. Το μέγεθος κάθε marker ποικίλλει και εξαρτάται από τον αριθμό προβολών του hotspot. Οσο πιο πολλοί χρήστες έχουν δει ένα συγκεκριμένο hotspot, τόσο μεγαλύτερο είναι το μέγεθος της φωτιάς. Έτσι, δημιουργείται μια αναλογία μεταξύ των hotspots και του ενδιαφέροντος των χρηστών.

Ο χρήστης μπορεί να επιλέξει να προβάλλει ένα hotspot από το χάρτη. Πατώντας επάνω στο marker του hotspot, θα εμφανιστεί ένα πλαίσιο (callout), το οποίο περιέχει τις κυριότερες πληροφορίες που σχετίζονται με το συγκεκριμένο hotspot. Στο πλαίσιο φαίνεται συνοπτικά

ένα τμήμα της περιγραφής του hotspot, το όνομα και η εικόνα προφίλ του χρήστη που έχει αναρτήσει το hotspot, ο χρόνος που έχει περάσει από την ώρα ανάρτησης, και ο αριθμός προβολών και σχολίων του hotspot.

Στην οθόνη χάρτη υπάρχουν τέσσερις επιπλέον διεπαφές. Η μπάρα αναζήτησης, το κρυφό συρταρωτό μενού, το πλήκτρο επαναφοράς τοποθεσίας χρήστη και το κεντρικό πλήκτρο πλοήγησης. Το πλήκτρο επαναφοράς τοποθεσίας χρήστη βρίσκεται στο κάτω αριστερά μέρος της οθόνης. Όπως φανερώνει και το όνομά του, βοηθάει το χρήστη να επαναφέρει τη θέση του χάρτη στην τοποθεσία την οποία βρίσκεται εκείνη τη στιγμή. Το κεντρικό πλήκτρο πλοήγησης βρίσκεται στο κάτω δεξιά μέρος της οθόνης και πατώντας το ο χρήστης μπορεί να επιλέξει σε ποια οθόνη θέλει να μεταφερθεί. Για τις άλλες δύο διεπαφές θα γίνει μια πιο εκτεταμένη αναφορά στη συνέχεια.

Μπάρα Αναζήτησης και Foursquare Suggest Completion

Εάν ο χρήστης επιθυμεί να πραγματοποιήσει μια συγκεκριμένη αναζήτηση ενός σημείου ενδιαφέροντος, μπορεί να το κάνει μέσω της μπάρας αναζήτησης στο πάνω μέρος της οθόνης. Ο χρήστης πληκτρολογεί λέξεις-κλειδιά που σχετίζονται με την τοποθεσία προς αναζήτηση, όπως το όνομα, την περιοχή κλπ. Η διεπαφή είναι εφοδιασμένη με την ιδιότητα αυτοσυμπλήρωσης του Foursquare API. Συνεπώς, τα αποτελέσματα εμφανίζονται καθώς ο χρήστης πληκτρολογεί, σε μια λίστα ακριβώς κάτω από την μπάρα.

Το Foursquare API, όπως αναφέρθηκε στην ενότητα 2.2.2, είναι μια διεπαφή ανοικτού κώδικα η οποία χρησιμεύει στην εύρεση πληροφοριών για διάφορα μέρη. Στην εφαρμογή, ένα από τα σημεία στα οποία ενσωματώνεται η διεπαφή αυτή είναι η μπάρα αναζήτησης στο επάνω μέρος του χάρτη, για την αυτοσυμπλήρωση των εμφανιζόμενων αποτελεσμάτων. Όταν πραγματοποιείται αναζήτηση, η είσοδος του χρήστη προστίθεται στο προς αναζήτηση URL και η εφαρμογή δημιουργεί ένα αίτημα GET προς την διεπαφή Foursquare API Suggest Completion

```
GET https://api.foursquare.com/v2/venues/suggestcompletion
```

επισυνάπτοντας τις λέξεις-κλειδιά και κάποια ακόμη δεδομένα ως endpoint στο παραπάνω URL, για την λήψη των απαιτούμενων πληροφοριών του σημείου ενδιαφέροντος. Το αποτέλεσμα είναι η εμφάνιση προτεινόμενων σημείων ανδιαφέροντος καθώς ο χρήστης πληκτρολογεί την επιθυμητή είσοδο.

Το αίτημα που τελικά αποστέλλεται στο Foursquare API αποτελείται από ένα νέο URL, το οποίο περιλαμβάνει τα απαραίτητα στοιχεία για την εύρεση των σχετικών αποτελεσμάτων. Το νέο URL που δημιουργείται, εμπεριέχει το παραπάνω URL, μαζί με ένα query που περιλαμβάνει τις εξής παραμέτρους αναζήτησης:

- ***ll*** – η παράμετρος αυτή προσδιορίζει την τοποθεσία στην οποία γίνεται η αναζήτηση
- ***query*** – οι λέξεις-κλειδιά που πληκτρολογεί ο χρήστης

- ***limit*** – η παράμετρος αυτή προσδιορίζει την πλήθος των εμφανιζόμενων αποτελεσμάτων
- ***radius*** – η παράμετρος αυτή προσδιορίζει την ακτίνα αναζήτησης σε m
- ***v*** – η παράμετρος αυτή προσδιορίζει την version της διεπαφής

Στον πίνακα που ακολουθεί, γίνεται ανάλυση ενός παραδείγματος για την καλύτερη κατανόηση του τρόπου λειτουργίας της διεπαφής αυτοσυμπλήρωσης.

Search Example: Acropolis Museum		
Name	***Example***	***Description***
ll	37.8,23.2	**required** latitude and longitude of the user's location.
query	acropolis museum	**required** search term to be applied against titles. Must be at least 3 characters long.
limit	20	number of results to return, up to 50.
radius	800	limit results to venues within this many meters of the specified location. Defaults to a city-wide area. The maximum supported radius is currently 80,000 meters.
v	20190501	the current API version.
client_id		a **unique** string that is given to the client for access privileges.
client_secret		a **unique** string that is given to the client for access privileges.

Πίνακας 5.1: Παράδειγμα αυτοσυμπλήρωσης για τις λέξεις-κλειδιά acropolis museum και δομή του URL αναζήτησης.

Συρταρωτό Μενού και Foursquare Explore

Στις δυνατότητες του χρήστη συμπεριλαμβάνεται και η φιλτραρισμένη αναζήτηση σημείων ενδιαφέροντος. Εάν ο χρήστης επιθυμεί να προβάλλει στο χάρτη σημεία που ανήκουν σε κάποια συγκεκριμένη κατηγορία, μπορεί να το κάνει, ανοίγοντας το κρυφό συρταρωτό μενού πατώντας το εικονίδιο της υδρόγειου σφαίρας στο επάνω αριστερά μέρος του χάρτη, ή σύροντας την οθόνη προς τα δεξιά. Στη συνέχεια, μπορεί να επιλέξει μία από τις κατηγορίες της λίστας και τα αποτελέσματα θα εμφανιστούν στο χάρτη.

Η παραπάνω υπηρεσία υλοποιείται με τη χρήση της διεπαφής Foursquare Explore, με βάση την οποία η εφαρμογή δημιουργεί αιτήματα αναζήτησης σημείων ενδιαφέροντος που ανήκουν σε κάποια από τις διαθέσιμες κατηγορίες. Τα αιτήματα που δημιουργούνται έχουν την ακόλουθη μορφή

```
GET https://api.foursquare.com/v2/venues/search
```

και το URL εμπεριέχει κάποιες επιπρόσθετες παραμέτρους που συνιστούν το query προς αποστολή στην διεπαφή. Στη συνέχεια θα αναλυθεί ένα παράδειγμα των αποτελεσμάτων που επιστρέφονται στην εφαρμογή από τη διεπαφή, όταν ο χρήστης επιλέγει την κατηγορία με ετικέτα '*arts*'.

Το νέο URL που θα δημιουργηθεί θα ξεκινάει με το παραπάνω URL και θα περιέχει κάποιες ακόμη πληροφορίες που θα συνιστούν το τελικό endpoint. Οι πληροφορίες αυτές είναι παρόμοιες με αυτές που χρησιμοποιήθηκαν για το σχηματισμό του endpoint που θα αποσταλλεί στη διεπαφή Foursquare Suggest Completion, με μία μόνο επιπλέον παράμετρο: την κατηγορία (section) στην οποία ανήκει το σημείο ενδιαφέροντος, για τον περιορισμό της αναζήτησης. Στο παράδειγμα όπου ο χρήστης επιλέγει σαν φίλτρο της αναζήτησης την κατηγορία '*arts*', τότε η παράμετρος section αντιστοιχίζεται με την κατηγορία arts.

Τα αποτελέσματα που επιστρέφει η διεπαφή Foursquare Explore θα περιέχουν μια σειρά από πληροφορίες, τις οποίες η εφαρμογή θα διαχειρίζεται κατάλληλα για την εμφάνισή τους στο χάρτη. Τα στοιχεία που επιστρέφονται από τη διεπαφή φαίνονται στον κάτωθι πίνακα:

https://api.foursquare.com/v2/venues?ll=37.8,23.7&section=arts&radius=800	
Field	***Description***
id	a **unique** string identifier for this venue.
name	the best known name for this venue.
location	an object containing none, some, or all of address (street address), crossStreet, city, state, postalCode, country, lat, lng, and distance. All fields are strings, except for lat, lng, and distance. Distance is measured in meters. Some venues have their locations intentionally hidden for privacy reasons (such as private residences). If this is the case, the parameter isFuzzed will be set to true, and the lat/lng parameters will have reduced precision.
categories	an array, possibly empty, of categories that have been applied to this venue. One of the categories will have a primary field indicating that it is the primary category for the venue. For the complete category tree, see categories.

Πίνακας 5.2: πληροφορίες που επιστρέφονται από τη διεπαφή Foursquare Explore.

Εκτός από την αναζήτηση σημείων ενδιαφέροντος, ο χρήστης μπορεί να προβάλλει κάποια από τα hotspots που εμφανίζονται στο χάρτη. Πατώντας επάνω σε κάποιο marker, ο χρήστης

μπορεί να δει τις πληροφορίες που αφορούν το συγκεκριμένο hotspot, όπως να διαβάσει την περιγραφή, να δει το χρήστη που δημιούργησε το hotspot, να δει τον αριθμό σχολίων ή προβολών κλπ.

5.1.2.3 CreateHotspotScreen

Για τη δημιουργία ενός προσωπικού hotspot, ο χρήστης μεταφέρεται στην οθόνη CreateHotspotScreen. Εκεί, μπορεί να δώσει στο νέο του hotspot μια περιγραφή, να θέσει χρονικό όριο ισχύος και να επισυνάψει μια φωτογραφία από τη συλλογή της συσκευής του ή να βγάλει καινούρια με την κάμερα. Μόλις ολοκληρώσει τη συμπλήρωση της φόρμας δημιουργίας με τα παραπάνω στοιχεία, ο χρήστης πατάει το πλήκτρο με επιγραφή '*Done*' που βρίσκεται στο δεξιά μέρος της επικεφαλίδας της οθόνης.

Σε περίπτωση σφάλματος, η διεπαφή μηνυμάτων λάθους αναλαμβάνει την εμφάνιση κατάλληλου μηνύματος προς το χρήστη. Το μήνυμα θα έχει τη μορφή ειδοποιητικού πλαισίου διαλόγου, όπου και θα περιγράφεται το αίτιο που προξενεί το σφάλμα στη φόρμα δημιουργίας hotspot. Στη συνέχεια, ο χρήστης αποδέχεται την ειδοποίηση πατώντας '*OK*' και προχωράει στη διόρθωση των σφαλμάτων. Το πεδίο περιγραφής του hotspot δεν πρέπει να είναι κενό, ενώ η τιμή χρονικού ορίου αρχικοποιείται στα 30 λεπτά. Η προσθήκη φωτογραφίας είναι προαιρετική. Ο χρήστης μπορεί να ακυρώσει τη δημιουργία του νέου hotspot ανά πάσα στιγμή πατώντας το πλήκτρο με επιγραφή '*Cancel*' που βρίσκεται στα αριστερά της επικεφαλίδας της οθόνης.

Μόλις η φόρμα δημιουργίας νέου hotspot διορθωθεί και δεν περιέχει άλλα λάθη, η υποβολή ολοκληρώνεται επιτυχώς. Ο χρήστης ενημερώνεται για τις ενέργειες που θα ακολουθήσουν μετά την υποβολή της φόρμας και καλείται να αποδεχθεί ή όχι την δημιουργία του νέου hotspot. Εάν ο χρήστης αποδεχθεί τη δημιουργία του νέου hotspot, τότε ανακατευθύνεται στο χάρτη και το καινούριο hotspot αποθηκεύεται στη βάση και εμφανίζεται στο χάρτη στην τοποθεσία του χρήστη. Διαφορετικά, το hotspot δεν αποθηκεύεται και ο χρήστης μεταφέρεται ξανά στο χάρτη.

5.1.2.4 HotspotListScreen

Η εξατομίκευση της εφαρμογής ανάλογα με το χρήστη επιτυγχάνεται με την λίστα προσωπικών hotspots στην HotspotListScreen. Εκεί, μπορεί να μεταφερθεί ο χρήστης όταν επιθυμεί να προβάλλει όλα τα προσωπικά hotspots που έχει δημιουργήσει στο παρελθόν. Η λίστα είναι ταξινομημένη με τα πιο πρόσφατα hotspots να εμφανίζονται πρώτα και τα παλαιότερα να βρίσκονται στο τέλος.

Η οθόνη προσωπικών hotspots εξυπηρετεί επίσης στην διευκόλυνση της διαδικασίας διαχείρισης των hotspots του χρήστη. Εάν ο χρήστης επιθυμεί να επεξεργαστεί την περιγραφή κάποιου hotspot για παράδειγμα, μπορεί να το κάνει από εδώ. Η διαδικασία είναι απλή και σύντομη, αφού ο χρήστης απλώς σύρει το συγκεκριμένο hotspot προς τα δεξιά και πατάει το πλήκτρο με την επιγραφή '*edit*'. Στην σελίδα επεξεργασίας, ο χρήστης μπορεί να μεταβάλλει όλες τις πληροφορίες που σχετίζονται με το hotspot. Μπορεί επίσης να αλλάξει την εικόνα

του hotspot, ή να προσθέσει μια εικόνα σε περίπτωση που δεν υπήρχε προηγουμένως. Επιπλέον, ο χρήστης μπορεί να διαγράψει όποιοδήποτε hotspot. Η διαδικασία είναι παρόμοια με αυτή της την επεξεργασίας: ο χρήστης σύρει το hotspot προς διαγραφή προς τα αριστερά και πατάει το πλήκτρο με την επιγραφή '*delete*'.

5.1.2.5 CommentScreen

Ο χρήστης μπορεί να αλληλεπιδράσει με ένα οποιοδήποτε hotspot. Αυτό μπορεί να γίνει είτε από το χάρτη, είτε από τη λίστα προσωπικών hotspots. Και στις δυο περιπτώσεις, ο χρήστης πατάει στο εικονίδιο με επιγραφή '*comments*' ενός hotspot και μεταφέρεται αυτόματα στην οθόνη σχολίων.

Από τη διεπαφή χάρτη, ο χρήστης μπορεί να αλληλεπιδράσει είτε με δικά του hotspots, είτε με hotspots άλλων χρηστών. Στην CommentScreen, ο χρήστης μπορεί να διαβάσει την περιγραφή του hotspot, να διαβάσει τα σχόλια των άλλων χρηστών, να προβάλλει την φωτογραφία του hotspot εάν αυτή υποάρχει και να απαντήσει στο hotspot γράφοντας το δικό του σχόλιο. Για να το κάνει αυτό, ο χρήστης περιηγείται στο κάτω μέρος της σελίδας, σύροντας την οθόνη προς τα πάνω. Εκεί μπορεί να εισάγει το σχόλιό του στο ειδικά διαμορφωμένο για αυτό το σκοπό τμήμα δημιουργίας σχολίων. Αφού ολοκληρώσει το σχόλιό του, ο χρήστης πατάει το πλήκτρο αποστολής που βρίσκεται στα δεξιά του reply box και το νέο σχόλιο εμφανίζεται τελευταίο.

Η ίδια διαδικασία μπορεί να επιτευχθεί και από την HotspotListScreen. Αυτή η μέθοδος συμφέρει περισσότερο εάν πρόκειται για κάποιο προσωπικό hotspot του χρήστη.

5.1.2.6 ProfileScreen

Ένα από τα σημαντικότερα χαρακτηριστικά της εφαρμογής είναι η οθόνη προσωπικού λογαριασμού του χρήστη. Η ProfileScreen προσθέτει έναν ακόμη βαθμό εξατομίκευσης της εφαρμογής. Ο χρήστης μπορεί να προβάλλει πληροφορίες σχετικά με τη δράση του εντός της εφαρμογής, να δει ολόκληρη τη συλλογή φωτογραφιών που έχει δημιουργήσει και να επεξεργαστεί τις προσωπικές του πληροφορίες και ρυθμίσεις.

Για την επεξεργασία των προσωπικών του στοιχείων και ρυθμίσεων, ο χρήστης πρέπει πρώτα να μεταφερθεί στην οθόνη επεξεργασίας του προσωπικού προφίλ. Αυτό επιτυγχάνεται πατώντας το πλήκτρο με την επιγραφή '*edit*' που βρίσκεται στα δεξιά του τίτλου της επικεφαλίδας της οθόνης. Έπειτα, ο χρήστης μεταφέρεται αυτόματα σε νέα σελίδα όπου υπάρχει μια φόρμα με τα παλιά στοιχεία του χρήστη. Με κατάλληλες περιγραφές των πεδίων, ο χρήστης καθοδηγείται στον τρόπο συμπλήρωσης της φόρμας επεξεργασίας προσωπικού προφίλ. Τα στοιχεία που μπορεί να επεξεργαστεί ο χρήστης είναι το πλήρες όνομα, το όνομα χρήστη, η ηλεκτρονική διεύθυνση, ο κωδικός πρόσβασης, η πόλη, η ημερομηνία γέννησης και το γένος. Επίσης, ο χρήστης μπορεί να αλλάξει την φωτογραφία προφίλ του. Εάν δεν υπάρχει φωτογραφία προφίλ, ο χρήστης μπορεί να προσθέσει μία από την συλλογή της συσκευής ή να βγάλει μια καινούρια με την κάμερα. Τέλος, ο χρήστης μπορεί να αλλάξει τις ρυθμίσεις του λογαριασμού του, από δημόσιο σε ιδιωτικό.

Σε περίπτωση σφάλματος, η διεπαφή μηνυμάτος λάθους αναλαμβάνει την καθοδήγηση του χρήστη για την διόρθωση των σφαλμάτων της φόρμας επεξεργασίας προφίλ. κατάλληλα μηνύματα λάθους εμφανίζονται κάτω από τα αντίστοιχα πεδία στα οποία έχει γίνει το λάθος. Τα μηνύματα εξαφανίζονται με την επίλυση του προβλήματος. Για την επιτυχή ολοκλήρωση της διαδικασίας επεξεργασίας του προσωπικού του προφίλ, ο χρήστης πρέπει να εισάγει τον κωδικό πρόσβασης. Στη συνέχεια, μεταφέρεται στο προφίλ του, όπου και τα στοιχεία του έχουν ανανεωθεί.

Η οθόνη προσωπικού λογαριασμού του χρήστη χρησιμεύει επίσης για την προβολή των στατιστικών δεδομένων. Ο χρήστης πατάει το βέλος που βρίσκεται στα δεξιά της επιγραφής '*Stats for nerds*', κάτω από το τμήμα '*SETTINGS*'. Στη συνέχεια, ο χρήστης μεταφέρεται αυτομάτως σε νέα σελίδα, όπου υπάρχουν τρία τμήματα με στατιστικά δεδομένα. Στο πρώτο τμήμα με τίτλο '*CCOUNT*' είναι συγκεντρωμένες οι πληροφορίες που αφορούν τη δράση του χρήστη εντός της εφαρμογής. Τα δεδομένα αυτά αφορούν τον αριθμό hotspots, τον αριθμό προβολών και τον αριθμό σχολίων του χρήστη. Στο δεύτερο τμήμα με τίτλο '*INSIGHTS*' βρίσκονται τα ποσοστά που αφορούν την πορεία του χρήστη στην εφαρμογή. Τα ποσοστά αυτά σχετίζονται με το βαθμό δημοτικότητας που έχει αποκτήσει ο χρήστης σε σχέση με το υπόλοιπο σύνολο. Τέλος, στο τρίτο τμήμα με τίτλο '*AUDIENCE*', βρίσκονται τα ποσοστά των χρηστών με βάση το φύλο.

Ο χρήστης μεταβαίνει στην ProfileScreen και για την έξοδο από την εφαρμογή. Για να γίνει αυτό, ο χρήστης πρέπει να πατήσει το πλήκτρο που βρίσκεται κάτω από το τμήμα '*SETTINGS*', με την επιγραφή '*Log out*' με κόκκινους χαρακτήρες.

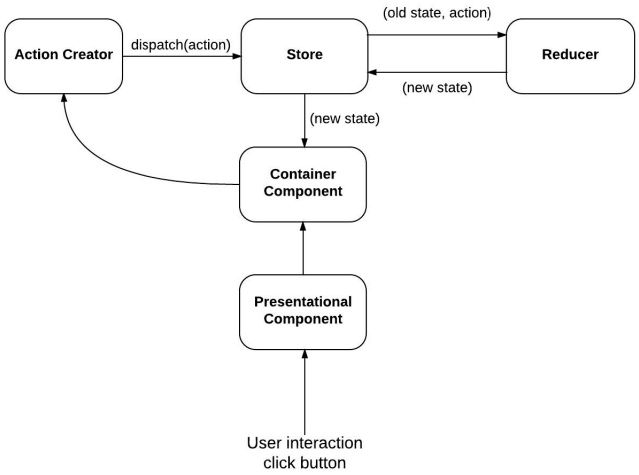
5.1.3 Τενικές Ενιαίας Διαχείρισης Δεδομένων

Στην εφαρμογή χρησιμοποιήθηκε μια τεχνολογία διαχείρισης δεδομένων για το συγχρονισμό μεταξύ των διαφόρων οθονών. Αυτό επιτεύχθηκε με την ενσωμάτωση της βιβλιοθήκης React Redux και παρεμφερών εργαλείων που διευκολύνουν την διαδικασία ροής δεδομένων εντός της εφαρμογής, προσφέροντας επιπρόσθετες δυνατότητες στον προγραμματιστή. Χαρακτηριστικότερη εξ' αυτών είναι η αποθήκευση δεδομένων σε έναν ενιαίο χώρο, στον οποίο έχουν πρόσβαση όλα τα τμήματα της εφαρμογής τα οποία είναι συνδεδεμένα κατά κάποιο τρόπο με το χώρο αυτό. Στις ενότητες που ακολουθούν θα γίνει μια πιο εκτεταμένη ανάλυση των εργαλείων διαχείρισης που χρησιμοποιήθηκαν στην εφαρμογή.

5.1.3.1 React Redux

Η React Redux είναι μια βοηθητική βιβλιοθήκη ανοικτού κώδικα η οποία χρησιμεύει στην διαρρύθμιση των δεδομένων και την ιεράρχιση των καταστάσεων μιας εφαρμογής. Η ιδεολογία της βιβλιοθήκης αποτελεί έμπνευση της ομάδας προγραμματιστών του Facebook και μιμείται τα χαρακτηριστικά μιας παρόμοιας βιβλιοθήκης, της Flux.

Το κυριότερο πλεονέκτημα της React Redux είναι ο τρόπος με τον οποίο διαχειρίζεται τα δεδομένα μιας εφαρμογής. Διαθέτει μια αμφίδρομη ροή δεδομένων μεταξύ ενός ενιαίου αποθηκευτικού χώρου και της εφαρμογής. Αυτό επιτρέπει στον προγραμματιστή τον εύκολο



Σχήμα 5.2: Δομή React Redux εντός της εφαρμογής.

και γρήγορο τρόπο ανάπτυξης και αποσφαλμάτωσης της εφαρμογής.

Η δομή της εφαρμογής με τη χρήση της React Redux φαίνεται στο σχ. 5.5. Στη συνέχεια, ακολουθεί μια ανάλυση κάθε παράγοντα της βιβλιοθήκης, όπως αυτός χρησιμοποιείται εντός της εφαρμογής.

Presentational & Container Components

Κατά την υλοποίηση της εφαρμογής, κάθε οθόνη διακριτοποιείται σε στοιχεία αναπαράστασης δεδομένων (presentational/functional or stateless components) και στοιχεία κατάστασης (container or ctateful components), όπως έχει ήδη αναφερθεί στην ενότητα 4.1.4. Τα presentational components δεν διαθέτουν λογική, το οποίο σημαίνει πως δε διαχειρίζονται καθόλου δεδομένα. Το μοναδικό συστατικό απαραίτητο στην λειτουργία τους είναι τα δεδομένα που λαμβάνουν μέσω των props από το στοιχείο-πατέρα. Στην περίπτωση που απαιτείται η εκτέλεση μιας ενέργειας σχετική με τη διαζείριση δεδομένων από ένα presentational component, αυτό επιτυγχάνεται με τον ίδιο τρόπο που λαμβάνονται τα δεδομένα. Με το '*πέρασμα*' της αντίστοιχης συνάρτησης ανάκλησης (callback) μέσω των props, αυτή καθίσταται διαθέσιμη ανά πάσα στιγμή στο presentational component, χωρίς την ανάγκη για πρόσθετη λογική. Κάθε presentational component έχει την παραπάνω ιδιότητα να δέχεται δεδομένα και συναρτήσεις ώς είσοδο κατά την αρχικοποίηση.

Τα container components είναι συνήθως στοιχεία-γονείς σε presentational components. Αυτά αναλαμβάνουν την διαχείριση των δεδομένων και των ενεργειών που έχουν αποσταλλεί από τα presentational components, όταν πραγματοποιείται κάποια δράση από την πλευρά του χρήστη. Επίσης, είναι υπεύθυνα για την πλοήγηση μεταξύ οθονών και πολλών άλλων σημείων λογικής εντός της εφαρμογής.

Action Creators

Για την αποστολή αιτημάτων που πυροδοτούν την εκτέλεση συγκεκριμένων ενεργειών όταν ο χρήστης αλληλεπιδρά με την εφαρμογή, η redux χρησιμοποιεί ειδικές συναρτήσεις δημιουργίας αιτημάτων δράσεων, τους λεγόμενους action creators. Οι action creators χρησιμοποιούν την συνάρτηση αποστολής αιτημάτων dispatch της redux, που έχει εμβέλεια σε όλα τα διασυνδεδεμένα τμήματα της εφαρμογής.

Redux Store

Η React Redux χρησιμοποιεί την Store ως αποθηκευτικό χώρο για να διατηρεί την κατάσταση της εφαρμογής. Η κατάσταση της εφαρμογής μεταβάλλεται όταν πυροδοτείται μια ενέργεια με κάποιον από τους action creators που έχουν δημιουργηθεί για διαφορετικές ενέργειες, καθ΄όλη τη διάρκεια της εφαρμογής. Οι αλλαγές που πραγματοποιούνται αποθηκεύονται στην store. Η κατάσταση της εφαρμογής μπορεί να οριστεί ως ένα σύνολο αντικειμένων, συλλογές από αντικείμενα, αποθηκευμένα δεδομένα από το server και πιθανόν να υπάρχουν booleans, integer ή string τιμές που ζητούνται από την εφαρμογή. Για παράδειγμα, η κατάσταση της store της εφαρμογής μπορεί να κρατήσει την τιμή του διακριτικού ελέγχου ταυτότητας (authorization token) ενός χρήστη, εάν ο χρήστης κατάφερε να συνδεθεί με επιτυχία στο σύστημα. Έτσι, ο χρήστης δεν χρειάζεται να επαναλαμβάνει την διαδικασία σύνδεσης στην εφαρμογή κάθε φορά που ανανεώνει την οθόνη, ή εξέρχεται προσορινά από την εφαρμογή.

Reducers

Οι Reducers είναι αγνές συναρτήσεις (pure functions). Έτσι ονομάζονται οι συναρτήσεις οι οποίες δεν τροποποιούν τα στοιχεία που δέχονται ως είσοδο (arguments). Λαμβάνοντας ως είσοδο την παρούσα κατάσταση των δεδομένων της εφαρμογής, πραγματοποιούν τις απαραίτητες αλλαγές στην κατάσταση της εφαρμογής, μέσω της εκτέλεσης μιας ενέργειας που απεστάλη από κάποιον action creator. Επιστρέφουν την νέα κατάσταση της εφαρμογής σε ένα νέο στοιχείο με τη μορφή αντικειμένου. Η νέα κατάσταση μπορεί να είναι απλά ένα αντίγραφο της προηγούμενης κατάστασης, αλλά δεν μπορεί σε καμία περίπτωση να είναι η παρούσα κατάσταση τροποποιημένη. Η παρούσα κατάσταση παραμένει πάντοτε απαράλλαχτη. Οι reducers είναι στην ουσία μηχανές κατάστασης, οι οποίες ορίζουν τη διαδοχή μεταξύ των καταστάσεων της εφαρμογής.

5.1.3.2 Redux Thunk Middleware

Στην περίπτωση που μια σειρά ενεργειών πρέπει να εκτελεστούν σε ένα συγκεκριμένο χρονικό σημείο που προηγείται του σταδίου των reducers, τότε γίνεται χρήση λογισμικού ειδικού σκοπού, γνωστού ως Middleware. Με αυτό τον τρόπο, δίνεται η δυνατότητα στον προγραμματιστή να υλοποιήσει ενδιάμεση λογική η οποία θα εκτελείται πρωτού γίνει η ενέργεια που είναι προορισμένη να πραγματοποιηθεί από κάποιον reducer σύμφωνα με τον αντίστοιχο

action creator.

Στην εφαρμογή γίνεται χρήση του Redux Thunk [60], με τη βοήθεια του οποίου ένας action creator μπορεί να επιστρέψει μια συνάρτηση ανάκλησης callback function στη θέση ενός αντικειμένου. Κατά τη διάρκεια εκτέλεσης του middleware, διακόπτεται προσωρινά η ροή εκτέλεσης μιας ενέργειας για να γίνει η απαιτούμενες επεξεργασίες στην κατάσταση (βλ. Σχ. 5.3). Υπεύθυνη για το τμήμα της λογικής που εκτελείται είναι η callback function. Η callback function εκτελείται ασύγχρονα και μόλις ολοκληρωθεί, συνεχίζεται κανονικά η ροή εκτέλεσης του προγράμματος. Έπειτα, αποστέλλεται η ενέργεια στον προοριζόμενο reducer και η διαδικασία συνεχίζεται όπως πριν.

Σχήμα 5.3: Υλοποίηση λογικής του Redux Thunk Middleware.

Στην εφαρμογή, η προαναφερθείσα ροή υλοποιείται σε ενέργειες που απαιτούν τη λήψη δεδομένων από τη βάση πρωτού πυροδοτήσουν μια ενέργεια. Για παράδειγμα, όταν ο χρήστης πραγματοποιεί σύνδεση στην εφαρμογή, πυροδοτείται μια ενέργεια σύνδεσης χρήστη. Για την επιτυχή ολοκλήρωση της ενέργειας σύνδεσης θα πρέπει να γίνει λήψη των διαπιστευτηρίων του χρήστη για να διασταυρωθούν με αυτά που έδωσε ως είσοδο κατά τη διαδικασία σύνδεσης στην εφαρμογή. Έτσι, η ενέργεια σύνδεσης διακόπτεται μέχρις ότου ολοκληρωθεί το αίτημα λήψης των διαπιστευτηρίων του χρήστη από τη βάση δεδομένων. Έπειτα, η ενέργεια αποστέλλεται στον αντίστοιχο reducer και η διαδικασία συνεχίζεται κανονικά. Στο παραπάνω παράδειγμα, η λογική που αφορά στη λήψη των διαπιστευτηρίων του χρήστη από τη βάση, αποτελεί το middleware που εκτελείται στο ενδιάμεσο της ενέργειας σύνδεσης χρήστη στην εφαρμογή.

5.1.4 Βιβλιοθήκες και Βοηθητικά Προγραμματιστικά Πακέτα

Για την εγκατάσταση πρόσθετων βιβλιοθηκών έγινε χρήση των εργαλείων npm και yarn. Και τα δύο είναι βοηθητικά πακέτα διαχείρισης βιβλιοθηκών. Σκοπός τους είναι η εγκατάσταση βιβλιοθηκών και η σύνδεσή τους με τις απαραίτητες βιβλιοθήκες εξάρτησης (dependencies)

καθ΄ όλη την έκταση της εφαρμογής.

Στη συνέχεια θα γίνει μια αναφορά στις πιο αξιοσημείωτες βιβλιοθήκες που χρησιμοποιήθηκαν κατά την υλοποίηση της εφαρμογής.

axios

Η βιβλιοθήκη `axios` χρησιμεύει για την δημιουργία και αποστολή ασύγχρονων αιτημάτων προς τον `server`. Τα αιτήματα μπορούν να είναι της μορφής `GET, POST, PUT, DELETE` και αναλόγως το είδος του αιτήματος μπορούν να συνοδεύονται ή όχι από δεδομένα [61].

native-base

Η βιβλιοθήκη `native-base` χρησιμοποιήθηκε ως επί το πλείστον για τη σχεδίαση των διεπιφανειών χρήστη της εφαρμογής. Διαδραστικά στοιχεία όπως φόρμες, πλήκτρα κλπ. έχουν σχεδιαστεί με τη βοήθεια της βιβλιοθήκης αυτής. Το χαρακτηριστικό της `native-base` που την καθιστά ιδιαίτερα εξυπηρετική είναι το γεγονός ότι προσφέρει στον προγραμματιστή έτοιμα στοιχεία, τα οποία είναι ευέλικτα και πλήρως λειτουργικά. Έτσι μειώνεται σημαντικά ο απαιτούμενος χρόνος σχεδίασης και ο προγραμματιστής μπορεί να επικεντρωθεί στην υλοποίηση της λογικής της εφαρμογής [62].

react-native-router-flux

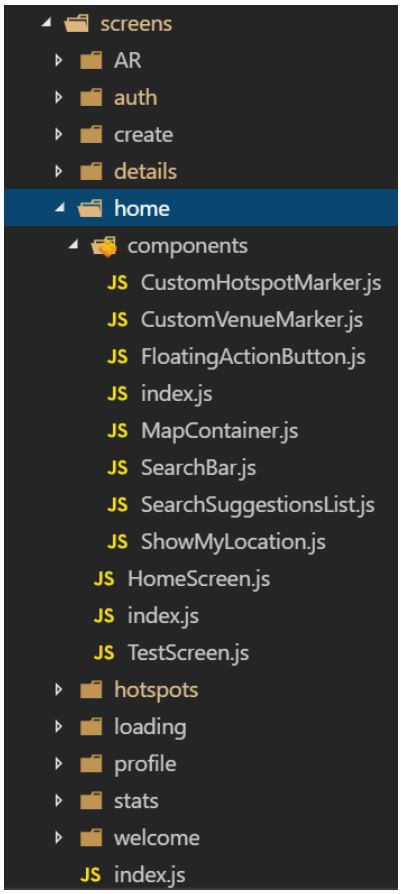
Η υλοποίηση του μενού πλοήγησης εντός της εφαρμογής έγινε με τη βοήθεια της βιβλιοθήκης `react-native-router-flux`. Σύμφωνα με αυτή, κάθε οθόνη αποτελεί ένα `scene` και πολλά `scenes` μαζί συνιστούν ένα `stack`. Κάθε `scene` φέρει ιδιότητες όπως επικεφαλίδα, τίτλο, πλήκτρα πλοήγησης κλπ. Όσα `scenes` βρίσκονται ομαδοποιημένα υπό ένα `stack` μπορούν να διαδέχονται η μία την άλλη χάρη στην ιδιότητα μνήμης που παρέχεται στην στοίβα. Όλες οι οθόνες βρίσκονται κάτω από έναν κεντρικό δρομολογητή (`router`) [63].

react-native-maps

Η βασική βιβλιογραφία του `facebook` σχετικά με τη `react native` προσφέρει αυτή τη βιβλιοθήκη για την υλοποίηση διεπαφών χάρτη ως μιας καλύτερης εναλλακτικής έναντι των έτοιμων διεπαφών που υπάρχουν υλοποιημένες. Η `react-native-maps` παρέχει στον προγραμματιστή μια σειρά από δυνατότητες που αφορούν στην διαμόρφωση διεπαφών χάρτη με μεγαλύτερη ευελιξία ως προς τον έλεγχο γεγονότων επάνω στο χάρτη, την μορφοποίηση της διεπιφάνειας χάρτη, την καταγραφή συντεταγμένων διαφόρων σημείων, την προγραμματισμένη μεταβολή θέσης, την χρήση διεπαφών με διαδραστικά στοιχεία και πολλά άλλα [;].

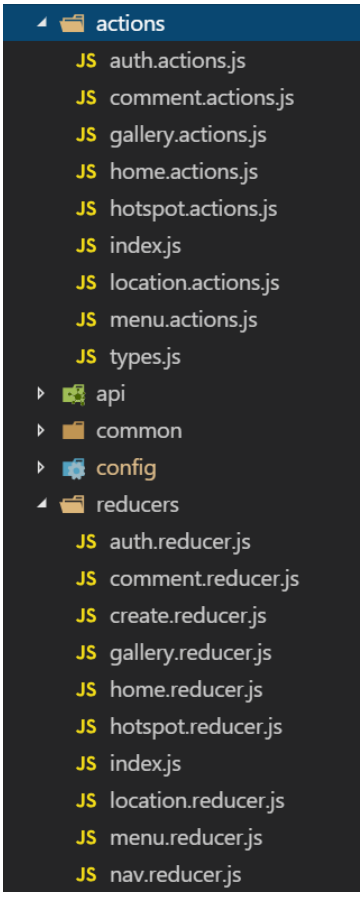
5.1.5 Ιεράρχηση Αρχείων και Δομή Φακέλων

Η ιεραρχία που ακολουθούν τα αρχεία δείχνει την πορεία για την κατάστρωση του σχεδίου της εφαρμογής. Υπάρχει χαρακτηριστική αντιστοιχία μεταξύ της δομής των φακέλων και της δομής της εφαρμογής. Κάθε οθόνη υλοποιείται σε ξεχωριστό φάκελο, ο οποίος περιλαμβάνει το αρχείο στο οποίο βρίσκεται ο σκελετός της συγκεκριμένης οθόνης, αλλά και τα υπόλοιπα στοιχεία που συνιστούν τα βοηθητικά `components` της οθόνης. Η ονομασία των αρχείων ακολουθεί μία σύμβαση κατά την οποία τα αρχεία που περιέχουν `components` ξεκινούν με κεφαλαία, ενώ οποιοδήποτε άλλο αρχείο ξεκινάει με μικρά. Αρχεία τα οποία αποτελούνται από περισσότερες από μια λέξεις ακολουθούν `camelCase` γραφή (βλ. Σχ. 5.4).



Σχήμα 5.4: Δομή φακέλων και αρχείων για τις οθόνες της εφαρμογής.

Η λογική που αφορά την `react redux` βρίσκεται σε ξεχωριστούς φακέλους. Οι ενέργειες που πυροδοτούνται από την αλληλεπίδραση του χρήστη με τις διεπιφάνειες της εφαρμογής είναι συγκεντρωμένες στο φάκελο `actions`. Κάθε `action` κατατάσσεται σε διαφορετικό αρχείο, ανάλογα με το είδος του. Για παράδειγμα, `actions` που αφορούν `hotspots` ομαδοποιούνται σε ένα αρχείο υπό το όνομα `hotspot.actions.js`. Η ίδια λογική ακολουθείται και για τα αρχεία που περιέχουν τους `reducers` (βλ. Σχ. 5.5).



Σχήμα 5.5: Δομή φακέλων και αρχείων που αφορούν react redux.

Τα αρχεία που σχετίζονται με την πλοήγηση εντός της εφαρμογής βρίσκονται σε έναν ξεχωριστό φάκελο με το όνομα `routes`. Στο αρχείο με όνομα `Navigator.js` βρίσκεται η υλοποίηση του κεντρικού δρομολογητή της εφαρμογής.

5.2 Υλοποίηση Εξυπηρετητή Αιτημάτων και Βάσης Δεδομένων

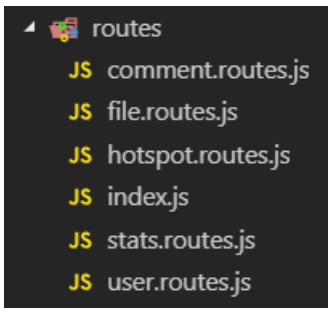
Σε αυτή την ενότητα θα γίνει ανάλυση των τεχνικών που εφαρμόστηκαν στην υλοποίηση του `server` της εφαρμογής. Θα γίνει αναφορά στους δρομολογητές και τα αντίστοιχα `routes`, καθώς και στους ελεγκτές που σχετίζονται με κάθε δρομολογητή. Θα παρουσιαστούν επίσης τα `endpoints` της εφαρμογής και θα επεξηγηθούν οι λειτουργίες που αντιστοιχούν σε καθένα από αυτά.

5.2.1 Server Routes

Ο `server` χρησιμοποιεί ένα σύστημα δρομολόγησης (`routes`) των αιτημάτων του `client` που προορίζονται προς εξυπηρέτηση. Κάθε ομάδα υπηρεσιών που απευθύνονται σε μια συγκεκριμένη λειτουργία της εφαρμογής αντιστοιχίζεται σε ένα δρομολογητή (`router`). Ο `router`

απαρτίζεται από επιμέρους routes, καθεμία εκ των οποίων εξυπηρετεί μια συγκεκριμένη λειτουργία. Έτσι, στο παραπάνω παράδειγμα, το αίτημα για τη δημιουργία νέου hotspot θα δρομολογηθεί για εξυπηρέτηση από τον router που είναι υπεύθυνος για τις ενέργειες που αφορούν ένα hotspot (*HotspotRoutes*).

Το σύστημα των δρομολογητών έχει σχεδιαστεί με στόχο την απλότητα και την αμεσότητα στην κατανόηση από τον προγραμματιστή (βλ. Σχ. 5.6). Αιτήματα που αφορούν τους χρήστες δρομολογούνται προς εξυπηρέτηση από τον router που είναι υπεύθυνος για τις ενέργειες που αφορούν τους χρήστες. Με τον ίδιο τρόπο, Αιτήματα που σχετίζονται με σχόλια χρηστών σε ένα hotspot, εξυπηρετούνται από εκείνο τον router που είναι υπεύθυνος για τα σχόλια.



Σχήμα 5.6: Ιεράρχηση του συστήματος δρομολογητών στην εφαρμογή.

Οι δρομολογητές του συστήματος είναι οι ακόλουθοι:

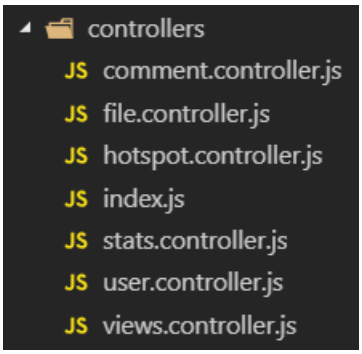
- **Hotspot Routes** – περιλαμβάνει routes που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με hotspots, όπως δημιουργία, επεξεργασία, διαγραφή, προβολή, ανάκτηση κλπ.
- **User Routes** – περιλαμβάνει routes που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με χρήστες, όπως εγγραφή, σύνδεση, επεξεργασία, προβολή, ανάκτηση κλπ.
- **Comment Routes** – περιλαμβάνει routes που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με σχόλια, όπως δημιουργία, προβολή, απαρίθμηση κλπ.
- **File Routes** – περιλαμβάνει routes που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με συνημμένα αρχεία σε hotspots.
- **Stats Routes** – περιλαμβάνει routes που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με στατιστικά δεδομένα.

5.2.2 Server Controllers

Αφότου τα αιτήματα δρομολογηθούν προς εξυπηρέτηση στους αντίστοιχους routers, ακολουθεί η επεξεργασία τους και η εκτέλεσή τους μέσω κατάλληλων μεθόδων. Οι μέθοδοι που αναλαμβάνουν την εκτέλεση των αιτημάτων ονομάζονται ελεγκτές (controllers) και αποτελούνται από ένα σύνολο ασύγχρονων εντολών και κλήσεων μεταξύ της βάσης δεδομένων και του server.

Όπως και με το σύστημα των δρομολογητών, έτσι και οι ελεγκτές έχουν σχεδιαστεί με στόχο να αποσυμπλέκουν τις ενέργειες που σχετίζονται με κάθε αίτημα (βλ. Σχ. 5.7). Όλες οι ενέργεεις που αφορούν τα `hotspots` εκτελούνται αποκλειστικά από έναν ελεγκτή. Το ίδιο συμβαίνει και με τις ενέργεεις γύρω από τους χρήστες. Το σύστημα των `controllers` έχει παραπλήσια μορφή με αυτό των `routers` καθώς στόχος της σχεδίασης είναι να υπάρχει ένας βαθμός αναλογίας μεταξύ των διαφόρων τμημάτων,ο οποίος μειώνει αισθητά το χρόνο κατανόησης και βελτιστοποιεί τη διαδικασία υλοποίησης:

- **Hotspot Controller –** περιλαμβάνει τις μεθόδους που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με `hotspots`, όπως δημιουργία, επεξεργασία, διαγραφή, προβολή, ανάκτηση κλπ.
- **User Controller –** περιλαμβάνει τις μεθόδους που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με χρήστες, όπως εγγραφή, σύνδεση, επεξεργασία, προβολή, ανάκτηση κλπ.
- **Comment Controller –** περιλαμβάνει τις μεθόδους που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με σχόλια, όπως δημιουργία, προβολή, απαρίθμηση κλπ.
- **File Controller –** περιλαμβάνει τις μεθόδους που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με συνημμένα αρχεία σε `hotspots`.
- **Stats Controller –** περιλαμβάνει τις μεθόδους που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με στατιστικά δεδομένα.
- **Views Controller –** περιλαμβάνει τις μεθόδους που αφορούν την εξυπηρέτηση αιτημάτων σχετικά με τα `views` ενός `hotspot`.



Σχήμα 5.7: Ιεράρχηση του συστήματος ελεγκτών στην εφαρμογή.

5.2.3 REST API - Server Endpoints

Κάθε ενέργεια από την πλευρά του `client` πυροδοτεί ένα αίτημα στην πλευρά του `server`. Στην ενότητα αυτή αναλύεται η τεχνική με την οποία ταξινομούνται και δρομολογούνται τα αιτήματα σε συγκεκριμένα `endpoints`.

- hotspots

  GET /hotspots/radius?lat=''lng=''

  -**lat** - συντεταγμένη γεωγραφικού πλάτους (latitude)
  -**lng** - συντεταγμένη γεωγραφικού μήκους (longitude)

  ανακτά όλα τα μηνύματα εντός της προκαθορισμένης ακτίνας των 5000 μέτρων από τη δεδομένη θέση που καθορίζεται από το γεωγραφικό πλάτος (lat) και το γεωγραφικό μήκος (lng).

  GET /hotspots/:hospotId

  ανακτά τις πληροφορίες που αφορούν το μηνύμα με το συγκεκριμένο αναγνωριστικό hotspotId.

  GET /users/:userId/hotspots

  ανακτά όλα τα μηνύματα του χρήστη με το συγκεκριμένο αναγνωριστικό userId.

  POST /hotspots/new

  δημιουργεί μια νέα είσοδο για την αποθήκευση ενός καινούριου hotspot στη βάση.

  PUT /hotspots/:hotspotId/edit

  επεξεργάζεται το hotspot με το προκαθορισμένο αναγνωριστικό hotspotId, με τα δοθέντα στοιχεία.

  DELETE /hotspots/:hotspotId/delete

  διαγράφει το hotspot με το προκαθορισμένο αναγνωριστικό hotspotId από τη βάση δεδομένων.

- users

  GET /users/:userId

  ανακτά τα στοιχεία του χρήστη που προσδιορίζεται από το αναγνωριστικό userId.

  POST /register

  πραγματοποιεί εγγραφή ενός νέου χρήστη στο σύστημα, τα αποθηκεύει τα στοιχεία στη βάση και εκδίδει σύμβολο ταυτοποίησης (access token).

POST /login

πραγματοποιεί σύνδεση ενός χρήστη στην εφαρμογή και εκδίδει σύμβολο ταυτοποίησης (access token).

POST /verify

πραγματοποιεί έλεγχο της ταυτότητας του χρήστη διασταυρώνοντας το σύμβολο ταυτοποίησης του χρήστη με αυτό που είναι αποθηκευμένο στη βάση.

PUT /users/:userId/edit

επεξεργάζεται τα στοιχεία του χρήστη που καθορίζεται από το αναγνωριστικό userId και αποθηκεύει τον ανανεωμένο χρήστη στη βάση.

- comments

GET /:userId/hotspots/:hotspotId/comments

ανακτά τα σχόλια ενός hotspot με τη βοήθεια των διακριτικών hotspotId και userId.

POST /:userId/hotspots/:hotspotId/comments/new

δημιουργεί μια είσοδο στη βάση για την αποθήκευση ενός καινούριου σχολίου. Χρειάζεται το αναγνωριστικό του hotspot στο οποίο έγινε το σχόλιο, καθώς επίσης και του διακριτικού του χρήστη που έκανε το σχόλιο.

- files

GET /users/:userId/gallery

ανακτά όλα τα αρχεία ενός χρήστη με το συγκεκριμένο αναγνωριστικό userId.

POST /users/:userId/gallery/upload

δημιουργεί μια νέα είσοδο στη βάση για την αποθήκευση ενός αρχείου που αφορά το χρήστη με το συγκεκριμένο αναγνωριστικό userId.

DELETE /gallery/:fileId/delete

διαγράφει το αρχείο με το προκαθορισμένο αναγνωριστικό fileId από τη βάση δεδομένων όταν διαγραφεί ένα hotspot.

- stats

GET /stats/users

ανακτά το συνολικό αριθμό των χρηστών που χρησιμοποιούν την εφαρμογή.

GET /stats/comments

ανακτά το συνολικό αριθμό των σχολίων που έχουν δημιουργηθεί εντός της εφαρμογής.

GET /stats/hotspots

ανακτά το συνολικό αριθμό των hotspots που έχουν δημιουργηθεί εντός της εφαρμογής.

GET /stats/views

ανακτά το συνολικό αριθμό των views που έχουν δημιουργηθεί εντός της εφαρμογής.

GET /stats/:userId/hotspots

ανακτά το συνολικό αριθμό των hotspots που έχουν δημιουργηθεί από τον χρήστη με το συγκεκριμένο αναγνωριστικό userId.

GET /stats/:userId/comments

ανακτά το συνολικό αριθμό των σχολίων που έχουν δημιουργηθεί από τον χρήστη με το συγκεκριμένο αναγνωριστικό userId.

GET /stats/:userId/ratio

ανακτά το ποσοστό αγοριών/κοριτσιών που έχουν προβάλλει τα hotspots του χρήστη με το συγκεκριμένο αναγνωριστικό userId.

5.2.3.1 Βιβλιοθήκες και Βοηθητικά Προγραμματιστικά Πακέτα

Για την εγκατάσταση πρόσθετων βιβλιοθηκών έγινε και εδώ χρήση των εργαλείων npm και yarn. Στη συνέχεια θα γίνει μια αναφορά στις πιο αξιοσημείωτες βιβλιοθήκες που χρησιμοποιήθηκαν κατά την υλοποίηση του εξυπηρετητή και της βάσης δεδομένων.

passport

Η βιβλιοθήκη passport αποτελεί ένα είδος middleware για την ταυτοποίηση χρηστών. Χρησιμοποιώντας τεχνικές ταυτοποίησης που ονομάζονται strategies, καθορίζει τον τρόπο με τον οποίο γίνεται η ταυτοποίηση χρηστών στα διάφορα σενάρια που περιλαμβάνει η εφαρμογή [65].

babel

Η βιβλιοθήκη babel χρησιμοποιήθηκε ως ο βασικός μεταγλωττιστής του κώδικα από ES6 σε ES5. Η babel υποστηρίζει την πιο πρόσφατη έκδοση JavaScript χρησιμοποιώντας συντακτικούς μετατροπείς (syntax transformers) [66].

mongoose

Η βιβλιοθήκη mongoose χρησιμοποιήθηκε για την προσθήκη κάποιων βασικών ιδιοτήτων στην βάση δεδομένων. Τέτοιες είναι η μοντελοποίηση της βάσης σε models με συγκεκριμένη δομή. Εξίσου σημαντική είναι η δυνατότητα σελιδοποίησης των collections εντός της βάσης και η επιστροφή σελιδοποιημένων αποτελεσμάτων σε κάθε αναζήτηση [67].

bcryptjs

Η βιβλιοθήκη bcryptjs χρησιμοποιείται για την κρυπτογράφηση ευαίσθητων πληροφοριών που αποθηκεύονται στη βάση δεδομένων. Συνήθως χρησιμεύει για την κρυπτογράφηση κωδικών πρόσβασης και την αποφυγή κατάχρησής τους από κακόβουλο λογισμικό [68].

5.3 Expo SDK & Expo Client

Η ανάπτυξη iOS εφαρμογών γίνεται μόνο στις αντίστοιχες iOS πλατφόρμες. Για την υλοποίηση εφαρμογών σε άλλες πλατφόρμες (πχ. Windows) είναι απαραίτητη η χρήση ενός ενδιάμεσου λογισμικού για την φιλοξενία της εφαρμογής. Το Expo είναι διαμορφωμένο ώστε να διευκολύνει τον προγραμματιστή με τον ταυτοχρονισμό της ανάπτυξης εφαρμογών και ελέγχου λειτουργίας.

Το Expo SDK παρέχει στον προγραμματιστή όλες τις μητρικές διεπαφές που διατίθενται στις μητρικές γλώσσες προγραμματισμού. Τα βασικά APIs όπως η κάμερα, οι ειδοποιήσεις, η τοποθεσία, η συλλογή φωτογραφιών κλπ. μπορούν να χρησιμοποιηθούν άμεσα, χωρίς πρόσθετη λογική, όπως θα γινόταν στον προγραμματισμό σε Swift. Στόχος του λογισμικού είναι να εξομαλύνει όσο το δυνατό περισσότερο την εμπειρία προγραμματισμού, αλλά και την εμπειρία χρήσης των διεπιφανειών, μειώνοντας τις διαφορές στο ελάχιστο. Η πλατφόρμα προσπαθεί να καλύπτει όλα τα δυνατά σενάρια χρήσης που εμφανίζονται στις σύγχρονες εφαρμογές, καθιστώντας τον κώδικα πλήρως δυναμικό.

Με την εγκατάσταση του Expo Client σε μια συσκευή iOS ο προγραμματιστής μπορεί να πραγματοποιεί έλεγχο σε πραγματικό χρόνο. Ο Expo Client φιλοξενεί την υπό ανάπτυξη εφαρμογή σε ένα περιβάλλον ειδικά διαμορφωμένο για αυτό τον σκοπό. Κατά τον έλεγχο της εφαρμογής, εμφανίζονται ειδικά μηνύματα σφάλματος στον προγραμματιστή που βοηθάνε στη διαδιακσία αποσφαλμάτωσης της εφαρμογής σε πραγματικό χρόνο. Έτσι, μειώνεται σε μεγάλο βαθμό η δυσκολία εύρεσης σφαλμάτων στον κώδικα, καθώς και ο χρόνος αποσφαλμάτωσης της εφαρμογής.

5.4 Visual Studio Code

Η πλατφόρμα προγραμματισμού που προτιμήθηκε στην ανάπτυξη της εφαρμογής ήταν το VSCode της *Microsoft*. Η επιλογή βασίστηκε στις αμέτρητες δυνατότητες που προσφέρει

η πλατφορμα αυτή. Σημαντικοί παράγοντες αποτέλεσαν επίσης η ευελιξία στον τρόπο προγραμματισμού, η αυξημένη ελαστικότητα ως προς τη διαμόρφωση του περιβάλλοντος προγραμματισμού εντός της πλατφόρμας, καθώς επίσης και η μεγάλη υποστήριξη που λαμβάνει η πλατφόρμα από την παγκόσμια προγραμματιστική κοινότητα.

Κεφάλαιο 6

Επίλογος

Στο παρόν κεφάλαιο συνοψίζεται η συνολική προσπάθεια για την σχεδίαση και την υλοποίηση του συστήματος της εφαρμογής μηνυμάτων σε διεπαφή χάρτη με τη βοήθεια του πληθοπορισμού. Θα εκφραστούν οι σκέψεις για μελλοντικές επεκτάσεις της εφαρμογής, για περαιτέρω λειτουργικότητες που μπορούν να προστεθούν, καθώς επίσης και για ανάγκες που μπορούν να εξυπηρετηθούν.

6.1 Τελικό Σύστημα

Το σύστημα σχεδιάστηκε με τέτοιο τρόπο ώστε να είναι ευέλικτο και γρήγορο στη χρήση. Από την πρώτη επαφή με την εφαρμογή, ο χρήστης εισάγεται στο νόημα της εφαρμογής άμεσα και αναλαμβάνει το ρόλο που του αναθέτει η εφαρμογή αβίαστα και χωρίς δυσκολία. Σε αυτό συνέβαλλαν τόσο οι σχεδιαστικές τεχνικές που χρησιμοποιήθηκαν, όσο και οι σύγχρονες τεχνολογίες με τις οποίες υλοποιήθηκαν οι διάφορες διεπιφάνειες της εφαρμογής. Η εξέλιξη πλέον των αρχιτεκτονικών σχεδίασης, του διαδικτύου, αλλά και των δυνατοτήτων των καθημερινών υπολογιστών είναι τέτοιες που η ανάπτυξη εφαρμογών δεν απαιτεί πολλά έξοδα, ενώ οι εφαρμογές δεν στερούνται τίποτα σε ταχύτητα ή πολυπλοκότητα. Επιπρόσθετα, η εκμάθηση των `frameworks` είναι πλέον σχετικά σύντομη και εύκολη, ενώ οι δυνατότητες που παρέχουν είναι πολύ εξελιγμένες και έτσι πολλές εφαρμογές μπορούν να παραχθούν σχετικά εύκολα και αποδοτικά. Μέσω αυτών των δυνατοτήτων το υλοποιημένο σύστημα μπόρεσε να παράσχει στους χρήστες της εφαρμογής τη δυνατότητα να δημιουργούν δημοσιεύσεις και σχόλια, ενώ δόθηκε και ένα ολοκληρωμένο κοινωνικό δίκτυο με σκοπό την επικοινωνία και παρακολούθηση των δράσεων μεταξύ χρηστών και εφαρμογής. Η δωρεάν διάθεση της εφαρμογής σε όλους τους χρήστες, καθιστά προσβάσιμη την απόκτησή της και τη χρήση της σε οποιαδήποτε τοποθεσία διαθέτει κάποιο τρόπο σύνδεσης στο διαδίκτυο. Η απλή επίσης σχεδιαστική τεχνοτροπία βοηθάει τους άπειρους χρήστες να μάθουν γρήγορα το σύστημα, χωρίς έτσι να απαιτείται μεγάλος χρόνος εκμάθησης.

6.2 Μελλοντικές επεκτάσεις

Το παραπάνω σύστημα μπορεί να χρησιμοποιηθεί ως μια βάση ώστε να εμπλουτιστεί με νέες δυνατότητες οι οποίες να παρέχουν στο χρήστη μια πιο ολοκληρωμένη εμπειρία σχετικά με την ενημέρωση σε θέματα πολιτισμικού περιεχομένου. Μερικά μελλοντικά θέματα προς διερεύνηση θα μπορούσαν να είναι:

- Δυνατότητα επισύναψης 3D αντικειμένων στις δημοσιεύσεις του χρήστη, τα οποία θα μπορούν να είναι φωτογραφίες, αρχεία βίντεο ή άλλης υποστηριζόμενης μορφής αρχεία (πχ. `.ttf`, `.fbx`, `.obj`, `.3ds` κλπ.).
- Δυνατότητα αλληλεπίδρασης του χρήστη με τα 3D αρχεία μιας δημοσίευσης, όπως προβολή, σχολιασμός και απάντηση με νέο μήνυμα αρχείου.
- Δυνατότητα βαθμολόγησης ενός `hotpot` με 3D αντικείμενα (όπως `stars`, `ratings`, `emoji reactions` κλπ.)
- Επιλογή αλληλεπίδρασης με την προσθήκη ενός `like button`.
- Καταγραφή περισσότερων λεπτομερειών για κάθε `hotspot` (όπως `likes`, `attendants`, `rating` κλπ.).
- Εμφάνιση χρονικής διάρκειας ισχύος στις λεπτομέρειες κάθε `hotspot` και διαγραφή μετά τη λήξη αυτού, ο χάρτης ανανεώνεται διαρκώς με τα ισχύοντα `hotspots`.
- Προσθήκη χρηστών σε λίστα φίλων για γρήγορη εύρεση και προβολή των δημοσιεύσεών τους.
- Δυνατότητα αποστολής μυνημάτων σε φίλους με την βοήθεια ενσωματωμένης πλατφόρμας γι΄ αυτό το σκοπό.

Βιβλιογραφία

[1] Jeff Sondermann, *American Press Institute*. Mobile and social media are intricately linked. https://www.americanpressinstitute.org/publications/reports/white-papers/mobile-and-social-media/ 2014. Last accessed on 07/03/2019.

[2] Ricardo Bilton. New data shows just how much social sharing has decreased since 2015 (and News Feed tweaks are just one factor). https://www.venturelean.team/hello-world/ 2018. Last accessed on 08/03/2019.

[3] Jessica Brown, *BBC Future*. Is social media bad for you? The evidence and the unknowns. http://www.bbc.com/future/story/20180104-is-social-media-bad-for-you-the-evidence-and-the-unknowns 2018. Last accessed on 08/03/2019.

[4] CrowdSourcingWeek. What is Crowdsourcing? https://crowdsourcingweek.com/what-is-crowdsourcing/ 2018. Last accessed on 08/03/2019.

[5] ISLAB Team, NTUA. About WITH http://withcrowd.eu/about 2018. Last accessed on 08/03/2019.

[6] Foursquare Developer Team. Foursquare API https://developer.foursquare.com/docs/api/endpoints 2018. Last accessed on 08/03/2019.

[7] Grace Fearon, *iStudent*. Our need to maintain social approval is actually making us lose what is best about ourselves – our individuality. https://www.independent.co.uk/student/istudents/our-need-to-maintain-social-approval-is-actually-making-us-lose-what-is-be
html 2016. Last accessed on 08/03/2019.

[8] Mansi Beniwal. Social Media and its Impact in Interpersonal Relationships. https://jarvee.com/

social-media-impact-interpersonal-relationships/ 2018. Last accessed on 08/03/2019.

[9] Ullah, Asmat. Client Side Scripting for Web Applications https://www.sqa.org.uk/e-learning/SiteHomeCD/page_26.htm 2007. Last accessed on 08/03/2019.

[10] Stroustrup, Bjarne. Bjarne Stroustrup's FAQ: What do you think of C++/CLI? http://www.stroustrup.com/bs_faq.html#CppCLI 2009. Last accessed on 08/03/2019.

[11] Gregory, Kate. Managed, Unmanaged, Native: What Kind of Code Is This? https://www.developer.com/net/cplus/article.php/2197621 2003. Last accessed on 08/03/2019.

[12] Meijer, Erik and Peter Drayton. Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War Between Programming Languages. Microsoft Corporation, 2005. Last accessed on 08/03/2019.

[13] L. Tratt. Dynamically Typed Languages. *Advances in Computers*, 77(2):149--184, July 2009. Last accessed on 08/03/2019.

[14] Apple Inc., *swift.org*. The Swift Linux Port https://swift.org/blog/swift-linux-port/ 2016. Last accessed on 10/03/2019.

[15] Chris Lattner. Chris Lattner's Homepage http://nondot.org/sabre/ 2014. Last accessed on 10/03/2019.

[16] Apple Worldwide Developers Conference, Session 102. Platforms State of the Union 2016. Last accessed on 10/03/2019.

[17] Rachel Metz, *MIT Technology*. Apple Seeks a Swift Way to Lure More Developers https://www.technologyreview.com/s/527821/apple-seeks-a-swift-way-to-lure-more-developers/ 2014. Last accessed on 10/03/2019.

[18] Harrison Weber, *VentureBeat*. Apple announces 'Swift', a new programming language for macOS iOS https://venturebeat.com/2014/06/02/apple-introduces-a-new-programming-language-swift-objective-c-with 2014. Last accessed on 10/03/2019.

[19] J. Gosling, J. Bill, G. Steele, G. Bracha, A. Buckley. The Java® Language Specification *Java SE 8th edition*, 2014. Last accessed on 11/03/2019.

[20] Computer Weekly. Write once, run anywhere? http://www.computerweekly.com/Articles/2002/05/02/186793/write-once-run-anywhere.htm 2002. Last accessed on 11/03/2019.

[21] Oracle Inc.. Design Goals of the Java™ Programming Language https://www.oracle.com/technetwork/java/intro-141325.html 2013. Last accessed on 11/03/2019 .

[22] R. McMillan, *wired.com*. Is Java Losing Its Mojo? https://www.wired.com/2013/01/java-no-longer-a-favorite/ 2013. Last accessed on 11/03/2019.

[23] Stephen O'Grady, *RedMonk*. The RedMonk Programming Language Rankings: January 2015 https://redmonk.com/sogrady/2015/01/14/language-rankings-1-15/ 2015. Last accessed on 11/03/2019.

[24] *langpop.com*. Programming Language Popularity https://web.archive.org/web/20090116080326/http://www.langpop.com/ 2009. Last accessed on 11/03/2019.

[25] Tata McGraw. Object-oriented Programming with Java: Essentials and Applications. *Hill Education*, 1:30--35, 2009. Last accessed on 11/03/2019.

[26] Sun Microsystems, *WaybackMachine*. JAVASOFT SHIPS JAVA 1.0 - *Programming environment available free for developers* https://web.archive.org/web/20070310235103/http://www.sun.com/smi/Press/sunflash/1996-01/sunflash.960123.10561.xml 2018. Last accessed on 11/03/2019.

[27] Alex Mullis, *Android Authority*. How to install the Android SDK (Software Development Kit) https://www.androidauthority.com/how-to-install-android-sdk-software-development-kit-21137/ 2016. Last accessed on 11/03/2019.

[28] Android Developers. Introduction to Android https://developer.android.com/guide/index.html 2017. Last accessed on 11/03/2019.

[29] Android Developers. Tools Overview https://developer.android.com/studio/command-line/ 2012. Last accessed on 11/03/2019.

[30] Stoyan Stefanov. JavaScript Patterns. *O'Reilly Media, Inc.*, 1(1): 5--10, 2010. Last accessed on 14/03/2019.

[31] *w3techs.com*. Usage Statistics of JavaScript for Websites https://w3techs.com/technologies/details/cp-javascript/all/all 2015. Last accessed on 14/03/2019.

[32] *pabbly.com*. NodeJS Event Loops https://www.pabbly.com/tutorials/node-js-event-loops/. 2015. Λαστ αςςεσσεδ ον 14/03/2019.

[33] Jesse James Garrett, *AdaptivePath*. Ajax: A New Approach to Web Applications https://adaptivepath.org/ideas/ajax-new-approach-web-applications/ 2005. Last accessed on 14/03/2019.

[34] Modzilla Developer Network Web Docs, *MDN.com*. Ajax - Web developer guides https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX 2018. Last accessed on 14/03/2019.

[35] David Flanagan. JavaScript - The Definitive Guide. *O'Reilly, Sebastpool, CA*, 1(5):495--450, 2006.

[36] Steve Burbeck. Applications Programming in Smalltalk-80:How to use Model{View{Controller (MVC) https://web.archive.org/web/20120729161926/http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html 1992. Last accessed on 14/03/2019.

[37] Facebook, *reactjs.org*. Components and Props https://reactjs.org/docs/components-and-props.html#props-are-read-only 2018. Last accessed on 17/03/2019.

[38] React Blog, *reactjs.org*. Refs and DOM https://reactjs.org/docs/refs-and-the-dom.html 2018. Last accessed on 17/03/2019.

[39] Facebook. JSX Specification https://facebook.github.io/jsx/ 2018. Last accessed on 17/03/2019.

[40] *reactjs.org*, tutorials. Tutorial: intro to React https://reactjs.org/tutorial/tutorial.html 2017. Last accessed on 17/03/2019.

[41] Bonnie Eisenman. Learning React Native. *O'Reilly Media, Inc.*, chapter 1, 2015. Last accessed on 18/03/2019.

[42] Tal Kol, *Wix.com*. Building a React Native App for 80 Million Users. From *ReactNext Conference 2016, Tel Aviv*, https://www.youtube.com/watch?v=abSNo2P9mMM&feature=youtu.be&t=17m34s, minutes 17:37--33:20, 2016. Last accessed on 18/03/2019.

[43] Natalia Chrzanowska. 13 Great Examples of React Native Apps https://www.netguru.com/blog/13-great-apps-written-with-react-native 2019. Last accessed on 18/03/2019.

[44] Marvin Frachet, *hackermoon.com*. Understanding the React Native bridge concept https://hackernoon.com/understanding-react-native-bridge-concept-e9526066ddb8 2018. Last accessed on 18/03/2019.

[45] Tomislav Capan, *toptal.com*. Why Use Node.js? https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js 2013. Last accessed on 19/03/2019.

[46] Kenneth Peeples, *dzone.com*. What are the Benefits of Node.js? https://dzone.com/articles/what-are-benefits-nodejs 2015. Last accessed on 19/03/2019.

[47] Node,js Foundation, *nodejs.org*. Node.js Docs https://nodejs:org. Last accessed on 19/03/2019.

[48] Expolarions of an Engineer, *timcosta.io*. The Node.js Event Loop https://www.timcosta.io/the-node-js-event-loop/ 2016. Last accessed on 19/03/2019.

[49] RESTful Web Services Architecture. Of the *World Wide Web Consortium*, Chapter 3.1.3: Relationship to the World Wide Web and REST Architectures, February 2004. Last accessed on 19/03/2019.

[50] R. T. Fielding Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). *Chapter 5: Representational State Transfer (REST)*. University of California, Irvine. 2000. Last accessed on 19/03/2019.

[51] L. Richardson, M. Amundsen RESTful Web APIs, *O'Reilly Media*. 2013 *ISBN 978-1-449-35806-8*.

[52] Davis Kerby Why MongoDB is the Way to Go, *DZone*. https://dzone.com/articles/why-mongodb-is-worth-choosing-find-reasons 2015. Last accessed on 20/03/2019.

[53] *ClusterHQ*. Ridiculously fast MongoDB replica recovery Part 1 of 2 http://clusterhq.com/2016/03/14/ridiculously-fast-mongodb-replica-recovery-with-flocker/. Last accessed on 20/03/2019.

[54] *Severalnines*. Turning MongoDB Replica Set to a Sharded Cluster https://severalnines.com/blog/turning-mongodb-replica-set-sharded-cluster 2013. Last accessed on 20/03/2019.

[55] *Compose*. GridFS & MongoDB: Pros And Cons https://www.compose.com/articles/gridfs-and-mongodb-pros-and-cons/ 2014. Last accessed on 20/03/2019.

[56] Md. Malick, *ExpertsTown*. MongoDB Overview https://www.hugedomains.com/domain_profile.cfm?d=expertstown&e=com 2013. Last accessed on 20/03/2019.

[57] MongoDB Docs, *docs.mongodb.com*. Aggregation | MongoDB Manual https://docs.mongodb.com/manual/aggregation/. Last accessed on 20/03/2019.

[58] MongoDB Docs, *docs.mongodb.com*. Map-Reduce | MongoDB Manual https://docs.mongodb.com/manual/core/map-reduce/. Last accessed on 20/03/2019.

[59] J. Bradley, N. Sakimura, M. Jones, JSON Web Token (JWT) *Mastering Identity and Access Management with Microsoft Azzure*, 1(1):84, ISBN: 9781785887888.

[60] reduxjs, *github.com*. Redux thunk Middleware https://github.com/reduxjs/redux-thunk. Last accessed on 2/05/2019.

[61] axios, *npmjs.com*. Promise based HTTP client for the browser and node.js https://www.npmjs.com/package/axios. Last accessed on 2/05/2019.

[62] NativeBase, *nativebase.io*. Essential cross-platform UI components for React Native Vue Native https://docs.nativebase.io/. Last accessed on 2/05/2019.

[63] aksonov, *github.com*. The first declarative React Native router https://github.com/aksonov/react-native-router-flux. Last accessed on 2/05/2019.

[64] react-native-community, *github.com*. React Native Mapview component for iOS + Android https://github.com/react-native-community/react-native-maps. Last accessed on 2/05/2019.

[65] Passportjs, *passportjs.org*. Simple, unobtrusive authentication for Node.js http://www.passportjs.org/packages/. Last accessed on 2/05/2019.

[66] Babel, *babeljs.io*. The compiler for next generation JavaScript https://babeljs.io/docs/en/. Last accessed on 2/05/2019.

[67] Automattic, *mongoosejs.com*. elegant mongodb object modeling for node.js https://github.com/Automattic/mongoose. Last accessed on 2/05/2019.

[68] npm, *npmjs.com*. Encrypted data https://www.npmjs.com/package/bcryptjs. Last accessed on 2/05/2019.

[69] Expo, *expo.io*. A set of tools, libraries and services you can use to build native iOS and Android apps faster than ever before! https://docs.expo.io/versions/latest/. Last accessed on 2/05/2019.

Κατάλογος Ακρωνύμων

IT Information Technology

NGO Non Governmental Organization

API Application programming Interface

URL Uniform Resource Locator

SDK Software Development Kit

CLR Common Language Runtime

CIL Common Intermediate Language

OS Operating System

IDE Integrated Development Environment

WWDC Worldwide Development Conference

OO Object Oriented

WORA Write Once Run Anywhere

JS JavaScript

ES EcmaScript

HTML Hypertext Markup Language

CSS Cascading Style Sheets

DOM Document Object Model

AJAX Asynchronous Javascript And XML

I/O Input Output

SPA Single Page Application

MVC Model View Controller

APP Application

PROP Property

JSX JavaScript and XML

VDOM Virtual Document Object Model

UI User Interface

XML Extensible Markup Language

JSON JavScript Object Notation

PHP Hypertext preprocessor

NPM Node Package Manager

RAM Random Access Memory

GB GigaByte

MB MegaByte

1M One Million

REST Representational State Transfer

HTTP Hypertext Tranfer Protocol

FS File System

OAuth Open Authentication

JWT JSON Web Token

FAB Floating Action Button

KM kilometer

Γλωσσάριο

Αγγλικός όρος	Ελληνικός όρος
social media	μέσα κοινωνικής δικτύωσης
facebook	μία από τις δημοφιλέστερες εταιρείες και υπηρεσίες κοινωνικής δικτύωσης σήμερα
instagram	υπηρεσία κοινωνικής δικτύωσης για κοινή χρήση φωτογραφιών και βίντεο
snapchat	εφαρμογή ανταλλαγής μηνυμάτων πολυμέσων
twitter	αμερικανική ηλεκτρονική υπηρεσία ειδήσεων και κοινωνικής δικτύωσης
tinder	εφαρμογή για κοινωνική αναζήτηση βασισμένη σε τοποθεσίες
New York Times	αμερικανική εφημερίδα
crowdsourcing	πληθοπορισμός
withcrowd	ψηφιακή πλατφόρμα αρχειοθέτησης ευρυμάτων της ευρωπαϊκής πολιτισμικής κληρονομιάς
portal	πύλη
repository	ψηφιακή αποθήκη
youtube	εφαρμογή για κοινή χρήση βίντεο
foursquare	εφαρμογή τοπικής αναζήτησης κοινωνικών τόπων
endpoint	τελικό σημείο
web application	εφαρμογή που τρέχει σε φυλλομετρητή
native application	μητρική εφαρμογή
programming language	γλώσσα προγραμματισμού
managed	διαχειριζόμενη
compiled	μεταγλωττισμένη
dynamic	δυναμική
native	μητρική
compiler	μεταγλωττιστής
high-level	υψηλού επιπέδου
overhead	χρόνος καθυστέρησης
low-level	χαμηλού επιπέδου
cross-platform	μεταξύ πολλών πλατφόρμων

bytecode	κώδικας ψηφίων
ios	λειτουργικό σύστημα που ανήκει στην Apple Inc.
android	λειτουργικό σύστημα που ανήκει στην Google Inc.
developer	προγραμματιστής
concurrent	ταυτόχρονος
class	κλάσση
web browser	φυλλομετρητής ιστού
web page	ιστοσελίδα
object	αντικείμενο
integer	ακέραιος αριθμός
floating point	μεταβλητή κινητής υποδιαστολής
boolean value	μεταβλητή λογικής τιμής
smartphone	έξυπνη κινητή συσκευή
tablet	φορητή συσκευή
debugger	διορθωτής σφαλμάτων
emulator	προσομοιωτής
tablet	φορητή συσκευή
functional	συναρτησιακός
event-driven	οδηγούμενο από γεγονότα
client-side	πλευρά πελάτη
object-oriented	αντικειμενοστραφής
fade out/in	ξεθώριασμα
slide out/in	ολίσθηση
script-objects	αντικείμενα κώδικα
host-objects	αντικείμενα φιλοξενίας
server	εξυπηρετητής/διακομιστής
requests	αιτήματα
event loop	βρόχος συμβάντων
non-blocking execution	εκτέλεση χωρίς μπλοκάρισμα
callback function	συνάρτηση ανάκλησης
callback content	περιεχόμενο ανάκλησης
run to completion logic	λογική πλήρους διεκπαιρέωσης
browser	περιηγητής
backend	η πλευρά του εξυπηρετητή
frontend	η πλευρά του πελάτη
desktop application	εφαρμογή επιφάνειας εργασίας
model	μοντέλο
view	όψη
controller	ελεγκτής
container class	κλάση-δοχείο
presentation logic	λογική παρουσίασης

mobile application	εφαρμογή για φορητή συσκευή
component	στοιχείο
component-based architecture	αρχιτεκτονική βασισμένη σε στοιχεία
property	ιδιότητα
one-way data binding	μονόδρομη ροή δεδομένων
state	κατάσταση
stateful component	στοιχείο με κατάσταση
child-component	στοιχείο-παιδί
parent-component	στοιχείο-πατέρας
hook	συναρτήσεις-γάντζοι
lifecycle method	συνάρτηση που εκτελείται σε κάποιο συγκεκριμένο χρονικό σημείο κατά τον κύκλο ζωής της εφαρμογής
framework	πλαίσιο
developer experience	προγραμματιστική εμπειρία
rebuild	επαναμεταγλώττιση
design	σχεδίαση
Chrome	περιηγητής της Google
Safari	περιηγητής της Apple
text editor	επεξεργαστής κειμένου
code-reuse	επαναχρησιμότητα κώδικα
knowledge sharing	κοινή χρήση γνώσης
resource saving	εξοικονόμηση πόρων
airbnb	ιδιωτική ηλεκτρονική εταιρεία που προσφέρει υπηρεσίες διαμονής και φιλοξενίας
skype	εφαρμογή τηλεπικοινωνίας
bridge	γέφυρα
realm	βασίλειο, σφαίρα, εδώ: κατηγορία
interoperable language	διαλειτουργική γλώσσα
node	κόμβος
event queue	ουρά συμβάντων
worker-thread	νήμα-εργάτης
worker-thread pool	δεξαμενή νημάτων εργασίας
interpreting	μεταγλώττιση
repersentational state transfer	αναπαραστατική μεταβίβαση κατάστασης
router	δρομολογητής
route	διαδρομή
response payload	φορτιο απόκρισης
key	κλειδί
value	τιμή
string	συμβολοσειρά
boolean	λογική τιμή

array	πίανακας
null	κενός χαρακτήρας
non-relational	μη-σχετικιστικός
document-oriented	εγγραφοστραφής
schema	μοντέλο εγγράφου σε MongoDB
collection	συλλογή
ad hoc query	αναζήτηση συγκεκριμένου σκοπού
field search	αναζήτηση με βάση πεδίο
range search	αναζήτηση με βάση συνθήκη
regular expression search	αναζήτηση με βάση μοτίβο
replication	αναπαραγωγή/αντιγραφή
set	σύνολο, ομάδα
indexing	δεικτοδότηση
load balancing	εξισορρόπηση φορτίου
sharding	κοπή, τεμαχιοποίηση
hashing	διαδικασία κρυπτογράφησης δεδομένων
file storage	αποθήκευση αρχείων
aggregation	άθροιση
pipeline	αγωγός
map-reduce function	μέθοδος αθροιστικής χαρτογράφησης
single-purpose aggregation	άθροιση μονού σκοπού
pagination	σελιδοποίηση
open authentication	σύστημα ταυτοποίησης ανοικτού τύπου
access token	σύμβολο πιστοποίησης
username	όνομα χρήστη
email	ηλεκτρονική διεύθυνση
password	κωδικός πρόσβασης
refresh token	σύμβολο ανανέωσης
register/signup form	φόρμα εγγραφής
handshake	χειραψία, διαδικασία ανταλλαγής
logout	αποσύνδεση
login/sigin form	φόρμα σύνδεσης
authorization header	επικεφαλίδα εξουσιοδότησης
passport strategy	στρατηγική ταυτοποίησης χρηστών
boilerplate	βασική δομή κώδικα
profile	προφίλ χρήστη
hotspot	σημείο ενδιαφέροντος στην εφαρμογή
comment	σχόλιο
point of interest	σημείο ενδιαφέροντος
performance	απόδοση
recoverability	ανάκτηση

documentation	βιβλιογραφία, υλικό υποστήριξης
security	ασφάλεια
scalability	επεκτασιμότητα
hosting	η διαδικασία κατά την οποία μια εφαρμογή φιλοξενείται από έναν κεντρικό server
use cases	σενάρια χρήσης
screen	οθόνη
register	εγγραφή
login	σύνδεση
fullname	πλήρες όνομα
confirm password	επιβεβαίωση κωδικού πρόσβασης
birthdate	ημερομηνία γέννησης
city	πόλη
gender	γένος
description	περιγραφή
validity	χρονική διάρκεια ισχύος
coffee	καφές
food	φαγητό
drinks	ποτά
sights	αξιοθέατα
arts	τέχνες
find my location	επαναφορά στην τοποθεσία χρήστη
account stats	στατιστικά δεδομλένα λογαριασμού
insights	πρόσθετες πληροφορίες
audience	κοινό
common	κοινότοπος
swipe	σύρω
spinner	ένδειξη προόδου μιας διαδικασίας
query	αναζήτηση
like button	πλήκτρο εκδήλωσης ενδιαφέροντος
attendants	παρευρισκόμενοι
rating	βαθμολογία
emoji reaction	αντίδραση με αυτοκολλητάκι

Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Σχεδιασμός και Υλοποίηση Κινητής Εφαρμογής Πληθοπορισμού για Αξιολόγηση και Ανασκόπηση Τόπων Πολιτιστικού Περιεχομένου σε Διεπαφή Χάρτη

(Εγχειρίδιο Εγκατάστασης)

Διπλωματικη Εργασια

του

ΕΥΘΥΜΙΟΥ ΠΕΓΚΑ

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Εργαστηριο Ευφυων Συστηματων, Περιεχομενου και Αλληλεπιδρασης

Αθήνα, Μάιος 2019

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Ευφυών Συστημάτων, Περιεχομένου και Αλληλεπίδρασης

Σχεδιασμός και Υλοποίηση Κινητής Εφαρμογής Πληθοπορισμού για Αξιολόγηση και Ανασκόπηση Τόπων Πολιτιστικού Περιεχομένου σε Διεπαφή Χάρτη

(Εγχειρίδιο Εγκατάστασης)

Διπλωματικη Εργασια

του

ΕΥΘΥΜΙΟΥ ΠΕΓΚΑ

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Μαίου 2019.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Στάμου	Ανδρέας Γ. Σταφυλοπάτης	Νικόλαος Παπασπύρου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάιος 2019

(Υπογραφή)

.....

Ευθυμιος Πεγκας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Ευφυών Συστημάτων, Περιεχομένου και Αλληλεπίδρασης

Κεφάλαιο 1

Server

Σε αυτό το κεφάλαιο θα επεξηγηθεί αναλυτικά η διαδικασία εγκατάστασης του εξυπηρετητή αιτημάτων της εφαρμογής. Κάθε βήμα συνοδεύεται από την εντολή που αντιστοιχεί στις διεργασίες που πρέπει να εκτελεστούν από τη γραμμή εντολών, καθώς επίσης και από πρόσθετα σχόλια που αφορούν πληροφορίες σχετικές με την εντολή αυτή.

1.1 Dependencies

- npm *or* yarn
- mongoDB
- express
- nodemon
- rimraf

Αφού αλλάξετε το directory ώστε να αντιστοιχεί σε αυτό όπου βρίσκονται τα αρχεία του server, ξεκινήστε τη διαδικασία εγκατάστασης των dependencies τρέχοντας την ακόλουθη εντολή από τη γραμμή εντολών

```
npm install or yarn
```

Όλα τα υπόλοιπα dependencies μπορούνε να βρεθούνε και να εγκατασταθούνε ανατρέχοντας στις επίσημες ιστοσελίδες τους.

1.2 Execution

Στη συνέχεια, ανοίξτε το αρχείο `package.json` και κάτω από το `prop` με όνομα '*scripts*' προσθέστε το εξής `object`

```
{
  "build:watch": "babel -w --out-dir=dist ./src",
  "devserver": "nodemon dist/index.js",
  "clean": "rimraf dist"

}
```

Η πρώτη εντολή αφορά τη δημιουργία ενός νέου φακέλου με όνομα `dist` όπου και θα αποθηκευτούν τα αρχεία που θα δημιουργηθούν μετά το πέρας της εντολής αυτής. Η δεύτερη εντολή ξεκινάει τον `server` τρέχοντας το αρχείο `index.js` που έχει παραχθεί από την προηγούμενη εντολή στο φάκελο `dist`. Εάν θελήσουμε να διαγράψουμε τον φάκελο, εκτελούμε την τρίτη εντολή.

Ξεκινάμε τον `server` τρέχοντας την εντολή

```
npm run devserver or yarn devserver
```

Με αυτη την εντολή ξεκινλάμε και τη βάση δεδομένων που τρέχει στο τοπικό δίκτυο όπου είμαστε συνδεδεμένοι. Μόλις ολοκληρωθεί η διαδικασία έναρξης του `server` θα εμφανιστεί το ακόλουθο μήνυμα στη γραμμή εντολών

```
Server is listening on port:  3000
Mongodb is up and running!
```

Κεφάλαιο 2

Mobile Application

Σε αυτό το κεφάλαιο θα επεξηγηθεί αναλυτικά η διαδικασία εγκατάστασης της διεπιφάνειας της εφαρμογής. Κάθε βήμα συνοδεύεται από την εντολή που αντιστοιχεί στις διεργασίες που πρέπει να εκτελεστούν από τη γραμμή εντολών, καθώς επίσης και από πρόσθετα σχόλια που αφορούν πληροφορίες σχετικές με την εντολή αυτή.

2.1 Dependencies

- `visual studio code editor` της `Microsoft`
- `npm` *or* `yarn`
- `react native`
- `expo-cli`

Αφού αλλάξετε το `directory` ώστε να αντιστοιχεί σε αυτό όπου βρίσκονται τα αρχεία του `client`, ξεκινήστε τη διαδικασία εγκατάστασης των `dependencies` τρέχοντας την ακόλουθη εντολή από τη γραμμή εντολών

```
npm install or yarn
```

Όλα τα υπόλοιπα `dependencies` μπορούνε να βρεθούνε και να εγκατασταθούνε ανατρέχοντας στις επίσημες ιστοσελίδες τους.

2.2 Execution

Για να προβάλλετε τη διεπιφάνεια χρήστη της εφαρμογής στην `iOS` κινητή συσκευή σας, εγκαταστήστε την εφαρμογή Εξπο από το `App Store`. Αφού η εφαρμογή εγκατασταθεί, εισέλθετε στην εφαρμογή και δημιουργήστε ένα λογαριασμό (ή συνδεθείτε στην περίπτωση που έχετε ήδη λογαριασμό).

Από τη γραμμή εντολών, τρέξτε τώρα την ακόλουθη εντολή

```
expo start
```

Η εντολή αυτή θα κάνει `build` και θα κατεβάσει από έναν `server` την συμπιεσμένη μορφή (`bundle`) του κώδικα της εφαρμογής. Μόλις ολοκληρωθεί η παραπάνω διαδικασία, η εφαρμογή είναι έτοιμη στην κινητή συσκευή. Για περισσότερες πληροφορίες σχετικά με το `Expo` ανατρέξτε στην επίσημη ιστοσελίδα `https://expo.io/tools` .