ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΠΛΑΤΦΟΡΜΑΣ ΓΙΑ ΤΗ ΔΙΕΡΕΥΝΗΣΗ ΤΗΣ ΧΡΗΣΗΣ ΤΟΥ ΔΗΜΟΣΙΟΥ ΧΩΡΟΥ ΚΑΤΑ ΤΗΝ ΔΙΑΡΚΕΙΑ ΤΗΣ ΠΑΝΔΗΜΙΑΣ COVID-19

Ανέστου Ευαγγελία

Τριμελής επιτροπή:

Στεφανέας Πέτρος (επιβλέπων), Αναπληρωτής Καθηγητής

Αρβανιτάκης Αλέξανδρος, Επίκουρος Καθηγητής

Παυλοπούλου Καλλιόπη, Ε.ΔΙ.Π.

Αθήνα, Οκτώβριος 2021

ΠΕΡΙΕΧΟΜΕΝΑ

Ευχαριστίες

Καταρχάς, θα ήθελα να ευχαριστήσω τον Αναπληρωτή Καθηγητή και επιβλέποντα της παρούσας διπλωματικής εργασίας κύριο Πέτρο Στεφανέα για το γεγονός πως μέσω των μαθημάτων του και του τρόπου διδασκαλίας του συνέβαλε στην ανάπτυξη του ενδιαφέροντος μου για την Πληροφορική καθώς και για τον τρόπο προσέγγισης του στην διαδικασία αναζήτησης θέματος. Επίσης, θα ήθελα να ευχαριστήσω τον κύριο Αλέξανδρο Αρβανιτάκη και την κυρία Καλλιόπη Παυλοπούλου για την τιμή που μου κάνουν να συμμετάσχουν στην επιτροπή εξέτασης της εργασίας.

Στην συνέχεια, θα ήθελα να ευχαριστήσω τον Χρήστο Ζαχαρόπουλο για την πολύτιμη βοήθεια του καθ' όλη την διάρκεια της διπλωματικής, τον Στέφανο Τσιγδινό που μέσα από την συζήτηση μας προέκυψε το θέμα και αναπτύχθηκαν οι ερωτήσεις για την διπλωματική εργασία και την Νεφέλη Τσιάμη για την δημιουργία της εικόνας που κοσμεί την αρχική σελίδα της εφαρμογής.

Τέλος, το ευχαριστώ είναι πολύ λίγο για να εκφράσει την ευγνωμοσύνη μου προς την οικογένεια μου καθώς και το Νίκο και τη Λουκία για τη στήριξη τους και για την πίστη τους σε εμένα.

Περίληψη

Στόχος αυτής της διπλωματικής εργασίας είναι η ανάπτυξη διαδικτυακής εφαρμογής για την έρευνα της αλλαγής στη συμπεριφορά των πολιτών προς δημόσιους χώρους σε αστικό περιβάλλον κατά τη διάρκεια της καραντίνας για τον Covid-19. Η εργασία περιέχει μια περιγραφή ολόκληρης της διαδικασίας ανάπτυξής της. Εκτός από την περιγραφή, το τελικό αποτέλεσμα της διπλωματικής εργασίας είναι επίσης μια λειτουργική εφαρμογή ιστού.

Λέξεις κλειδιά: HTML, CSS, Bootstrap, JavaScript, NodeJs, ExpressJs, εφαρμογή ιστού, εφαρμογή, Covid-19, Δημόσιοι χώροι, αστικό περιβάλλον

Abstract

The aim of this thesis is the development of a web application in order to explore the change in the behavior of the citizens in an urban environment towards public spaces during the Covid-19 quarantine. The thesis contains a description of the whole process of its development. In addition, the final output of the thesis is also a functioning web application.

Keywords: HTML, CSS, Bootstrap, JavaScript, NodeJs, ExpressJs, web application, web app, website, Covid-19, Public spaces, urban environment

ΓΛΩΣΣΑΡΙ[1]

- Back-End: παρασκήνιο
- Browser: πρόγραμμα πλοήγησης
- Buffer: κύκλωμα / χώρος απομόνωσης
- Client: πελάτης
- Client-Side: από την μεριά του πελάτη
- Compiler: μεταγλωττιστής
- Data: δεδομένα
- Database: βάση δεδομένων
- Domain name: όνομα τομέα
- Framework: πλαίσιο, σκελετός
- Front-End: εμπρόσθιο τμήμα/κομμάτι, αρχικό κομμάτι προγράμματος σε μια σειριακή ροή διαδικασιών-επεξεργασίας πληροφορίας
- Host: εξυπηρετητής, οικοδεσπότης υπολογιστής
- Hyperlink: υπερσύνδεσμος
- Hypertext: υπερκείμενο
- Input: είσοδος
- Interface: διεπαφή
- Internet: Διαδίκτυο
- Layer: στρώμα
- Library: βιβλιοθήκη
- Link: σύνδεσμος
- Node: κόμβος
- Output: έξοδος
- Pipe/Pipeline: διοχέτευση
- Port: θύρα
- Router: δρομολογητής
- Script: σενάριο
- Server: διακομιστής
- Shell: κέλυφος/φλοιός
- Stream: ροή δεδομένων/ροή
- User Interface: διεπαφή χρήστη
- Web application: διαδικτυακή εφαρμογή
- Web Developer: προγραμματιστής ιστού

[1] Πηγή: https://el.wikipedia.org/wiki/%CE%92%CE%B9%CE%BA%CE%B9%CF%80%CE%B1%CE%AF%CE%B4%CE%B5%CE%B9%CE%B1:%CE%9C%CE%B5%CF%84%CE%AC%CF%86%CF%81%CE%B1%CF%83%CE%B7_%CF%8C%CF%81%CF%89%CE%BD_%CF%80%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%B9%CE%BA%CE%AE%CF%82

1. ΕΙΣΑΓΩΓΗ

1.1 ΣΚΟΠΟΣ

Τον τελευταίο ενάμιση χρόνο η παγκόσμια κοινωνία αντιμετωπίζει μια πρωτοφανή κατάσταση. Με την εμφάνιση του Covid-19 σε μια προσπάθεια μείωσης των ανθρώπινων απωλειών οι πολίτες κλήθηκαν να περιορίσουν τις μετακινήσεις τους, ακόμα και να μένουν αποκλειστικά στο σπίτι ανά περιόδους. Δεν αρμόζει στην παρούσα εργασία να γίνει κριτική για τη διαχείριση της κατάστασης, όμως ήταν η συγκεκριμένη συνθήκη που αποτέλεσε «έμπνευση» για το θέμα της διπλωματικής εργασίας. Μέσα σε πολύ ιδιαίτερες συνθήκες που ο κόσμος προσπάθησε να ανταπεξέλθει κοινωνικά και ψυχολογικά, οι δημόσιοι χώροι έγιναν ο τόπος συνάντησης, ένας χώρος έκφρασης και κοινωνικοποίησης. Με αφορμή την παρατήρηση αυτής της αλλαγής της στάσης ως προς αυτούς τους χώρους και μέσα από τη συζήτηση προέκυψε η ιδέα για το θέμα της διαδικτυακής εφαρμογής. Πρωταρχικός σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη και η λειτουργικότητα μιας διαδικτυακής εφαρμογής. Μια εφαρμογή όμως δεν μπορεί να σταθεί από μόνη της, πρέπει να έχει και κάποιο περιεχόμενο. Με αυτό το σκεπτικό, η εφαρμογή θα αναπτυχθεί ώστε να διερευνήσει εάν τελικά υπήρξε και σε ποιο βαθμό αλλαγή της χρήσης του δημόσιου χώρου κατά τη διάρκεια της πανδημίας.

Τα τελευταία χρόνια η χρήση του διαδικτύου και των έξυπνων κινητών συσκευών έχει εκτοξευθεί. Ενδιαφέρον παρουσιάζουν οι δυνατότητες που προσφέρουν τα διαδικτυακά μέσα για τη διεξαγωγή κάποιας έρευνας. Αντί να υπάρχουν μόνο οι απλές φόρμες, όπως για παράδειγμα της εταιρείας Google, η μορφή των ερωτηματολογίων μπορεί να πάρει τη μορφή εφαρμογής που μπορεί να καταστήσει τη διαδικασία πιο ευχάριστη έστω και αν αυτό γίνεται μόνο σε αισθητικό επίπεδο. Εάν εμπλουτιστεί ακόμα περισσότερο αποτελεί μια πιο «διασκεδαστική» εμπειρία για τον χρήστη στην οποία καλείται να συμμετέχει και αυτό ίσως οδηγήσει και στην περαιτέρω ενασχόλησή του με το θέμα. Σε ένα διαδικτυακό περιβάλλον όπου βρίσκονται «μαζεμένα» για παράδειγμα, κάποιες πηγές και η διαδικασία διεξαγωγής της έρευνας, δίνεται στον χρήστη η δυνατότητα «με ένα κλικ» να διαβάσει και ενημερωθεί για το θέμα που διαπραγματεύεται η έρευνα αντί να αφιερώσει χρόνο για να βρει μόνος του πηγές.

Συνοψίζοντας, σκοπός της παρούσας διπλωματικής εργασίας είναι η υλοποίηση μιας ολοκληρωμένης διαδικτυακής εφαρμογής έτοιμης προς χρήση, που θα μπορεί να αποτελέσει εργαλείο στη διερεύνηση της αλλαγής της χρήσης των δημόσιων χώρων από τους κατοίκους της Αθήνας.

1.2 ΔΟΜΗ ΕΡΓΑΣΙΑΣ

Αρχικά, στο δεύτερο κεφάλαιο θα αναλυθούν οι περισσότερες και κυριότερες τεχνολογίες και λογισμικά που χρησιμοποιήθηκαν για να υλοποιηθεί η εφαρμογή.
Το τρίτο κεφάλαιο θα ασχοληθεί με το λειτουργικό κομμάτι της εφαρμογής. Θα αναλυθεί η μέθοδος υλοποίησης της, δηλαδή ο τρόπος που χρησιμοποιήθηκαν οι τεχνολογίες που αναφέρθηκαν στο πρώτο κεφάλαιο ώστε να αναπτυχθεί επιτυχώς.
Το τέταρτο κεφάλαιο, θα ασχοληθεί με το αισθητικό κομμάτι της εφαρμογής. Θα γίνει η παρουσίαση της εφαρμογής ώστε να γίνει μια οπτικοποίηση της εφαρμογής για τον αναγνώστη.
Τέλος, στο πέμπτο και τελευταίο κεφάλαιο, αρχικά θα βγουν τα συμπεράσματα και έπειτα θα γίνει αναφορά στις δυσκολίες που αντιμετωπίστηκαν και στις βελτιώσεις που θα μπορούσαν να γίνουν μελλοντικά.

2. Η ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

2.1 ΓΕΝΙΚΑ

Τη σημερινή εποχή που η τεχνολογία κυριαρχεί στην καθημερινότητα, οι τρόποι ανάπτυξης μιας εφαρμογής είναι αμέτρητες. Υπάρχουν πάρα πολλές γλώσσες προγραμματισμού, πολλές πλατφόρμες π.χ. wordpress, πολλοί τύποι εφαρμογών (π.χ. desktop, mobile), πολλά περιβάλλοντα ανάπτυξης κλπ. Σε αυτό το κεφάλαιο θα αναπτύξουμε τις τεχνολογίες που επιλέχθηκαν για την ανάπτυξη της συγκεκριμένης εφαρμογής.

2.2 HTML – CSS – BOOTSTRAP

2.2.1 HTML (HyperText Markup Language)

Το 1980, ο Tim Berners-Lee, φυσικός στο CERN, πρότεινε και προτυποποίησε το INQUIRE, ένα σύστημα για τους ερευνητές του CERN με το οποίο θα μπορούσαν να μοιράζονται έγγραφα. Το 1989, ο Berners-Lee έγραψε ένα υπόμνημα που πρότεινε ένα σύστημα υπερκειμένου που

βασίζεται στο Διαδίκτυο. Ο Berners-Lee διευκρίνισε την HTML και έγραψε το πρόγραμμα περιήγησης και διακομιστή στα τέλη του 1990. Η πρώτη μορφή της HTML έγινε διαθέσιμη τέλη του 1991.

Η HTML (HyperText Markup Language) είναι από τις πιο βασικές τεχνολογίες για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών, καθώς αποτελεί τη δομή της σελίδας/εφαρμογής. Υπερκείμενο (Hypertext) σημαίνει ότι το έγγραφο περιέχει συνδέσμους που επιτρέπουν στον αναγνώστη να μεταβεί σε άλλα μέρη του εγγράφου ή σε ένα τελείως άλλο έγγραφο. Η γλώσσα σήμανσης (Markup Language) είναι ένας τρόπος με τον οποίο οι υπολογιστές μιλούν μεταξύ τους για να ελέγχουν τον τρόπο επεξεργασίας και παρουσίασης του κειμένου. Για να γίνει αυτό, το HTML χρησιμοποιεί δύο πράγματα: ετικέτες (tags) και χαρακτηριστικά(attributes). Οι ετικέτες και τα χαρακτηριστικά είναι η βάση του HTML, συνεργάζονται αλλά εκτελούν διαφορετικές λειτουργίες. Οι ετικέτες χρησιμοποιούνται για τη σήμανση της έναρξης ενός στοιχείου HTML και συνήθως περικλείονται σε γωνιακές αγκύλες. Ένα παράδειγμα ετικέτας είναι: <h1>(Επικεφαλίδα μεγέθους 1). Επίσης, όταν γίνεται χρήση πολλαπλών ετικετών, πρέπει να κλείνουν με τη σειρά με την οποία ανοίχθηκαν.

Τα χαρακτηριστικά περιέχουν πρόσθετες πληροφορίες. Έχουν τη μορφή μιας ετικέτας ανοίγματος και πρόσθετες πληροφορίες τοποθετούνται μέσα. Για παράδειγμα: *<img src = "map.jpg" alt = "A map image.">*

Τα προγράμματα πλοήγησης, διαβάζοντας το συντακτικό της HTML το μεταφράζουν στις ιστοσελίδες χωρίς να εμφανίζει τις ετικέτες της, ο χρήστης βλέπει μόνο το περιεχόμενο των ετικετών και των χαρακτηριστικών.

Συνοψίζοντας, η HTML δίνει στον σχεδιαστή τη δυνατότητα να ανεβάσει στο διαδίκτυο σελίδες με κείμενο, πίνακες, εικόνες, υπερκείμενο, συνδέσμους, ανακατεύθυνση σε άλλες ιστοσελίδες με το πάτημα κάποιου κουμπιού, φόρμες συμπλήρωσης κλπ. Πλέον χρησιμοποιείται η HTML5. Τα ονόματα αρχείων HTML χρησιμοποιούν την επέκταση .html. Οι ετικέτες και τα χαρακτηριστικά της HTML μπορούν να μορφοποιηθούν με την CSS.

2.2.2 CSS (Cascading Style Sheets)

Η CSS (Cascading Style Sheets) είναι γλώσσα η οποία μορφοποιεί και ασχολείται με την παρουσίαση της ιστοσελίδας ή της διαδικτυακής εφαρμογής. Η CSS μπορεί να γραφεί μέσα στις ετικέτες HTML για να μορφοποίηση την συγκεκριμένη ετικέτα, μπορεί να γραφεί μέσα στο αρχείο HTML σε μια <style> </style> ετικέτα, αλλά μπορεί και να είναι ανεξάρτητη της HTML, σε ένα ξεχωριστό αρχείο, με επέκταση .css. Ο τελευταίος τρόπος είναι ο καλύτερος και προτιμάται διότι διαχωρίζει το περιεχόμενο με την παρουσίαση. Με αυτόν τον διαχωρισμό, βελτιώνεται η προσβασιμότητα περιεχομένου, η ευελιξία, η ευαναγνωσιμότητα και ο έλεγχος για τυχόν λάθη. Επίσης, ο διαχωρισμός επιτρέπει σε πολλές ιστοσελίδες να μοιραστούν μορφοποίηση (π.χ. αν ο σχεδιαστής επιθυμεί να έχει την ίδια μορφοποίηση στην αρχική σελίδα και στη σελίδα επικοινωνίας), πράγμα που μειώνει την πολυπλοκότητα και την επανάληψη του δομικού περιεχομένου καθώς βελτιώνει την ταχύτητα φόρτωσης των σελίδων.

Ο διαχωρισμός μορφοποίησης και περιεχομένου καθιστά επίσης εφικτή την παρουσίαση της ίδιας σελίδας σε διαφορετικά στυλ για διαφορετικές μεθόδους απόδοσης (rendering), όπως για παράδειγμα στην οθόνη ενός επιτραπέζιου υπολογιστή, στην εκτύπωση, με φωνή (μέσω προγράμματος περιήγησης που βασίζεται σε ομιλία ή πρόγραμμα ανάγνωσης οθόνης) και σε γραφή Μπράιγ αν η συσκευή είναι κατάλληλη (απτικές συσκευές: Braille-based tactile devices). Τέλος, η CSS έχει δυνατότητα για εναλλακτική μορφοποίηση εάν το περιεχόμενο εμφανίζεται σε μια κινητή συσκευή. Ονομάζεται responsive design (ανταποκρίσιμος σχεδιασμός) και ο κώδικας προσαρμόζεται ώστε η εμφάνιση του περιεχομένου να αλλάζει σύμφωνα με το μέγεθος της οθόνης για την βέλτιστη παρουσίαση του.

Το συντακτικό της CSS είναι πάρα πολύ απλό. Μια εντολή ξεκινάει προσδιορίζοντας ποιο αντικείμενο θέλει ο προγραμματιστής να μορφοποιήσει: *h2 { . . }* όλα τα στοιχεία ενός συγκεκριμένου τύπου, π.χ. οι κεφαλίδες δεύτερου επιπέδου h2, *.homepage-h2 { .. }* για τα στοιχεία που προσδιορίζονται από μια κλάση πριν το όνομα της κλάσης τους μπαίνει μια τελεία, *#homepage-h2 { .. }* για τα στοιχεία που έχουν προσδιοριστεί με μια ταυτότητα, id, η οποία είναι μοναδική μέσα στον κώδικα, έχουν μπροστά από το όνομα τους το σύμβολο #. Μέσα στις αγκύλες περιλαμβάνονται όλες οι εντολές μορφοποίησης.

2.2.3 Bootstrap

Η Bootstrap είναι μια «εργαλειοθήκη» ανοιχτού κώδικα η οποία περιέχει εκτεταμένα προκατασκευασμένα εξαρτήματα (πολλές φορές είναι και αποκριτικά) στα οποία είναι ήδη ενσωματωμένη η JavaScript, δηλαδή η λειτουργικότητα αυτών των εξαρτημάτων. Τα εξαρτήματα που διαθέτει η Bootstrap μπορούν να ενσωματωθούν με διάφορους τρόπους σε μια ιστοσελίδα. Η Bootstrap διαθέτει ορισμένα έτοιμα εξαρτήματα που έχουν ενσωματωμένη javascript και CSS και ο κώδικας είναι διαθέσιμος στην ιστοσελίδα της Bootstrap για όποιον επιθυμεί να τον συμπεριλάβει στο έργο του. Επίσης, διαθέτει κάποιες κλάσεις οι οποίες προσθέτουν μορφοποίηση και άρα εισάγοντας τη συγκεκριμένη κλάση στο αντικείμενο που επιθυμεί ο προγραμματιστής, προστίθεται αυτόματα και η μορφοποίηση που έχει ορίσει η Bootstrap. Για να λειτουργήσουν όλα τα παραπάνω απαιτείται η σύνδεση της Bootstrap μέσω του HTML αρχείου με την μορφοποίηση, δηλαδή τα stylesheets της Bootstrap στο <head> καθώς και το script αρχείο που περιέχει την απαραίτητη JavaScript πριν το τέλος του </body>. Και τα δύο είναι διαθέσιμα στην επίσημη ιστοσελίδα της Bootstrap.

2.3 JAVASCRIPT

Η JavaScript είναι από τις πιο ισχυρές και ευέλικτες γλώσσες προγραμματισμού καθώς αποτελεί βάση για τη δυναμική συμπεριφορά των περισσότερων ιστοσελίδων. Η JavaScript είναι μια γλώσσα δέσμης ενεργειών ή προγραμματισμού που επιτρέπει την εφαρμογή σύνθετων λειτουργιών σε ιστοσελίδες, δηλαδή όταν μια ιστοσελίδα είναι περισσότερο διαδραστική από το να εμφανίζει στατικές πληροφορίες π.χ. έγκαιρες ενημερώσεις περιεχομένου, διαδραστικούς χάρτες, κινούμενα γραφικά, βίντεο κ.λπ. Αποτελεί το τρίτο κομμάτι των τυποποιημένων τεχνολογιών ιστού, με τα άλλα δύο να είναι οι HTML και CSS που αναφέρθηκαν παραπάνω. Η HTML και η CSS είναι γλώσσες που δίνουν δομή και μορφοποίηση σε ιστοσελίδες, η JavaScript δίνει στις ιστοσελίδες διαδραστικά στοιχεία που προσελκύουν τον χρήστη. Πάνω από το 97% των ιστότοπων τη χρησιμοποιούν από την πλευρά του πελάτη για τη συμπεριφορά ιστοσελίδων, πολύ συχνά ενσωματώνοντας «εξωτερικές» βιβλιοθήκες (third-party libraries). Η JavaScript διαθέτει επίσης διάφορες διεπαφές προγραμματισμού εφαρμογών (API) για εργασία με κείμενο, ημερομηνίες, κανονικές εκφράσεις, τυπικές δομές δεδομένων και το Document Object Model (DOM).

Πιο συγκεκριμένα, το API DOM επιτρέπει τον χειρισμό HTML και CSS, δημιουργώντας, καταργώντας και αλλάζοντας HTML, εφαρμόζοντας δυναμικά νέες μορφοποιήσεις σε μια ιστοσελίδα ή εφαρμογή κ.λπ. Για παράδειγμα, όπως θα αναλυθεί στο κεφάλαιο της υλοποίησης, το DOM χρησιμοποιήθηκε για την υλοποίηση της διαδραστικότητας των κουμπιών της αρχικής σελίδας, αλλάζοντας κλάση σε HTML στοιχεία μέσω της JavaScript.

Όλα ξεκίνησαν όταν ο Marc Andreessen, ο ιδρυτής της Netscape Communications, οραματίστηκε ένα πιο δυναμικό διαδίκτυο όπου θα μπορούσε να συμπεριλάβει κινούμενα σχέδια, μικρούς αυτοματισμούς και ο χρήστης να έχει τη δυνατότητα να αλληλεπιδράσει με αυτό. Για αυτόν τον λόγο προσέλαβαν τον Brendan Eich, ο οποίος έπρεπε σε πολύ μικρό χρονικό διάστημα να δημιουργήσει μια γλώσσα που θα μπορούσε να προσδώσει όλα τα προαναφερθέντα χαρακτηριστικά στις ιστοσελίδες. Η γλώσσα αρχικά ονομάστηκε Mocha, όμως πολύ σύντομα - όταν κυκλοφόρησε πρώτη φορά με το Netscape Navigator 2.0 - μετονομάστηκε σε LiveScript και μέχρι το Δεκέμβρη του 1995 η γλώσσα είχε πάρει το τελικό της όνομα, δηλαδή Javascript. Σκοπός ήταν να παρουσιαστεί η JavaScript σαν γλώσσα δέσμης ενεργειών για να εκτελεί τις διεργασίες από τη μεριά του πελάτη στον πλοηγό ενώ η Java, η οποία χρησιμοποιούνταν ήδη, θα προωθούνταν σαν το «μεγαλύτερο», επαγγελματικό εργαλείο για την ανάπτυξη πλούσιων συστατικών ιστού. Εδώ αξίζει να αναφέρουμε ότι η JavaScript μπορεί να μοιράζεται κάποια στοιχεία με την Java, όπως για παράδειγμα, το όνομα και κάποιες ομοιότητες στο συντακτικό, αλλά στην πραγματικότητα οι δύο γλώσσες έχουν πολύ μεγάλη διαφορά στον σχεδιασμό τους και σε καμία περίπτωση δεν πρέπει να συγχέονται.

Στη συνέχεια η Microsoft «πήρε» την JavaScript, τη μετονόμασε σε JScript και την κυκλοφόρησε με τον Internet Explorer 3.0 το 1996. Τότε ήταν που ο Eich ξανάγραψε το μεγαλύτερο κομμάτι της Mocha σε μια πιο καθαρή εφαρμογή και το ονόμασε SpiderMonkey, τον «παππού» του Firefox. Το Νοέμβριο του 1996 η Netscape κατέθεσε την JavaScript στην ECMA ως σημείο εκκίνησης για να υπάρχουν κάποιες προδιαγραφές στις οποίες θα μπορούσαν να συμμορφωθούν όλοι οι προμηθευτές προγραμμάτων περιήγησης. Αυτό οδήγησε στην πρώτη επίσημη κυκλοφορία των πρώτων προδιαγραφών γλώσσας ECMAScript τον Ιούνιο του 1997. Για αρκετά χρόνια η JScript και η SpiderMonkey ήταν οι πρώτες και μόνες JavaScript μηχανές.

Από τότε πολλά άλλαξαν, αλλά το μόνο σίγουρο είναι ότι η JavaScript καθιερώθηκε ως μια από τις πιο σημαντικές γλώσσες που χρησιμοποιούνται ευρέως από τις διαδικτυακές ιστοσελίδες και εφαρμογές για την πλευρά του πελάτη (client side) και έχει εμπλουτιστεί με πάρα πολλούς σκελετούς (frameworks) και βιβλιοθήκες (libraries).

2.4 JSON

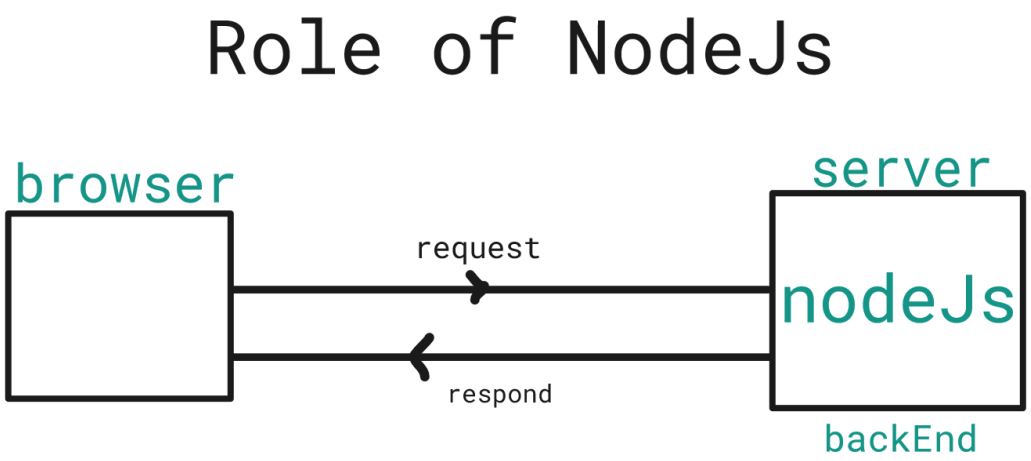
Το JSON είναι μια ανοικτή τυπική μορφή αρχείου και μορφή ανταλλαγής δεδομένων που χρησιμοποιεί αναγνώσιμο από τον άνθρωπο κείμενο για την αποθήκευση και τη μετάδοση δεδομένων που αποτελούνται από ζεύγη χαρακτηριστικών-τιμών και πίνακες (ή άλλες σειριοποιήσιμες τιμές). Είναι μια κοινή μορφή δεδομένων με ένα ευρύ φάσμα λειτουργιών στην ανταλλαγή δεδομένων, συμπεριλαμβανομένης της επικοινωνίας εφαρμογών Ιστού με διακομιστές. Το JSON είναι μια μορφή δεδομένων ανεξάρτητη από τη γλώσσα. Προήλθε από την JavaScript, αλλά πολλές σύγχρονες γλώσσες προγραμματισμού περιλαμβάνουν κώδικα για τη δημιουργία και ανάλυση δεδομένων μορφής JSON. Η βάση δεδομένων που χρησιμοποιήθηκε στην παρούσα εργασία αποθηκεύει τα δεδομένα σαν αντικείμενα JSON. Τα ονόματα αρχείων JSON χρησιμοποιούν την επέκταση .json.

2.4 NODEJS

Το nodejs δημιουργήθηκε το 2009 από τον Ryan Dahl. Ο Dahl επέκρινε τις περιορισμένες δυνατότητες του πιο δημοφιλούς ιστού το 2009, του Apache HTTP Server, να χειρίζεται πολλές ταυτόχρονες συνδέσεις και τον πιο συνηθισμένο τρόπο δημιουργίας κώδικα (διαδοχικός προγραμματισμός), όταν ο κώδικας είτε μπλοκάρει ολόκληρη την διαδικασία είτε συνεπαγόμενες πολλαπλές στοίβες εκτέλεσης στην περίπτωση ταυτόχρονων συνδέσεων.

Το Nodejs επιτρέπει τη δημιουργία διακομιστών ιστού και εργαλείων δικτύωσης χρησιμοποιώντας JavaScript και μια συλλογή «ενοτήτων» (“modules”) που χειρίζονται διάφορες βασικές λειτουργίες. Οι «ενότητες» παρέχονται για σύστημα αρχείων I/O, για δικτύωση (DNS, HTTP, TCP, TLS/SSL ή UDP), δυαδικά δεδομένα (buffers), λειτουργίες κρυπτογραφίας, ροές δεδομένων και άλλες βασικές λειτουργίες. Οι μονάδες του Node.js χρησιμοποιούν ένα API

σχεδιασμένο να μειώνει την πολυπλοκότητα της γραφής εφαρμογών διακομιστή (server applications). Πιο συγκεκριμένα μια εφαρμογή Node.js εκτελείται σε μία μόνο διαδικασία, χωρίς να δημιουργεί νέο νήμα για κάθε αίτημα. Το Node.js παρέχει ένα σύνολο ασύγχρονων διαδικασιών εισόδου/εξόδου στη βιβλιοθήκη του που εμποδίζουν τον αποκλεισμό κώδικα JavaScript καθώς και γενικότερα οι βιβλιοθήκες στο Node.js γράφονται χρησιμοποιώντας πρότυπα μη αποκλεισμού, καθιστώντας τη συμπεριφορά αποκλεισμού εξαίρεση και όχι κανόνα. Το σύνολο των ασύγχρονων διαδικασιών, δηλαδή η ασύγχρονη επικοινωνία μεταξύ των υπολογιστικών πόρων επιτυγχάνεται με τη χρήση συμβάντων που προσφέρει η JavaScript και ονομάζονται callbacks. Κάθε φορά που συνδέεται ένας χρήστης η επανάκληση (callback) ενεργοποιείται, εάν δεν υπάρχει κάποια διεργασία να γίνει το Node.js μπαίνει σε «κατάσταση ύπνου».



Εικόνα 2.1: Ο ρόλος του NodeJs

Αυτό συμβαίνει γιατί όταν το Node.js εκτελεί μια λειτουργία εισόδου/εξόδου, όπως η ανάγνωση από το δίκτυο, η πρόσβαση σε μια βάση δεδομένων ή στο σύστημα αρχείων, αντί να μπλοκάρει το νήμα και να σπαταλά τους κύκλους της CPU σε αναμονή, το Node.js θα συνεχίσει τις λειτουργίες όταν επιστρέψει η απάντηση. Αυτό επιτρέπει στο Node.js να χειρίζεται χιλιάδες ταυτόχρονες συνδέσεις με έναν μόνο διακομιστή χωρίς να εισάγει το βάρος της διαχείρισης του νήματος συγχρονισμού, το οποίο θα μπορούσε να είναι σημαντική πηγή σφαλμάτων. Επιπλέον, το Node.js έχει ένα μοναδικό πλεονέκτημα επειδή εκατομμύρια προγραμματιστές που γράφουν

JavaScript για το πρόγραμμα περιήγησης είναι πλέον σε θέση να γράψουν τον κωδικό από την πλευρά του διακομιστή και τον κώδικα από την πλευρά του πελάτη χωρίς να χρειάζεται να μάθουν μια εντελώς διαφορετική γλώσσα. Για αυτό ακριβώς τον λόγο προτιμήθηκε και για τη συγκεκριμένη εργασία.

Εντούτοις, αξίζει να σημειωθεί ότι ενώ η JavaScript είναι η μόνη γλώσσα που υποστηρίζει εγγενώς το Node.js, είναι διαθέσιμες πολλές γλώσσες compile-to-JS. Αυτό σημαίνει ότι μπορούν να χρησιμοποιηθούν και άλλες γλώσσες οι οποίες μετέπειτα μεταφράζονται (compile) σε JavaScript και έτσι το node.js στο τέλος «διαβάζει» JavaScript.

Τον Ιανουάριο του 2010 εισήχθη ένας διαχειριστής πακέτων για το περιβάλλον του nodejs, το npm. Το npm (Node Package Manager), εγκαθίσταται μαζί με το Node.js και είναι το εργαλείο με το οποίο εγκαθιστούμε, διαγράφουμε ή αναβαθμίζουμε εξωτερικά (third-party) πακέτα του Node. Διαθέτει ιστοσελίδα στην οποία είναι καταχωρημένα όλα τα πακέτα που διαθέτει. Μέσα στον φάκελο που έχει τοποθετήσει τα αρχεία της εφαρμογής του ο προγραμματιστής, κατεβάζει και τα πακέτα που χρειάζονται για την ανάπτυξη της είτε μέσω του φλοιού μέσα στον επεξεργαστή κειμένου της επιλογής του (Atom, Visual Studio) είτε μέσω του φλοιού εκτός του επεξεργαστή κειμένου (CMD, terminal, hyperterminal). Οι ενότητες (modules) είναι κομμάτια ενθυλακωμένου κώδικα που επικοινωνούν με μια εξωτερική εφαρμογή με βάση τη σχετική λειτουργικότητά τους. Οι ενότητες μπορούν να είναι ένα μόνο αρχείο ή μια συλλογή πολλαπλών αρχείων/φακέλων. Ο λόγος που οι προγραμματιστές εξαρτώνται σε μεγάλο βαθμό από τις ενότητες είναι λόγω της επαναχρησιμοποίησής τους καθώς και της δυνατότητας να διασπάσουν ένα πολύπλοκο κομμάτι κώδικα σε διαχειρίσιμα κομμάτια. Οι ενότητες είναι τριών τύπων: Βασικές ενότητες, τοπικές ενότητες, ενότητες τρίτων (third-party).

Βασικές ενότητες: Το Node.js διαθέτει πολλές ενσωματωμένες ενότητες που αποτελούν μέρος της πλατφόρμας και συνοδεύεται από εγκατάσταση του Node.js. Αυτές οι ενότητες μπορούν να φορτωθούν στο πρόγραμμα χρησιμοποιώντας τη συνάρτηση απαιτήσεων:

```
const module = requie ('module_name');
```

Η συνάρτηση require() θα επιστρέψει έναν τύπο JavaScript ανάλογα και με το τι επιστρέφει η συγκεκριμένη ενότητα (module).

Τοπικές ενότητες: Σε αντίθεση με τις ενσωματωμένες και τις εξωτερικές ενότητες, οι τοπικές ενότητες δημιουργούνται τοπικά στην εφαρμογή Node.js.

Εξωτερικές ενότητες (third-party modules): Οι ενότητες τρίτων είναι ενότητες που είναι διαθέσιμες στο διαδίκτυο και εγκαθίστανται χρησιμοποιώντας τον Διαχειριστή πακέτων κόμβων (npm). Αυτές οι ενότητες μπορούν να εγκατασταθούν στον φάκελο του έργου ή σε «συνολικό» επίπεδο (δηλαδή είναι εγκατεστημένες πλέον στο σύστημα και μπορούν να χρησιμοποιηθούν ξανά χωρίς εγκατάσταση εκ νέου). Αυτό γίνεται εισάγοντας στον φλοιό την εντολή:

>npm install package_name

ή την εντολή:

>npm install package_name -g ή *>npm install package_name --global*

Και έπειτα «απαιτούμε» το πακέτο που εγκαταστάθηκε με την συνάρτηση JavaScript:

```
const express = require("package_name");
```

Όλες οι παραπάνω ενότητες βρίσκονται σε έναν φάκελο που δημιουργείται και ονομάζεται node_modules.

Πέρα από τον φάκελο node_modules, όταν ο προγραμματιστής εγκαθιστά οποιοδήποτε πακέτο στο έργο του, θα εγκατασταθεί η τελευταία έκδοση αυτού του πακέτου και θα σωθεί η εξάρτηση στο package.json. Το package-lock.json δημιουργείται για το κλείδωμα της εξάρτησης από την εγκατεστημένη έκδοση. Για να αποφευχθούν διαφορές στις εγκατεστημένες εξαρτήσεις από διαφορετικό περιβάλλον, θα πρέπει να χρησιμοποιείται το αρχείο package-lock.json για να εγκατασταθούν οι εξαρτήσεις. Στην ιδανική περίπτωση, αυτό το αρχείο θα πρέπει να βρίσκεται στον έλεγχο της πηγής (source control) με το αρχείο package.json, οπότε όταν ο προγραμματιστής ή οποιοσδήποτε άλλος χρήστης θα κλωνοποιήσει το έργο και θα εκτελέσει την εντολή «npm i», θα εγκαταστήσει την ίδια ακριβώς έκδοση που είναι αποθηκευμένη στο αρχείο package-lock.jason και θα είναι σε θέση να παράγει τα ίδια αποτελέσματα που είχε ο προγραμματιστής όταν ανέπτυξε τη συγκεκριμένη εφαρμογή με το συγκεκριμένο πακέτο.

2.5 EXPRESS & ΔΟΜΗ ΑΡΧΕΙΩΝ

2.5.1 EXPRESS.JS

Το express.js είναι ένας ανοιχτού κώδικα σκελετός/δομή (framework) του node.js γραμμένος σε javascript ο οποίος λειτουργεί στο παρασκήνιο (back-end) και σχεδιάστηκε για την ανάπτυξη διαδικτυακών εφαρμογών και APIs. Το Express αποτελεί κομμάτι του λεγόμενου MEAN stack, που περιλαμβάνει επίσης τη MongoDB για βάση δεδομένων, την AngularJS για το εμπρόσθιο κομμάτι που βλέπουν οι χρήστες (front-end) και το NodeJs.

Το express είναι σχετικά απλό μα ισχυρό εργαλείο για την δημιουργία ενός διακομιστή. Έχει μινιμαλιστική και «αόρατη» προσέγγιση που επικεντρώνεται στα βασικά χαρακτηριστικά ενός διακομιστή και δεν παρεμποδίζει τις δυνατότητες του node.js. Το Express εγκαθίσταται από μέσω του τερματικού του node.js μέσω του πακέτου npm.

2.5.2 MVC file structure (δομή φακέλου)

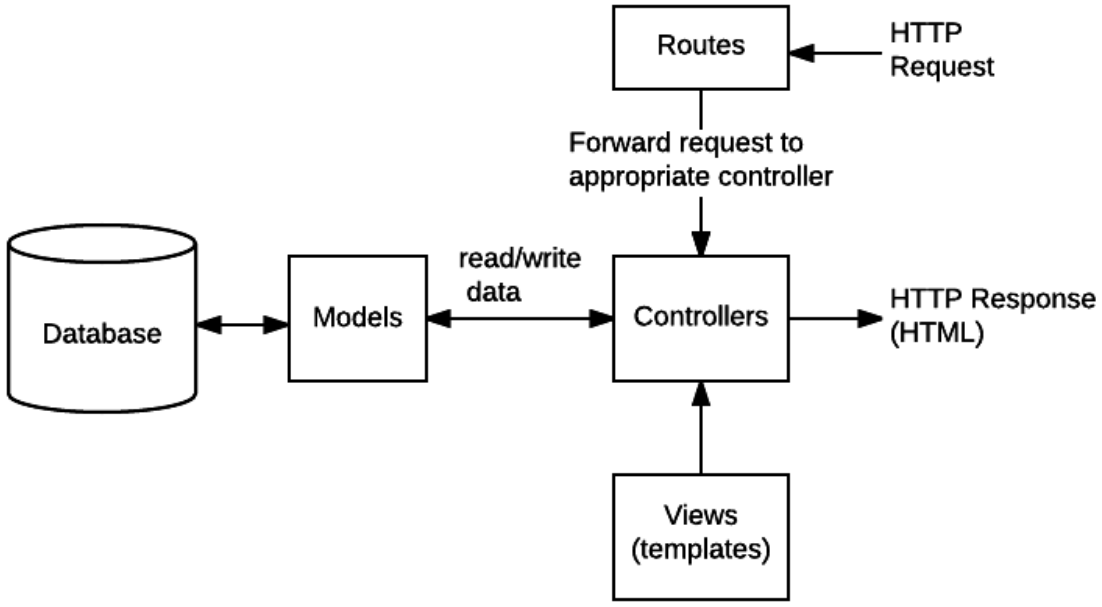
Το MVC (Model View Controller) είναι ένα αρχιτεκτονικό μοτίβο που χωρίζει μια εφαρμογή σε τρία κύρια λογικά στοιχεία: το μοντέλο (model), την προβολή (View) και τον ελεγκτή (Controller). Κάθε ένα από αυτά τα στοιχεία είναι κατασκευασμένο να χειρίζεται συγκεκριμένες πτυχές ανάπτυξης μιας εφαρμογής.

Model (Μοντέλο) είναι η διεπαφή βάσης δεδομένων που επιτρέπει την αλληλεπίδραση με το API της βάσης δεδομένων και τη δημιουργία διαφορετικών σχημάτων οντοτήτων της εφαρμογής στη βάση δεδομένων (MySQL, MongoDB). Καλείται από τον ελεγκτή ανάλογα με το αίτημα του πελάτη, εάν χρειάζεται αποθηκευμένα δεδομένα τότε ο ελεγκτής θα ζητήσει το μοντέλο διεπαφής για την παροχή των απαραίτητων δεδομένων.
Αυτός ο φάκελος περιέχει τη λειτουργικότητα και τις λογικές που σχετίζονται με τη βάση δεδομένων, όπως εισαγωγή, λήψη, ενημέρωση, διαγραφή ερωτημάτων. Ακόμη παίρνει το αίτημα ερωτήματος από τον ελεγκτή και στέλνει την απάντηση πίσω στον ελεγκτή.

View (Προβολή) είναι αυτό που μεταγλωττίζεται και μετατρέπεται σε απλό HTML και αυτό που ο πελάτης στις περισσότερες περιπτώσεις πρόκειται να πάρει πίσω ως απάντηση σε αυτό που ζήτησε (για παράδειγμα: για να δει κάποιος τα στοιχεία του προφίλ του). Η προβολή πρέπει να

χρησιμοποιεί μια μηχανή προτύπων για την διαδικασία μετατροπής όταν ο ελεγκτής το τροφοδοτεί με τα απαραίτητα δεδομένα (δεδομένα από την βάση δεδομένων και τον πελάτη) και αποδίδει και μετατρέπει τα πάντα σε απλό HTML που θα μπορούσε να κατανοήσει και να εμφανίσει το πρόγραμμα περιήγησης.

Controller (Ελεγκτής) είναι εκείνο που φροντίζει για την επεξεργασία αιτήματος πελάτη, χειρίζεται το αίτημα HTTP και επιστρέφει μια απάντηση, η απάντηση θα μπορούσε να είναι είτε JSON εάν καλείται ένα τελικό σημείο API είτε κανονική ιστοσελίδα HTML.

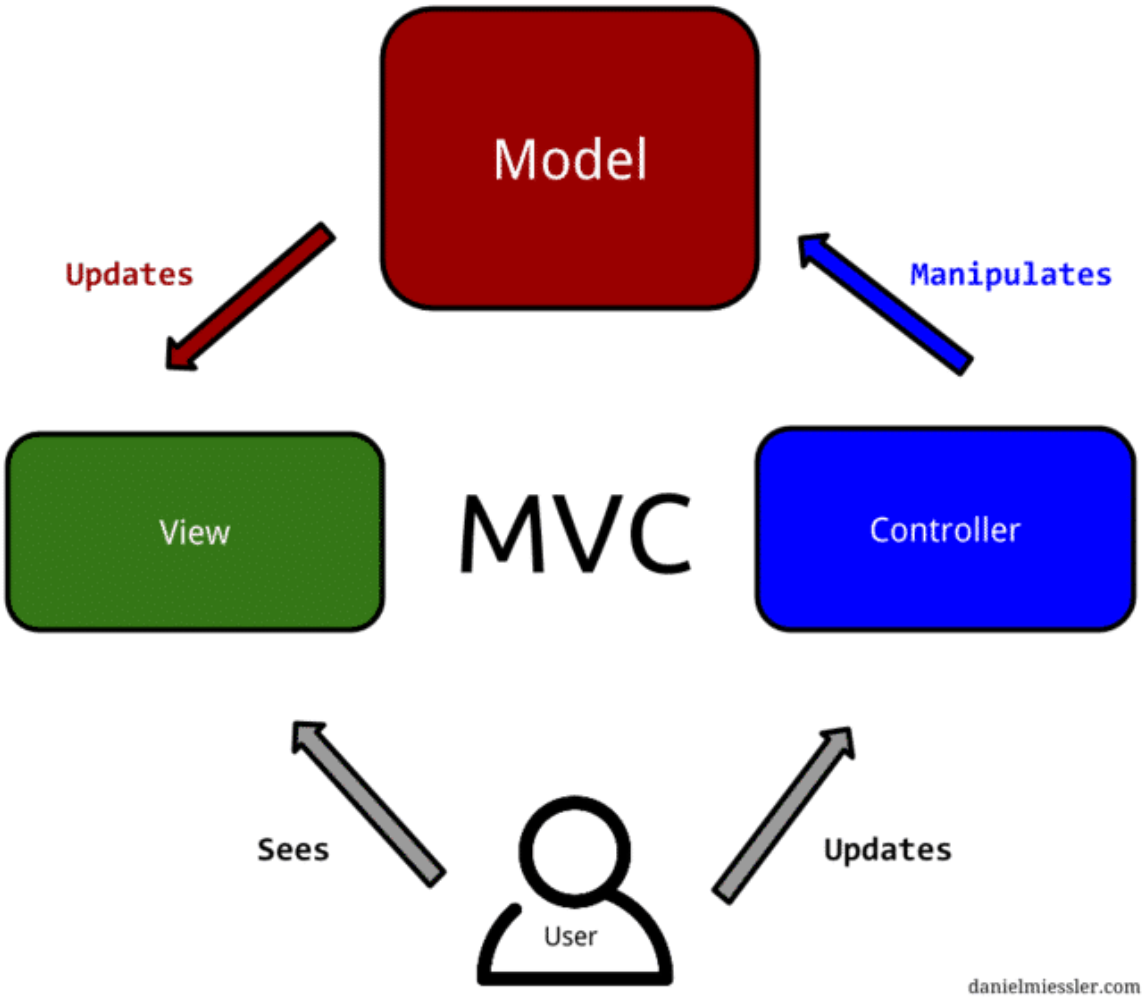


Εικόνα 2.2: Δομή MVC[2]

Έτσι, τα τρία διαφορετικά συστατικά αλληλεπιδρούν μεταξύ τους άρτια για να εξασφαλίσουν αξιόπιστη και ευέλικτη επεξεργασία κάθε αιτήματος του πελάτη, είτε πρόκειται για απλές εργασίες όπως η πρόσβαση στην αρχική σελίδα είτε για πιο περίπλοκες όπως η ενημέρωση των

[2] Πηγή εικόνας: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes

δεδομένων του προφίλ του.



Εικόνα 2.3 Δομή MVC από την πλευρά του χρήστη[3]

Τέλος, για να υπάρχει επικοινωνία με τα τελικά σημεία πρέπει να οριστούν οι διαδρομές με αντίστοιχα αιτήματα. Τα στοιχεία που είναι υπεύθυνα για αυτή την επικοινωνία βρίσκονται στον φάκελο routes (διαδρομές).

2.6 MINIFYING FILES

Για να μπορέσει ένας χρήστης να δει την ιστοσελίδα ή την εφαρμογή στον υπολογιστή του (κινητό κλπ) πληκτρολογεί τη διεύθυνση της στο πρόγραμμα πλοήγησης που διαθέτει. Μέσω του DNS (διακομιστής τομέα ονομάτων) ο διακομιστής μετατρέπει τη διεύθυνση της

[3] Πηγή εικόνας: https://medium.com/@ipenywis/what-is-the-mvc-creating-a-node-js-express-mvc-application-da10625a4eda

ιστοσελίδας/εφαρμογής στη διεύθυνση IP και έτσι απαντάει με τη συγκεκριμένη ιστοσελίδα/εφαρμογή.

Η διαδικασία του minifying ή αλλιώς της ελαχιστοποίησης αφαιρεί από τα CSS ή JS αρχεία, τους μη απαραίτητους χαρακτήρες στον κώδικα χωρίς να αλλάξει τη λειτουργικότητα του, ώστε να μειωθεί το μέγεθος του αρχείου. Όταν δηλαδή, υπάρχουν ελαχιστοποιημένα αρχεία μέσα στον φάκελο της εφαρμογής και ένας χρήστης κάνει αίτηση για την ιστοσελίδα ή την εφαρμογή θα αποσταλούν αυτόματα οι ελαχιστοποιημένες εκδόσεις με αποτέλεσμα γρηγορότερη φόρτωση και χαμηλότερο κόστος εύρος ζώνης.
Η διαδικασία αυτή μπορεί να γίνει και μέσω ιστοσελίδων όπως το https://cssminifier.com/ όπου καταχωρείς τον κώδικα του αρχικού CSS αρχείου και το μετατρέπει αυτόματα σε minified αρχείο είτε με τη βοήθεια διάφορων εργαλείων όπως το gulp.

2.7 GULP

Το σύγχρονο web development αποτελείται από πολλές επαναλαμβανόμενες διεργασίες όπως το να «τρέχει» έναν τοπικό διακομιστή, να ελαχιστοποιεί κώδικα CSS ή JavaScript, να βελτιστοποιεί εικόνες, να επεξεργάζεται τον κώδικα κλπ. Για αυτό τον λόγο υπάρχουν κάποια εργαλεία τα οποία αυτοματοποιούν τέτοιου είδους διεργασίες, όπως για παράδειγμα το gulp το οποίο είναι ένα ανοιχτού κώδικα εργαλείο ροής εργασιών.

Το gulp λοιπόν είναι ένα εργαλείο αυτοματοποίησης, το οποίο διαφέρει από άλλα εργαλεία επειδή χρησιμοποιεί Node streams (ροές κόμβου), δηλαδή μεταφέρει την έξοδο από τη μία διεργασία ως είσοδο στην επόμενη. Χρειάζεται να διαβάσει ένα αρχείο μόνο μια φορά, στη συνέχεια να το επεξεργαστεί μέσω πολλαπλών διεργασιών και, τέλος, να γράψει το αρχείο εξόδου. Αυτό έχει ως αποτέλεσμα ταχύτερες κατασκευές επειδή δεν υπάρχει ανάγκη δημιουργίας και ανάγνωσης ενδιάμεσων αρχείων στον σκληρό δίσκο.

Το gulp απαιτεί το NodeJs και το διαχειριστή πακέτων του, npm ώστε να εγκατασταθεί μαζί με τα πρόσθετα του (plugins).

2.8 LOCALHOST

Το localhost είναι σαν ένα όνομα τομέα, το οποίο αντιστοιχεί σε IP διεύθυνση ίδια με του υπολογιστή, δηλαδή το όνομα localhost:(αριθμός θύρας) αντιστοιχεί στην IP του ίδιου του υπολογιστή που έχει ανεβάσει τον διακομιστή τοπικά. Όταν γίνεται σύνδεση με τον localhost τομέα μέσω ενός προγράμματος πλοήγησης, το πρόγραμμα πλοήγησης ουσιαστικά συνδέεται με τον ίδιο τον υπολογιστή ο οποίος λειτουργεί σαν οικοδεσπότης (host).

Θύρες:

Οι αριθμοί των θυρών λειτουργούν κατά κάποιον τρόπο σαν «πόρτες» σε έναν υπολογιστή. Ο αριθμός θύρας αντιπροσωπεύει μια συγκεκριμένη πύλη δικτύου (gateway), κανάλι (channel) δικτύου ή θύρα (port) όπου ένα συγκεκριμένο κομμάτι λογισμικού (software) ή διακομιστή κυκλοφορεί. Ένας υπολογιστής διαθέτει πληθώρα διαφορετικών λογισμικών που συνδέονται στο διαδίκτυο και αποστέλλουν και λαμβάνουν δεδομένα, όπως για παράδειγμα το skype, discord, το ηλεκτρονικό ταχυδρομείο. Αυτή η διαδικασία γίνεται για κάθε λογισμικό μέσω διαφορετικών θυρών έτσι ώστε οι πληροφορίες και τα δεδομένα να υπάρχουν και να «κυκλοφορούν» χώρια μεταξύ τους. Έτσι, ο υπολογιστής και ο διακομιστής χρειάζεται αριθμό θύρας για να μπορεί μέσω αυτής να «επικοινωνεί». Η πιο συχνή θύρα είναι η 3000.

2.9 EJS

Το EJS είναι ένα πρότυπο σύστημα. Ο προγραμματιστής ιστού μπορεί να ορίσει τις σελίδες HTML σε σύνταξη EJS και να καθορίσει πού θα πάνε διάφορα δεδομένα στη σελίδα. Στη συνέχεια, η εφαρμογή του συνδυάζει δεδομένα με το πρότυπο και «αποδίδει» μια πλήρη σελίδα HTML όπου το EJS λαμβάνει τα δεδομένα του και τα εισάγει στην ιστοσελίδα σύμφωνα με τον τρόπο που έχει ορίσει το πρότυπο. Για παράδειγμα, θα μπορούσε να υπάρχει ένας πίνακας δυναμικών δεδομένων από μια βάση δεδομένων και το EJS να δημιουργεί τον πίνακα δεδομένων σύμφωνα με τους κανόνες εμφάνισης. Το EJS είναι συμβατό με το Express για χρήση στο παρασκήνιο καθώς συνδέεται με την αρχιτεκτονική View engine που παρέχει το Express και επιτρέπει στον προγραμματιστή να αποδώσει ιστοσελίδες στον πελάτη με την εντολή res.render () στο Express.

Το EJS είναι ένα εργαλείο για τη δημιουργία ιστοσελίδων που μπορεί να περιλαμβάνει δυναμικά δεδομένα, κώδικα JavaScript(μέσα στο ίδιο το αρχείο, χωρίς script) και μπορεί να διαμοιρασθεί πρότυπα κομμάτια με άλλες ιστοσελίδες (όπως κοινές κεφαλίδες/υποσέλιδα). Για παράδειγμα, στην συγκεκριμένη εφαρμογή η μπάρα πλοήγησης αποτελεί ένα ξεχωριστό αρχείο ejs και με αυτό τον τρόπο δεν υπάρχει επανάληψη στον κώδικα της κάθε επιμέρους ιστοσελίδας. Η ίδια τακτική ακολουθήθηκε και για τους συνδέσμους Bootstrap που πρέπει να μπαίνουν πριν από το </body>, καθώς και για το κομμάτι του <head>. Έτσι ο κώδικας γίνεται πιο ευανάγνωστος και καθαρός και μπορεί ο προγραμματιστής ιστού να καταλάβει πιο εύκολα ποιο κομμάτι δεν λειτουργεί σε περίπτωση σφάλματος και να ανατρέξει σε αυτό.

Τέλος, το EJS δεν είναι ένα πλαίσιο του εμπρόσθιου τμήματος (front-end). Ενώ μπορεί να χρησιμοποιηθεί από την πλευρά-του-πελάτη με την Javascript (client-side JavaScript) για τη δημιουργία από την πλευρά του πελάτη (client-side) HTML, χρησιμοποιείται πιο συχνά από το παρασκήνιο (back-end) για τη δημιουργία ιστοσελίδων σε απάντηση κάποιου αιτήματος URL.

2.10 LEAFLET MAP

Το Leaflet είναι μια από τις καλύτερες βιβλιοθήκες ανοιχτού κώδικα JavaScript για διαδραστικούς χάρτες φιλικούς προς κινητά. Το Leaflet επιτρέπει στους προγραμματιστές χωρίς υπόβαθρο GIS να εμφανίζουν πολύ εύκολα χάρτες ιστού με «πλακίδια» που φιλοξενούνται σε δημόσιο διακομιστή, με προαιρετικές επικαλύψεις «πλακιδίων». Μπορεί να φορτώσει δεδομένα χαρακτηριστικών από αρχεία GeoJSON, να μορφοποιήσει και να αποκτήσει διαδραστικά επίπεδα, όπως δείκτες με αναδυόμενα παράθυρα όταν ο χρήστης κάνει κλικ πάνω στον χάρτη. Χρησιμοποιείται ευρύτερα και έχει όλες τις δυνατότητες χαρτογράφησης που χρειάζονται οι περισσότεροι προγραμματιστές. Προτιμήθηκε διότι προσφέρει απλότητα, απόδοση και χρηστικότητα. Λειτουργεί αποτελεσματικά σε όλες τις μεγάλες πλατφόρμες για επιτραπέζιους υπολογιστές και φορητές συσκευές, μπορεί να επεκταθεί με πολλά πρόσθετα, έχει ένα όμορφο, εύχρηστο και καλά τεκμηριωμένο API και έναν απλό, αναγνώσιμο πηγαίο κώδικα. Η σύνδεση του χάρτη με την εφαρμογή γίνεται με έναν μοναδικό τρόπο. Η διαδικασία θα περιγραφεί στο κεφάλαιο της υλοποίησης της εφαρμογής, προκειμένου να αποφευχθεί η επανάληψη εντός της παρούσας εργασίας.

2.11 MONGODB – MONGOOSE

2.11.1 MongoDB

Η MongoDB είναι μια βάση δεδομένων που άρχισε να αναπτύσσεται από την εταιρεία λογισμικού 10gen το 2007 ως μέρος μιας πλατφόρμας που είχαν προγραμματίσει να βγάλουν ως προϊόν υπηρεσίας. Το 2009, η εταιρεία στράφηκε σε ένα μοντέλο ανάπτυξης ανοιχτού κώδικα, με την εταιρεία να προσφέρει εμπορική υποστήριξη και άλλες υπηρεσίες. Το 2013, η 10gen άλλαξε το όνομά της σε MongoDB Inc.

Στους υπολογιστές, η βάση δεδομένων είναι μια οργανωμένη συλλογή δεδομένων που αποθηκεύονται και προσπελάσσονται ηλεκτρονικά από ένα υπολογιστικό σύστημα. Όπου οι βάσεις δεδομένων είναι πιο περίπλοκες, συχνά αναπτύσσονται χρησιμοποιώντας επίσημες τεχνικές σχεδιασμού και μοντελοποίησης.

Το σύστημα διαχείρισης βάσεων δεδομένων (DBMS: DataBase Management System) είναι το λογισμικό που αλληλεπιδρά με τους τελικούς χρήστες, τις εφαρμογές και την ίδια τη βάση δεδομένων για τη συλλογή και ανάλυση των δεδομένων.

Οι βάσεις δεδομένων χωρίζονται σε δύο μεγάλες ομάδες, τις σχεσιακές και τις μη σχεσιακές βάσεις δεδομένων. Οι σχεσιακές έγιναν κυρίαρχες τη δεκαετία του 1980 και χρησιμοποιούν γραμμές και στήλες σε μια σειρά πινάκων για την οργάνωση των δεδομένων τους, με τη συντριπτική πλειοψηφία να χρησιμοποιεί SQL(Structured Query Language) για εγγραφή και αναζήτηση δεδομένων. Στη δεκαετία του 2000, οι μη σχεσιακές βάσεις δεδομένων άρχισαν να γίνονται δημοφιλείς, αναφερόμενες ως NoSQL(No Structured Query Language) επειδή χρησιμοποιούν διαφορετικό τρόπο δόμησης των δεδομένων τους.

Στον παρακάτω πίνακα, ο οποίος διαμορφώθηκε για λόγους ευαναγνωσίας, είναι οι διαφορές των δύο βάσεων:

SQL (Structured Query Language)	**NoSQL** (No structured Query Language)
Δομή (Structure)	
Αρκετά άκαμπτη (quite inflexible) Όταν δεν έχει δεδομένα σε μια καταχώρηση παίρνει σαν τιμή το null. Απαιτεί από την βάση δεδομένων να έχει συγκεκριμένη δομή.	**Ευέλικτη (flexible)** Μπορεί να έχει κενές καταχωρήσεις και δεν χρειάζεται να ακολουθεί μια συγκεκριμένη δομή η βάση δεδομένων.
Σχέσεις (Relationships)	
Σχεσιακή (Relational) Καλή επιλογή για να έχουν τα δεδομένα σχέσεις μεταξύ τους, για παράδειγμα για ένα e-shop.	**Μη σχετική (Non-relational)** Καλή επιλογή αν δεν ενδιαφέρουν τόσο οι σχέσεις μεταξύ των δεδομένων. Λειτουργεί περισσότερο με ενσωματωμένα έγγραφα και όχι με πίνακες.
Επεκτασιμότητα (Scalability)	
Κλιμακώνεται κάθετα. Είναι γενικά πολύ κοστοβόρο το να κλιμακώνεται μια βάση κάθετα. Για να σχηματιστεί μια εικόνα στο μυαλό του αναγνώστη, είναι σαν να χτίζεται ένας ουρανοξύστης.	Έχει καλύτερη επεκτασιμότητα, τα δεδομένα είναι περισσότερο κατανεμημένα. Η αντίστοιχη εικόνα, είναι σαν να χτίζονται μικρότερα αλλά πιο σταθερά κτίρια.

Από τα παραπάνω, γίνεται κατανοητό ότι δεν υπάρχει καλύτερη ή χειρότερη βάση δεδομένων συνολικά, αλλά υπάρχει καλύτερη επιλογή βάσης δεδομένων ανάλογα με το τι λειτουργικότητα θέλει ο προγραμματιστής να έχει η εφαρμογή του. Στην συγκεκριμένη περίπτωση επειδή δεν ενδιαφέρουν τόσο οι σχέσεις μεταξύ των δεδομένων, καθώς είναι απλές, έγινε η επιλογή της MongoDB που είναι NoSQL βάση δεδομένων.

Η MongoDB λειτουργεί με συλλογές και έγγραφα. Η συλλογή είναι μια ομάδα εγγράφων MongoDB. Μια συλλογή υπάρχει μόνο σε μια μεμονωμένη βάση δεδομένων. Οι συλλογές δεν

επιβάλλουν ένα σχήμα, αν ο προγραμματιστής όμως το επιθυμεί μπορεί να καθορίσει το σχήμα των συλλογών και των δεδομένων. Τα έγγραφα μιας συλλογής μπορούν να έχουν διαφορετικά πεδία. Συνήθως, όλα τα έγγραφα σε μια συλλογή έχουν παρόμοιο ή σχετικό σκοπό, όχι επειδή το απαιτεί η MongoDB, αλλά για ευκολία οργάνωσης των δεδομένων. Ένα έγγραφο είναι ένα σύνολο ζευγών κλειδιού-τιμής κωδικοποιημένο σε JSON. Τα έγγραφα έχουν δυναμικό σχήμα, το οποίο σημαίνει ότι τα έγγραφα της ίδιας συλλογής δεν χρειάζεται να έχουν το ίδιο σύνολο πεδίων ή την ίδια δομή και τα κοινά πεδία στα έγγραφα μιας συλλογής ενδέχεται να περιέχουν διαφορετικούς τύπους δεδομένων. Τα δεδομένα στο MongoDB όπως προαναφέρθηκε έχουν ευέλικτο σχήμα.

Το MongoDB παρέχει δύο τύπους μοντέλων δεδομένων: Μοντέλο ενσωματωμένων δεδομένων/εγγράφων και μοντέλο κανονικοποιημένων δεδομένων. Με βάση την απαίτηση της εφαρμογής, μπορεί να χρησιμοποιηθεί οποιοδήποτε από τα μοντέλα κατά την προετοιμασία του εγγράφου.

Μοντέλο ενσωματωμένων δεδομένων:

Σε αυτό το μοντέλο, μπορούν να ενσωματωθούν όλα τα σχετικά δεδομένα σε ένα μόνο έγγραφο, είναι επίσης γνωστό ως μοντέλο απο-κανονικοποιημένων δεδομένων. Επί παραδείγματι, στην περίπτωση που η βάση δεδομένων περιέχει τα στοιχεία των εργαζομένων σε μια εταιρεία, σε τρία διαφορετικά έγγραφα, πιο συγκεκριμένα: Προσωπικά_Στοιχεία, Στοιχεία_Επικοινωνίας και

Διευθύνσεις, τα τρία αυτά έγγραφα μπορούν να ενσωματωθούν σε ένα, όπως φαίνεται παρακάτω:

```
{
        _id: ,
        Emp_ID: "10025AE336"
        Personal_details:{
                First_Name: "Giannis",
                Last_Name: "Papadopoulos",
                Date_Of_Birth: "1995-09-26"
        },
        Contact: {
                e-mail: "giannispapad@gmail.com",
                phone: "6948022338"
        },
        Address: {
                city: "Athens",
                Area: "Pagrati",

        }
}
```

Κανονικοποιημένο μοντέλο δεδομένων:

Σε αυτό το μοντέλο, μπορούν να παραπεμφθούν τα δευτερεύοντα έγγραφα στο αρχικό έγγραφο, χρησιμοποιώντας αναφορές. Δηλαδή, για το συγκεκριμένο παράδειγμα ας θεωρηθεί ότι θα χρησιμοποιηθούν για τα δεδομένα τρεις συλλογές και τα στοιχεία από τις συλλογές που αναφέρονται στον κάθε εργαζόμενο θα έχουν σαν αναφορά την ταυτότητα που καταχωρεί ο προγραμματιστής ή η MongoDB αυτόματα. Για παράδειγμα, στο παρακάτω παράδειγμα στον εργαζόμενο με ταυτότητα _id: <ObjectId101> που αντιστοιχούν τα δεδομένα τα οποία έχουν για empDocID την _id του.

Employee:

```
{
        _id: <ObjectId101>,
        Emp_ID: "10025AE336"
}
```

Personal_details:

```
{
        _id: <ObjectId102>,
        empDocID: " ObjectId101",
        First_Name: "Giannis",
        Last_Name: "Papadopoulos",
        Date_Of_Birth: "1995-09-26"
}
```

Contact:

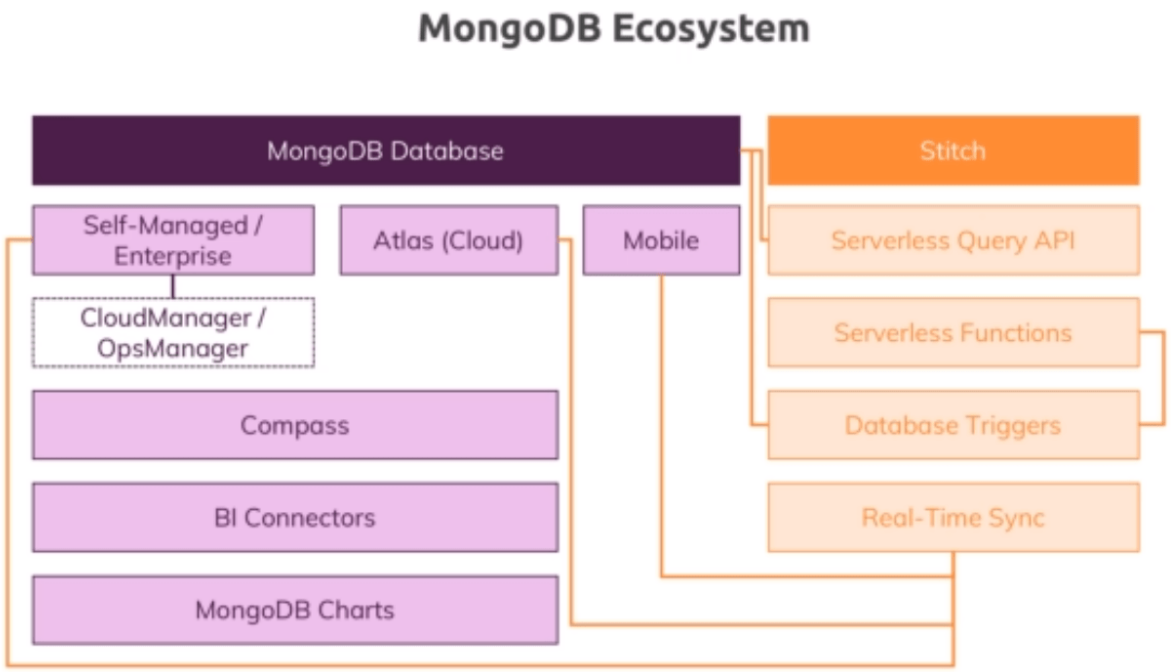
```
{
        _id: <ObjectId103>,
        empDocID: " ObjectId101",
        e-mail: "giannispapad@gmail.com",
        phone: "6948022338"
}
```

Address:

```
{
        _id: <ObjectId104>,
        empDocID: " ObjectId101",
        city: "Athens",
        Area: "Pagrati",

}
```

Η διαχείρηση των δεδομένων που περιέχει μια βάση δεδομένων μπορεί να γίνει με διάφορους τρόπους. Ο πρώτος είναι χειροκίνητα από τον φλοιό, ανοίγοντας δύο παράθυρα, στο ένα εισάγεται η εντολή:
>*mongod* που τρέχει την βάση και ανοίγοντας το δεύτερο παράθυρο εισάγεται η εντολή >*mongo* που συνδέει με τον φλοιό του MongoDB (Mongo Shell). Οι βασικές λειτουργίες είναι γνωστές και σαν CRUD (Create, Read, Update & Delete) και μπορούν όλες να δοθούν σαν εντολές από τον φλοιό. Μια σημαντική λειτουργία της MongoDB είναι το Projection (προβολή) που σημαίνει ότι μπορεί να γίνει επιλογή μόνο των απαραίτητων δεδομένων και όχι απαραίτητα επιλογή ολόκληρων των δεδομένων ενός εγγράφου.

Στο παρακάτω σχήμα φαίνεται το οικοσύστημα της MongoDB. Θα αναφερθούν ενδεικτικά κάποια από τα κομμάτια, διότι δεν χρησιμοποιήθηκαν για την παρούσα πτυχιακή εργασία.



Εικόνα 2.4 Οικοσύστημα MongoDB[4]

Αρχικά, υπάρχει η βάση δεδομένων MongoDB που είναι το βασικό χαρακτηριστικό. Υπάρχει μια αυτοδιαχειριζόμενη και μια επιχειρησιακή λύση, η πρώτη είναι δωρεάν και η δεύτερη είναι μια εμπορική έκδοση του MongoDB ως μέρος της συνδρομής MongoDB Enterprise Advanced. Στη συνέχεια, υπάρχουν κάποια εργαλεία που σχετίζονται με τη διαχείριση αυτής της βάσης δεδομένων. Το ένα είναι περισσότερο μια εργασία διαχειριστή βάσης δεδομένων συστήματος, το MongoDB Atlas, μια παγκόσμια υπηρεσία βάσης δεδομένων cloud. Είναι μια βάση δεδομένων ως υπηρεσία που επιτρέπει στον προγραμματιστή να εστιάσει περισσότερο στην ανάπτυξη των εφαρμογών αντί να αφιερώσει χρόνο στη διαχείριση των βάσεων δεδομένων καθώς οι διεργασίες που σχετίζονται με το σύστημα γίνονται αυτόματα.

[4] Πηγή εικόνας: https://mrtgr.com/post/180308864563/mongodb-ecosystem

Υπάρχει επίσης μια συγκεκριμένη λύση για φορητές συσκευές, η οποία σημαίνει ότι ο χρήστης μπορεί να εγκαταστήσει το MongoDB σε μια κινητή συσκευή για άμεση αποθήκευση δεδομένων και δίνει τη δυνατότητα εργασίας χωρίς σύνδεση στο Διαδίκτυο.

Στην παρούσα εργασία χρησιμοποιήθηκε η αυτοδιαχειριζόμενη λύση, καθώς υπήρχε ενδιαφέρον και για το παρασκήνιο, δηλαδή πως γίνεται η σύνδεση της βάσης «χειροκίνητα» και η μελέτη της ανάπτυξης μιας εφαρμογής εξ' ολοκλήρου, από τη διεπαφή χρήστη μέχρι και το κομμάτι της αποθήκευσης δεδομένων. Συνεπώς, η βάση δεδομένων αναπτύχθηκε, δημιουργήθηκε με κώδικα και προγραμματίστηκε ώστε να αποθηκεύει αυτόματα τα δεδομένα όταν ο χρήστης «δώσει την εντολή» από την εφαρμογή.

Η MongoDB παρέχει και άλλα εργαλεία για τη βάση δεδομένων όπως το Compass, μια γραφική διεπαφή χρήστη, η οποία επιτρέπει την σύνδεση στη βάση μέσω μιας εφαρμογής Desktop και μπορεί ο προγραμματιστής μέσω αυτής να εκτελέσει τις βασικές CRUD λειτουργίες. Το Compass χρησιμοποιήθηκε στην παρούσα εργασία για να γίνει ο έλεγχος της σωστής λειτουργίας της βάσης.

Κάποια άλλα εργαλεία, περιλαμβάνουν το BI Connector ή γραφήματα Mongodb, αν η εφαρμογή θέλει να μαζέψει δεδομένα και να τα αναλύσει. Το BI Connector επιτρέπει τη χρήση του MongoDB ως πηγή δεδομένων για το BI και τις πλατφόρμες ανάλυσης, για τη δημιουργία απεικονίσεων και πινάκων που βοηθούν στην εξαγωγή των πληροφοριών και της αξίας των πολυδομημένων δεδομένων. Τα γραφήματα MongoDB είναι ο γρηγορότερος και ευκολότερος τρόπος δημιουργίας οπτικοποιημένων δεδομένων MongoDB. Με τη σύνδεση στη βάση δεδομένων μπορεί ο προγραμματιστής να εξάγει τα δεδομένα, να δημιουργήσει γραφήματα, πίνακες εργαλείων καθώς δίνει και τη δυνατότητα να τα μοιραστεί με άλλους χρήστες για συνεργασία.

Ένα από τα τελευταία εργαλεία που αναπτύχθηκαν από τη MongoDB είναι το stitch το οποίο είναι μια πλατφόρμα χωρίς διακομιστή, που διευκολύνει την εργασία που έχει να κάνει ο προγραμματιστής στο παρασκήνιο, καθώς επιτρέπει στους προγραμματιστές να εκτελούν απλές λειτουργίες JavaScript στην πλατφόρμα χωρίς διακομιστή Stitch, διευκολύνοντας την εφαρμογή της λογικής των εφαρμογών, την ασφαλή ενσωμάτωση με υπηρεσίες cloud και μικροϋπηρεσίες και τη δημιουργία API.

Αξίζει μια μικρή αναφορά στους ενεργοποιητές βάσης δεδομένων (database triggers), μια υπηρεσία που επιτρέπει στον προγραμματιστή να «ακούσει» γεγονότα σε μια βάση δεδομένων, όπως για παράδειγμα την εισαγωγή ενός εγγράφου και στη συνέχεια να εκτελέσει μια λειτουργία ως απάντηση σε αυτό. Τέλος, υπάρχει μια δυνατότητα που ονομάζεται συγχρονισμός σε πραγματικό χρόνο (Real-Time Sync), η οποία έχει σχεδιαστεί για να συγχρονίσει μια βάση δεδομένων σε ένα «σύννεφο» με τη συγκεκριμένη βάση δεδομένων.

2.11.2 Mongoose

Ανακεφαλαιώνοντας τα προαναφερθέντα, εισαγόμαστε στο Mongoose. Η MongoDB είναι μια βάση δεδομένων που αποθηκεύει τα δεδομένα ως έγγραφα τα οποία συνήθως έχουν δομή JSON :

```
{
  firstName: "Dimitris",
  lastName: "Oikonomou"
}
```

Ένα έγγραφο στη συνέχεια τοποθετείται σε μια συλλογή. Το παραπάνω παράδειγμα εγγράφου ορίζει ένα αντικείμενο χρήστη. Αυτό το αντικείμενο χρήστη θα ήταν συνήθως μέρος μιας συλλογής που ονομάζεται χρήστες.

Το Mongoose είναι ένας Object Document Mapper (ODM). Αυτό σημαίνει ότι το Mongoose επιτρέπει στον προγραμματιστή να ορίσει αντικείμενα με ένα σχήμα που αντιστοιχεί σε ένα έγγραφο MongoDB. Όλα στο Mongoose ξεκινούν με ένα σχήμα. Κάθε σχήμα αντιστοιχεί σε μια συλλογή MongoDB και καθορίζει το σχήμα των εγγράφων μέσα σε αυτήν τη συλλογή. Οι επιτρεπόμενοι τύποι σχήματος είναι:

1. Αλφαριθμητικός (String)
2. Αριθμός (Number)
3. Ημερομηνία (Date)
4. Ρυθμιστής (Buffer)
5. Δυαδική τιμή (Boolean)
6. Μικτό (Mixed): Ένας αυθαίρετος τύπος σχήματος.

7. ObjectId: Αντιπροσωπεύει την ταυτότητα ενός μοντέλου στη βάση δεδομένων. Μπαίνει από προεπιλογή ή επιλέγει ο προγραμματιστής.
8. Διάνυσμα (Array): Μπορούν να εκτελεστούν λειτουργίες πίνακα JavaScript σε αυτά τα μοντέλα (push, pop, unshift κ.λπ.)
9. Δεκαδικός 128 (Decimal128)
10. Χάρτης (Map)

Ο τύπος σχήματος δεν καθορίζει μόνο τη δομή των αντικειμένων και των εγγράφων, αλλά και λειτουργίες όπως τις μεθόδους εμφάνισης (instance method), τις στατικές μεθόδους μοντέλου (static Model method), καθώς και το ενδιάμεσο λογισμικό (middleware). Κάθε τύπος δεδομένων επιτρέπει να καθορισθούν:

- Μια προεπιλεγμένη τιμή
- Μια προσαρμοσμένη συνάρτηση επικύρωσης
- Εάν ένα πεδίο απαιτείται, δηλαδή εάν θα πρέπει οπωσδήποτε να έχει τιμή
- Μια συνάρτηση get() που επιτρέπει τον χειρισμό των δεδομένων πριν αυτά επιστρέψουν ως αντικείμενο
- Μια συνάρτηση set() που επιτρέπει τον χειρισμό των δεδομένων πριν αυτά αποθηκευτούν στην βάση δεδομένων.
- Την δημιουργία ευρετηρίων ώστε να υπάρχει η δυνατότητα για ταχύτερη ανάκτηση των δεδομένων.

Πέρα από τις παραπάνω επιλογές, ορισμένοι τύποι δεδομένων επιτρέπουν την περαιτέρω προσαρμογή στον τρόπο αποθήκευσης και ανάκτησης των δεδομένων από τη βάση δεδομένων.

ΕΓΚΑΤΑΣΤΑΣΗ & ΛΕΙΤΟΥΡΓΙΑ:

Το Mongoose εγκαθίσταται με την εντολή:

>*npm install mongoose –save*

Το πρώτο πράγμα που πρέπει να κάνει ο προγραμματιστής μετά την εγκατάσταση είναι να συμπεριλάβει το mongoose στο έργο του και να ανοίξει μια σύνδεση με τη δοκιμαστική βάση δεδομένων στην τοπική παρουσία του MongoDB. Αυτό γίνεται με την request().

```
const mongoose = require("mongoose");
```

Το Mongoose επιτρέπει στον προγραμματιστή να αρχίσει να χρησιμοποιεί τα μοντέλα αμέσως, χωρίς να πρέπει να περιμένει να δημιουργηθεί η σύνδεση με το MongoDB.

```
mongoose.connect('mongodb://localhost:27017/myapp');
mongoose.connect(mongoDB, {useNewUrlParser: true, useUnifiedTopology: true});
```

Έστω το παρακάτω σχήμα:

```
const blogSchema = new Schema({
  title:  String, // String is shorthand for {type: String}
  author: String,
  body:   String,
  comments: [{ body: String, date: Date }],
  date: { type: Date, default: Date.now },
  hidden: Boolean,
  meta: {
    votes: Number,
    favs:  Number
  }
});
```

Κάθε κλειδί στον κώδικα για το blogSchema ορίζει μια ιδιότητα στα έγγραφά, η οποία θα μεταδίδεται στο σχετικό SchemaType. Για παράδειγμα, ο τίτλος ιδιοκτησίας θα μεταδοθεί στο String SchemaType και η ημερομηνία θα μεταδοθεί σε SchemaType ημερομηνίας. Στα κλειδιά μπορούν επίσης να εκχωρηθούν ένθετα αντικείμενα που περιέχουν περαιτέρω ορισμούς κλειδιών/τύπων όπως η παραπάνω ιδιότητα meta. Αυτό θα συμβαίνει κάθε φορά που η τιμή ενός κλειδιού είναι ένα POJO (Plain Old JavaScript Object) που δεν έχει ιδιότητα τύπου. Σε αυτές τις περιπτώσεις, το Mongoose δημιουργεί μόνο πραγματικές διαδρομές σχήματος για φύλλα στο δέντρο (όπως meta.votes και meta.favs παραπάνω) και τα κλαδιά δεν έχουν πραγματικές διαδρομές. Μια παρενέργεια αυτού είναι ότι το meta δεν μπορεί να έχει τη δική του επικύρωση. Εάν απαιτείται επικύρωση στο δέντρο, πρέπει να δημιουργηθεί μια διαδρομή στο δέντρο.

Τα σχήματα για να μπορούν να χρησιμοποιηθούν από τον προγραμματιστή πρέπει να «μεταγλωττιστούν» σε μοντέλα. Αυτό γίνεται με το mongoose.model (modelName, schema):

```
const Blog = mongoose.model('Blog', blogSchema);
```

Όταν καλείται το mongoose.model () σε ένα σχήμα, το Mongoose συντάσσει ένα μοντέλο για τον προγραμματιστή. Η πρώτη μεταβλητή στην παρένθεση είναι το όνομα της συλλογής σε ενικό (και του μοντέλου) και η δεύτερη το όνομα του σχήματος. Το Mongoose αναζητά αυτόματα την πληθυντική, σε πεζά έκδοση του ονόματος του μοντέλου.
Από προεπιλογή, το Mongoose προσθέτει μια ιδιότητα _id στα σχήματα. Ο προγραμματιστής μπορεί αν θελήσει να αντικαταστήσει το προεπιλεγμένο _id του Mongoose με το δικό του _id.

3.ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

3.1 ΓΕΝΙΚΑ

Σε αυτό το κεφάλαιο θα αναλυθεί η διαδικασία υλοποίησης της εφαρμογής. Περιλαμβάνει την εξήγηση χρήσης των τεχνολογιών που αναφέρθηκαν στο κεφάλαιο 1, καθώς και κάποια κομμάτια κώδικα, όπου απαιτείται ώστε να παρουσιαστεί με ποιον τρόπο επιτεύχθηκε η λειτουργικότητα.

3.2 ΣΧΕΔΙΑΣΜΟΣ ΤΗΣ ΔΙΕΠΑΦΗΣ ΧΡΗΣΤΗ (USER INTERFACE: UI)

Ο σχεδιασμός της διεπαφής χρήστη έχει να κάνει με το πώς αντιλαμβάνεται και βιώνει ο χρήστης μια ιστοσελίδα/εφαρμογή. Κάθε προγραμματιστής ιστού πρέπει να λαμβάνει υπόψιν του τη διεπαφή χρήστη, έχοντας στο μυαλό του σε ποιους απευθύνεται η ιστοσελίδα/εφαρμογή που αναπτύσσει, για παράδειγμα σε τι ηλικιακή ομάδα ανήκουν. Κάποιοι γενικοί «κανόνες» είναι οι εξής:

- Να είναι απλή. Όσο πιο «φορτωμένη» είναι μια ιστοσελίδα τόσο πιο πολύ μπορεί να κουράσει τον χρήστη ή ακόμα και να μην παρατηρήσει αυτά που θέλει ο προγραμματιστής ιστού. Οι καλύτερες ιστοσελίδες είναι αυτές που ο σχεδιασμός τους είναι σχεδόν αόρατος στο μάτι, τα στοιχεία δένουν μεταξύ τους και δίνουν μια ομαλή εικόνα.
- Να υπάρχει συνοχή. Όλες οι σελίδες ή τα διαφορετικά κομμάτια της εφαρμογής να ακολουθούν το ίδιο μοτίβο και την ίδια σχεδίαση. Με αυτό τον τρόπο ο χρήστης «συνηθίζει» το περιβάλλον που έχει δημιουργήσει ο προγραμματιστής, αισθάνεται πιο άνετα και είναι σε θέση να κάνει πιο γρήγορα αυτό που θέλει.
- Να δίνεται η απαραίτητη προσοχή στη διάταξη της σελίδας. Ο προγραμματιστής πρέπει ανάλογα με το είδος της εφαρμογής που φτιάχνει να διατάσσει το κάθε κομμάτι αντίστοιχα. Η προσεκτική τοποθέτηση αντικειμένων (ανάλογα με τα κενά ανάμεσα τους, τον τρόπο που συνδέονται) μπορεί να βοηθήσει τον προγραμματιστή να τραβήξει την προσοχή στα πιο σημαντικά κομμάτια πληροφοριών και μπορεί να βοηθήσει στη σάρωση και την αναγνωσιμότητα.
- Να δίνεται προσοχή στην επιλογή χρωμάτων και υφής. Ο προγραμματιστής μπορεί να στρέψει την προσοχή προς ή να ανακατευθύνει την προσοχή του χρήστη από αντικείμενα χρησιμοποιώντας χρώμα, φως, αντίθεση και υφή προς όφελος του.

- Να δίνεται προσοχή στην τυπογραφία ώστε να δημιουργηθεί ιεραρχία και σαφήνεια. Είναι προτιμότερο να χρησιμοποιούνται διαφορετικά μεγέθη, γραμματοσειρές και διάταξη του κειμένου που βοηθούν στην αύξηση της σαρωσιμότητας και της αναγνωσιμότητας.

Σύμφωνα με τα παραπάνω έγινε προσπάθεια να αναπτυχθεί η διεπαφή χρήστη με τον καλύτερο δυνατό τρόπο. Ο σχεδιασμός είναι απλός, χωρίς να αποσπάται η προσοχή του χρήστη από τον στόχο, που είναι να πάρει μέρος στην έρευνα ή/και να διαβάσει περισσότερα για τη χρήση του δημόσιου χώρου είτε σε σχέση με τον Covid-19 είτε γενικά σε αστικό περιβάλλον. Τα χρώματα που επιλέχθηκαν είναι συνολικά πέντε, χρησιμοποιούνται τα ίδια σε όλη την εφαρμογή και επιλέχθηκαν από το site colorhunt.co, το οποίο περιλαμβάνει ομάδες αποχρώσεων που έχουν δημιουργηθεί από προγραμματιστές ιστού ώστε να ταιριάζουν μεταξύ τους και είναι αισθητικά ομαλή η συνύπαρξή τους. Η απόχρωση του χρώματος που φαίνεται στο φόντο είναι προς το μαύρο και έγινε η συγκεκριμένη επιλογή γιατί οι σκούρες αποχρώσεις είναι πιο «ξεκούραστες» στο μάτι, ιδιαίτερα για διάβασμα.

3.3 ΔΟΜΗ ΦΑΚΕΛΟΥ ΕΡΓΟΥ

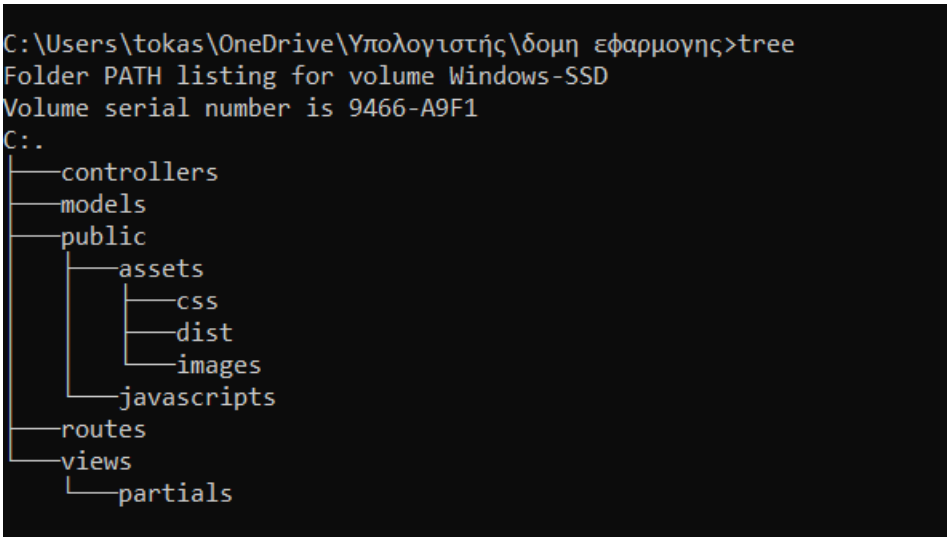
Στο κεφάλαιο 2.5.2 διατυπώθηκε ο τρόπος με τον οποίο γίνεται η ταξινόμηση των φακέλων στον φάκελο ενός έργου (εφαρμογής/ιστοσελίδας) που χρησιμοποιεί Node, Express κλπ. Στο παρόν κεφάλαιο θα αναλυθεί η δομή φακέλου για τη συγκεκριμένη εφαρμογή.

Από το τερματικό των Windows, ο προγραμματιστής εφόσον μεταφερθεί στον φάκελο του έργου μπορεί με την εντολή:

>tree

να δει τους φακέλους που περιέχει το έργο του.

Να σημειωθούν δύο πράγματα, πρώτον ότι δεν βλέπει τα αρχεία, αλλά μόνο τους φακέλους και δεύτερον ότι έχει εξαιρεθεί ο φάκελος node_modules διότι περιέχει 592 φακέλους. Τα σημαντικότερα πακέτα που εγκαταστάθηκαν για την ανάπτυξη της εφαρμογής αναφέρονται στην παρούσα εργασία. Παρακάτω φαίνεται η δομή του φακέλου:



Εικόνα 3.1 Δομή φακέλου

- Ο φάκελος controllers περιέχει τα αρχεία που διαχειρίζονται τα αιτήματα (requests) και ττις απαντήσεις (responses) των αρχείων των models και views. Δηλαδή, λαμβάνει τα αιτήματα δεδομένων (data requests), τα στέλνει στα αρχεία του models και από εκεί λαμβάνει την απάντηση και τη στέλνει πίσω στο views.
- Ο φάκελος models περιέχει τα αρχεία που καθιστούν την εφαρμογή λειτουργική ως προς την επικοινωνία της με τη βάση δεδομένων.
- Ο φάκελος public περιέχει ότι αρχείο μπορεί να δει ο χρήστης αν πατήσει στην σελίδα στον ιστό πλοήγησης «έλεγχο». Στον υποφάκελο *css* υπάρχουν τα αρχεία .css που μορφοποιούν την εφαρμογή. Στη συγκεκριμένη εφαρμογή υπάρχουν δύο, ένα για την μορφοποίηση ολόκληρης της εφαρμογής και ένα για την μορφοποίηση του χάρτη. Ο υποφάκελος *dist* περιλαμβάνει τα αρχεία css και JavaScript που ελαχιστοποιούνται (Κεφάλαιο 2.6 Minifying) από το gulp. Ο υποφάκελος *images* περιέχει όλες τις εικόνες που χρησιμοποιούνται στην εφαρμογή. Ο υποφάκελος *js* περιέχει ένα αρχείο JavaScript που χρειάζεται για τον χάρτη leaflet. Τέλος, στον φάκελο javascripts περιέχονται ότι αρχεία JavaScript χρειάζονται για την εφαρμογή, είτε για τη λειτουργία του χάρτη, είτε για τη λειτουργικότητα της υπόλοιπης εφαρμογής (για παράδειγμα για τη λειτουργία των διαδραστικών κουμπιών στην αρχική σελίδα).
- Ο φάκελος routes περιλαμβάνει τα αρχεία που προωθούν τα υποστηριζόμενα αιτήματα (και τυχόν πληροφορίες που κωδικοποιούνται σε αιτήματα διευθύνσεων URL) στις κατάλληλες λειτουργίες του controller. Δηλαδή, δημιουργούν τις συνδέσεις και τις διαδρομές για να εμφανίζονται δυναμικά οι σελίδες.

- Ο φάκελος views περιέχει όλα τα .ejs αρχεία, δηλαδή τις ιστοσελίδες από τη μεριά του πελάτη. Στον υποφάκελο partials, περιέχονται τα .ejs αρχεία που περιλαμβάνουν τα κομμάτια κώδικα που υπάρχουν σε όλες τις σελίδες, δηλαδή τα κομμάτια για την κεφαλίδα, <head> (head.ejs), τη μπάρα πλοήγησης (navbar.ejs) και τα scripts της Bootstrap (jsFiles.ejs).

3.4 ΣΧΕΔΙΑΣΗ ΠΛΟΗΓΗΣΗΣ ΧΡΗΣΤΗ

Σκοπός είναι να πάρει μέρος ο χρήστης στην έρευνα και αν ενδιαφέρεται να μάθει περισσότερα να πλοηγηθεί στις σελίδες About και βιβλιογραφίας. Σε αυτό το κεφάλαιο θα αναφερθεί πως υλοποιήθηκαν κάποιες χρηστικές (για τον προγραμματιστή) λεπτομέρειες, όπως για παράδειγμα η αδυναμία να προχωρήσει ο χρήστης στην επόμενη σελίδα αν δεν έχει απαντήσει σε όλες τις ερωτήσεις και κάποιες αισθητικές λεπτομέρειες (για τον χρήστη) όπως τα δύο κυκλικά κουμπιά στην αρχική σελίδα.

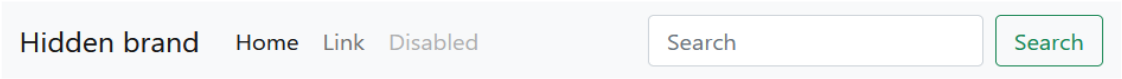
Αρχικά, θα γίνει αναφορά στην υλοποίηση των κομματιών που επηρεάζουν την αισθητική της εφαρμογής και στη λειτουργικότητα τους.

Μπάρα πλοήγησης:

Για τη λειτουργία της μπάρας πλοήγησης χρησιμοποιήθηκε Bootstrap και συγκεκριμένα η μπάρα πλοήγησης που διαθέτει η οποία έχει αποκρισιμότητα στις κινητές συσκευές. Ο κώδικας υλοποίησης της είναι ο παρακάτω:

```
<nav class="navbar navbar-expand-lg">
  <div class="container-fluid">
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarTogglerDemo02" aria-controls="navbarTogglerDemo02" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarTogglerDemo02">
      <img src="/assets/images/tree-logo.png" class="tree-logo" />
      <ul class="unordered-list ml-auto navbar-nav mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link" aria-current="page" href="/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/about">About</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Η λειτουργικότητα της μπάρας πλοήγησης παρέχεται από την Javascript που έχει ενσωματωμένη η Bootstrap όπως εξηγήθηκε στο κεφάλαιο 2.5.3. Το αποτέλεσμα της μπάρας πλοήγησης απλώς με τον κώδικα της Bootstrap θα ήταν αυτό:



Εντούτοις η μορφοποίηση άλλαξε για να ταιριάξει στην αισθητική της εφαρμογής μέσω CSS, με το αποτέλεσμα να είναι αυτό που φαίνεται στην παρακάτω εικόνα:

Αυτό επιτεύχθηκε, αφαιρώντας από τον κώδικα της Bootstrap το κομμάτι με τη μπάρα αναζήτησης διότι δεν χρειαζόταν για την παρούσα εφαρμογή, φτιάχνοντας τους συνδέσμους για να οδηγούν τον χρήστη εκεί που πρέπει και με την παρακάτω μορφοποίηση στο style.css αρχείο:

```css
/* NAVIGATION BAR */
a {
  text-decoration: none;
  color: #f8f5f1;
}
a:hover {
  color: #5aa897;
}
.navbar {
  background-color: #202125;
  font-family: "Roboto Condensed", sans-serif;
  font-size: 20px;
  display: flex;
  justify-content: space-between;
}
.tree-logo {
  height: 3.75rem;
  width: 3.75rem;
  margin-left: 2rem;
}
.unordered-list {
  margin-left: auto;
}
```

Επεξήγηση της εικόνας:

- **a :** ετικέτα «αγκύρωσης» (anchor tag), οι σύνδεσμοι στην μπάρα πλοήγησης, δηλαδή το Home και το About.
- **a:hover:** όταν ο κέρσορας περνάει πάνω από την ετικέτα αλλάζει το χρώμα.
- **.navbar**: Με την τελεία μπροστά από το όνομα του κάθε αντικειμένου, γίνεται δήλωση ότι πρόκειται για κλάση στον HTML κώδικα όπως εξηγήθηκε και στο κεφάλαιο 1.2.1. Η κλάση navbar αναφέρεται σε ολόκληρη τη μπάρα πλοήγησης.
- **.tree-logo**: Αναφέρεται στη μικρή εικόνα του δένδρου που φαίνεται στην αριστερά άκρη της μπάρας πλοήγησης.
- **.unordered-list**: αναφέρεται στην αταξινόμητη λίστα της μπάρας πλοήγησης. Η μπάρα πλοήγησης είναι δομημένη με την βοήθεια αταξινόμητης λίστας της HTML, δηλαδή οι ετικέτες Home, About αποτελούν τα δύο κομμάτια της λίστας.

Κουμπιά αρχικής σελίδας:

Η εναλλαγή στα κουμπιά χρειάστηκε εκτός από CSS και JavaScript. Το αποτέλεσμα που φαίνεται στην αρχική σελίδα επιτεύχθηκε με την αλλαγή κλάσης, προσθέτοντας σε κάθε κλάση τα χαρακτηριστικά των κουμπιών πριν και μετά το πέρασμα του κέρσορα από πάνω τους. Ο κώδικας JavaScript που χρειάστηκε είναι ο ακόλουθος:

```
const buttonOne = document.getElementById("homeButtonOne");
const buttonTwo = document.getElementById("homeButtonTwo");
buttonOne.addEventListener("mouseenter", addClassFirst);
buttonTwo.addEventListener("mouseenter", addClassSecond);

function addClassFirst() {
  buttonOne.classList.add("dotButton1");
  buttonOne.classList.remove("dotButtonBeforeClicked1");
}
function addClassSecond() {
  buttonTwo.classList.add("dotButton2");
  buttonTwo.classList.remove("dotButtonBeforeClicked2");
}
```

Στον κώδικα HTML έχουν καθοριστεί με βάση την ταυτότητα (Id) τα δύο κουμπιά: buttonOne, buttonTwo. Με την εντολή document.getElementById(“name”) καλείται το αντικείμενο που έχει την ταυτότητα name. Με την εντολή name.addEventListener(“…”, function_name) προστίθεται στο αντικείμενο η ικανότητα να «ακούσει» κάποιο γεγονός που ορίζεται μέσα στην παρένθεση και όταν θα το «ακούσει» θα καλεστεί η συνάρτηση function_name. Στη συγκεκριμένη περίπτωση το γεγονός που ακούνε τα δύο κουμπιά είναι το εισέλθει στην περιοχή που βρίσκονται ο κέρσορας. Τότε καλείται η συνάρτηση addClass η οποία αφαιρεί την κλάση που είχαν προηγουμένως τα κουμπιά που ήταν μορφοποιημένη στο style.css αρχείο (μαύρο χρώμα) και προσθέτει την καινούρια κλάση η οποία είναι επίσης μορφοποιημένη στο style.css αρχείο (προσθέτει χρώμα και αλλαγή χρώματος στα γράμματα όταν περνάει ο κέρσορας από πάνω). Οι παραπάνω κλάσεις έχουν μορφοποιηθεί ως εξής:

```css
/* buttons before the user clicks on them */
.dotButtonBeforeClicked1,
.dotButtonBeforeClicked2 {
  background-color: black;
  color: black;
  border-radius: 50%;
  width: 10rem;
  height: 10rem;
  text-align: center;
  position: absolute;
}
.dotButtonBeforeClicked1 {
  margin: 14rem 0 0 20.37rem;
  padding-top: 3rem;
}
.dotButtonBeforeClicked2 {
  margin: 14rem 0 0 60.74rem;
  padding-top: 3rem;
}

/* buttons when the user clicks on them */
.dotButton1,
.dotButton2{
  position: absolute;
  color: #f8f5f1;
  border-radius: 50%;
  width: 10rem;
  height: 10rem;
  text-align: center;
  padding-top: 3rem;
}
.dotButton1 {
  background-color: #5aa897;
  margin: 14rem 0 0 20.37rem;
}
.dotButton2 {
  background-color: #45526c;
  margin: 14rem 0 0 60.74rem;
}
.dotButton1:hover {
  color: #45526c;
}
.dotButton2:hover {
  color: #5aa897;
}
```

Στο επόμενο κομμάτι, θα γίνει αναφορά στα λειτουργικά για τον προγραμματιστή εξαρτήματα της εφαρμογής.

Κουμπί Next στις σελίδες με τις ερωτήσεις:

Προκειμένου να υπάρχει σιγουριά ότι ο χρήστης θα απαντήσει σε όλες τις ερωτήσεις και θα εισάγει και το όνομα του ώστε να μπορεί να είναι ολοκληρωμένη η καταχώρηση στην βάση δεδομένων, πρέπει να προσαρμοστεί ο κώδικας ώστε να «μην αφήνει» τον χρήστη να προχωρήσει στην επόμενη σελίδα εάν δεν έχει ολοκληρώσει αυτά που του ζητούνται. Αυτό γίνεται μέσω JavaScript. Για την εισαγωγή του ονόματος:

```
document.getElementById('nextButtonNamePre').onclick = function() {
              var name = document.getElementById("name").value
              if (name == "") {
                alert("Εισάγετε το όνομα σας!")
                return
              }
```

Για τις απαντήσεις στις ερωτήσεις:

```
if (value1 == "") {
                alert("Ξέχασες να απαντήσεις στην πρώτη ερώτηση!")
                return
              }
```

Για την καταχώρηση στον χάρτη:

```
if (document.getElementById("position_x") == undefined) {
                alert("Εισάγετε τοποθεσία στον χάρτη!")
                return
```

Σε κάθε περίπτωση, εάν ο χρήστης έχει αφήσει κενό κάτι από τα παραπάνω, εμφανίζεται ένα alert, δηλαδή ένα μικρό παράθυρο που τον ενημερώνει τι έχει αφήσει κενό και ότι πρέπει να το συμπληρώσει.

3.5 ΣΥΝΔΕΣΗ LEAFLET MAP

Αρχικά, πρέπει να γίνει κάποια προετοιμασία πριν ενσωματωθεί ο χάρτης στην σελίδα. Το leaflet έχει ήδη ενσωματωμένα χαρακτηριστικά, μορφοποίηση και JavaScript αποκρισιμότητα, τα οποία θα πρέπει να μπουν στο κομμάτι του κώδικα. Οι σελίδες που θα περιέχουν τον χάρτη θα πρέπει στο κομμάτι του <head> στο ejs αρχείο να περιλαμβάνουν το παρακάτω κομμάτι κώδικα που οδηγεί στην μορφοποίηση του leaflet δηλαδή στο CSS.

```
<link rel="stylesheet"
href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
   integrity="sha512-
xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZMZ19scR4
PsZChSR7A=="
   crossorigin=""/>
```

Έπειτα, κάτω από το παραπάνω <link> θα πρέπει να προστεθεί το παρακάτω κομμάτι κώδικα που οδηγεί στην JavaScript λειτουργικότητα του χάρτη που έχει ήδη ενσωματώσει η leaflet για τον προγραμματιστή. Μπορούμε να παρατηρήσουμε πως το leaflet «ζητάει» το <script> της JavaScript να μπει στο <head> section σε αντίθεση με της Bootstrap και σε αντίθεση γενικότερα με τα <script> τα οποία μπαίνουν πριν το κλείσιμο του σώματος της HTML δηλαδή πριν το </body>.

```
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
   integrity="sha512-
XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaXaVUearIOBhiX
Z5V3ynxwA=="
   crossorigin=""></script>
```

Στη συγκεκριμένη εφαρμογή όμως, ακριβώς επειδή πρόκειται για εφαρμογή που «χτίζεται» με το Node.js έγινε εγκατάσταση του πακέτου του leaflet map με τη βοήθεια του npm πακέτου. Με αυτόν τον τρόπο περιέχονται τα αρχικοποιημένα αρχεία JavaScript και CSS οπότε παραλείφθηκε το παραπάνω βήμα και στο αρχείο webmap.js στον φάκελο assets>js «ζητήθηκε» το πακέτο της leaflet που εγκαταστάθηκε ως εξης:

```
var L = require("leaflet");
```

Στη συνέχεια διαμορφώνεται το κομμάτι της σελίδας στο οποίο θα μπει ο χάρτης. Δημιουργούμε ένα «κουτί», ένα που θα περιέχει τον χάρτη. Μαζί με το προστίθεται και το <script> που θα περιέχει τη λειτουργικότητα του χάρτη, δηλαδή την JavaScript:

```
<div id="map"></div>
<script src="javascripts/webmap.js"></script>
```

Τέλος, για το αρχικό κομμάτι διαμόρφωσης του χάρτη, θα πρέπει στο CSS αρχείο να προσδιοριστεί το μέγεθος του χάρτη και ιδιαίτερα το ύψος, διότι δεν είναι διαμορφωμένο από τα αρχικά κομμάτια κώδικα που προστέθηκαν στο HTML αρχείο.

```
#map {
  width: auto;
  height: 500px;
  margin-top: 13%;

}
```

Το κουτί που θα φιλοξενήσει τον χάρτη είναι έτοιμο και μπορεί να αρχίσει να διαμορφώνεται ανάλογα με τις προτιμήσεις και τους στόχους του προγραμματιστή. Γίνεται αρχικοποίηση του χάρτη και επιλογή των γεωγραφικών συντεταγμένων και επιπέδου ζουμ που θα βλέπει ο χρήστης όταν θα φορτώνει η σελίδα με τον χάρτη. Στη συγκεκριμένη περίπτωση επιλέχθηκε η περιοχή γύρω από Μοναστηράκι, Θησείο και Πλάκα. Από προεπιλογή (δηλαδή από τα αρχικά κομμάτια κώδικα και καθώς δεν αλλάχθηκε καμία επιλογή κατά τη δημιουργία του χάρτη), όλες οι αλληλεπιδράσεις με τον κέρσορα στον χάρτη είναι ενεργοποιημένες και διαθέτει στοιχεία ελέγχου ζουμ και απόδοσης. Στη συνέχεια προστίθεται το στρώμα πλακιδίων (tile layers) του χάρτη, στη συγκεκριμένη περίπτωση επιλέχθηκε το στρώμα από το OpenStreetMap αλλά μπορεί κάλλιστα να επιλεχθεί και από το MapBox και από άλλες ιστοσελίδες που διαθέτουν αντίστοιχο API. Τα παραπάνω πραγματοποιήθηκαν με τον εξής κώδικα:

```
var map = L.map("map").setView([37.9709797, 23.7241881], 20);
```

&

```
L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
  maxZoom: 19,
  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStre
etMap</a> contributors',
}).addTo(map);
```

Με τα παραπάνω βήματα, στήθηκε ο χάρτης και είναι έτοιμος για χρήση.

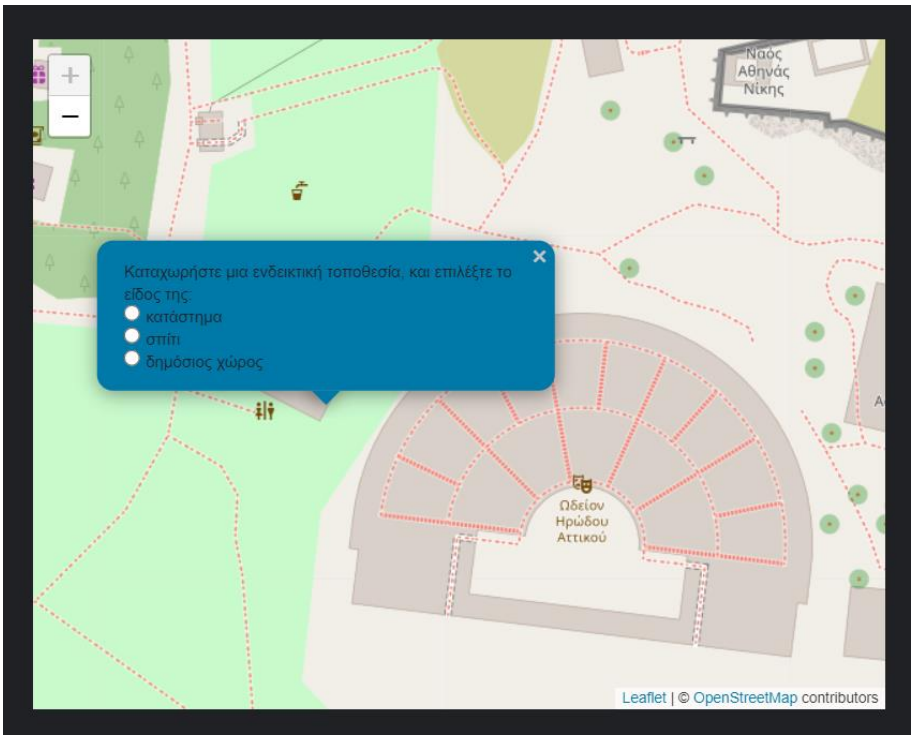
Στο μεγαλύτερο μέρος του, διατηρήθηκε η από προεπιλογή μορφοποίηση του χάρτη. Το μόνο που αλλάχθηκε είναι το αναδυόμενο παράθυρο που εμφανίζεται στον χάρτη όταν ο χρήστης πατάει πάνω σε μια τοποθεσία αναδύεται ένα μικρό παράθυρο. Το αναδυόμενο παράθυρο μορφοποιήθηκε με τον εξής τρόπο:

```
.leaflet-popup-content-wrapper,
.leaflet-popup-tip {
  background: #0078a8;
  color: #202125;
  box-shadow: 0 3px 14px rgba(0, 0, 0, 0.4);
}
```

και αλλάχθηκε το περιεχόμενο της, προσθέτοντας μια ερώτηση που ο χρήστης πρέπει να επιλέξει κάποια από τις εμφανιζόμενες επιλογές:

```
map.on("click", function (e) {
  var lt = String(e.latlng.lat),
    lg = String(e.latlng.lng);
  var popup = L.popup()
    .setLatLng(e.latlng)
    .setContent('<div class="custom-control custom-radio question-
div"><input type = "hidden" id = "position_x" value = '+lt+'><input type = "hidde
n" id = "position_y" value = '+lg+'><label>Καταχωρήστε μια ενδεικτική τοποθεσία,
και επιλέξτε το είδος της:</label><br><label for="q8a1"><input class="custom-
control-
input" type="radio" id="q8a1" name="question8" required> κατάστημα</label><br><la
bel for="q8a2"><input class="custom-control-
input" type="radio" id="q8a2" name="question8" required> σπίτι</label><br><label
for="q8a3"><input class="custom-control-
input" type="radio" id="q8a3" name="question8" required> δημόσιος χώρος</label><b
r></div>')
    .openOn(map);
});
```

Τα παραπάνω έχουν το ακόλουθο αποτέλεσμα:



Εικόνα 3.2 Αναδυόμενο παράθυρο χάρτη

Η καταχώρηση έχει κωδικοποιηθεί σαν q8, δηλαδή σαν συνέχεια των ερωτήσεων. Η επιλογή που θα κάνει ο χρήστης αποθηκεύεται σαν επιλογή της «ερώτησης» 8. Στο επόμενο κεφάλαιο που θα αναλυθεί την διαδικασία σύνδεσης της βάσης δεδομένων και της αποθήκευσης τους, θα εξηγηθεί και η διαδικασία αποθήκευσης των δεδομένων του χάρτη.

3.6 ΔΗΜΙΟΥΡΓΙΑ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΣΥΝΔΕΣΗ ΜΕ ΤΗΝ ΕΦΑΡΜΟΓΗ

3.6.1 Σύνδεση βάσης δεδομένων με την εφαρμογή

Η αρχική σύνδεση της βάσης δεδομένων, δηλαδή του mongoose με την εφαρμογή γίνεται με τις εξής εντολές στο αρχείο app.js:

```
const mongoose = require("mongoose");
```

Όπως έχει προαναφερθεί «απαιτείται» το πακέτο mongoose.

```
// connect to mongodb & listen for requests
const db = require("./models/db");
```

Συνδέεται με τη βάση και «ακούει» τα αιτήματα. Το αρχείο το οποίο απαιτείται στη συγκεκριμένη περίπτωση είναι το db.js το οποίο βρίσκεται στον φάκελο models και είναι το τελευταίο κομμάτι που απαιτείται για να ολοκληρωθεί η σύνδεση.

Το αρχείο db.js, που βρίσκεται στον φάκελο models, αποτελείται από τον εξής κώδικα:

```
const mongoose = require("mongoose");

mongoose.set("useCreateIndex", true); // This prevents collection.ensureIndex dep
reciation warning
mongoose.set("useFindAndModify", false); // This prevents current mongoose deprec
iation warning


  mongoose.connect("mongodb://localhost:27017/myCovidApp", { useNewUrlParser: tru
e ,useUnifiedTopology: true}).then(() => {
      console.log('Database is connected');
  })
```

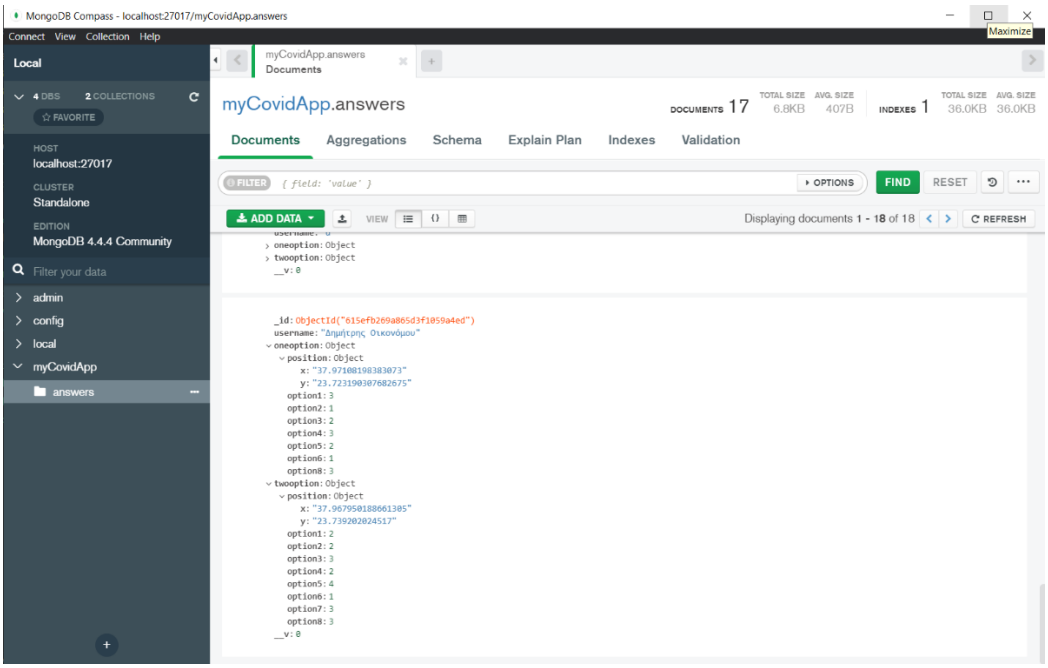
Το mongoose συνδέεται με τη MongoDB μέσω της θύρας 27017 τοπικά, μέσω localhost. Όταν γίνει η σύνδεση εμφανίζεται το μήνυμα επιτυχίας στην οθόνη του τερματικού που χρησιμοποιείται για την «εκτέλεση» της εφαρμογής.

Το επόμενο βήμα είναι η δημιουργία του σχήματος που θα ακολουθούν τα δεδομένα. Αυτό γίνεται στο αρχείο answer.js το οποίο βρίσκεται στον φάκελο models:

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const ObjectId = Schema.ObjectId;

const answerSchema = new Schema({
  username: {
    type: String,
    required: true,
  },
  oneoption: {
    type: Object,
    required: true,
  },
  twooption: {
    type: Object,
    required: true,
  }
});

module.exports = mongoose.model("Answers", answerSchema);
```

Στο συγκεκριμένο αρχείο, «απαιτείται» πάλι το mongoose και έπειτα, καλείται ένα σχήμα mongoose, με την εντολή const Schema = mongoose.Schema. Δηλαδή, δίνεται η εντολή ότι δηλώνεται ένα σχήμα που θα ακολουθήσει τη δομή ενός σχήματος όπως το ορίζει το mongoose. Το όνομα του σχήματος είναι answer στον ενικό και το όνομα της συλλογής είναι η πληθυντική εκδοχή του ονόματος, δηλαδή answers. Όπως αναλύθηκε και στο κεφάλαιο 2.11 τα σχήματα για να μπορούν να χρησιμοποιηθούν από τον προγραμματιστή πρέπει να «μεταγλωττιστούν» σε μοντέλα. Αυτό επιτυγχάνεται με την τελευταία εντολή: module.exports = mongoose.model("Answers", answerSchema). Μέσα στα άγκιστρα καθορίζονται τι τύπου δεδομένο είναι η κάθε καταχώρηση και εάν είναι απαιτούμενη η συμπλήρωση του. Τα oneoption και twooption είναι αντικείμενα, διότι έχουν ενσωματωμένες τις πληροφορίες για τις συντεταγμένες που καταχωρεί ο χρήστης στον χάρτη.

Η δομή για την συγκεκριμένη εφαρμογή είναι η εξής: αποθηκεύονται το όνομα του χρήστη, οι απαντήσεις στις ερωτήσεις της πρώτης σελίδας και οι απαντήσεις στις ερωτήσεις της δεύτερης σελίδας. Για να σχηματιστεί μια καλύτερη εικόνα της δομής των δεδομένων παρακάτω έχουμε ένα στιγμιότυπο από τη διεπαφή του MongoDB Compass που απεικονίζει τα δεδομένα ενός χρήστη που έχει απαντήσει στις ερωτήσεις.



Εικόνα 3.3 Διεπαφή Compass, δομή δεδομένων

Μπορούμε να δούμε το όνομα του χρήστη, τις απαντήσεις στις ερωτήσεις που αφορούν την περίοδο πριν τον Covid-19 στην καταχώρηση oneoption και ενσωματωμένο αντικείμενο μέσα σε αυτήν την καταχώρηση τις συντεταγμένες της τοποθεσίας που εισάγει ο χρήστης στον χάρτη. Η περιγραφή της τοποθεσίας του χάρτη βρίσκεται καταχωρημένη στην επιλογή οκτώ, δηλαδή στο option8.

3.6.2 Αποθήκευση δεδομένων

Το κομμάτι του κώδικα που αποθηκεύει τα δεδομένα που λαμβάνει από τον χρήστη είναι ο εξής:

```
const showEndOfSurveyPage = async (_req, res) => {
  const save_Data = _req.query;
  const data = { username: save_Data.username, oneoption: JSON.parse(save_Data.on
eoption), twooption:JSON.parse(save_Data.twooption)}
  const Newdata = new Answer(data)
  const result = await Newdata.save()
  if (result == "") {
      res.send({
          status: false
      })
  }
  else {
    res.render("endofsurvey");
  }
};
```

και περιέχεται στο αρχείο index.js στον φάκελο controllers.

Η διαδικασία αποθήκευσης εκτελείται σε τρία στάδια. Ο χρήστης πλοηγείται στην πρώτη σελίδα με τις ερωτήσεις που αφορά την περίοδο προ-Covid-19. Καθώς απαντάει στις ερωτήσεις, αυτές αποθηκεύονται στο localStorage, δηλαδή στην τοπική μνήμη του προγράμματος πλοήγησης.

```
  console.log(Oneoption)
  localStorage.username = name;
  localStorage.Oneoption = JSON.stringify(Oneoption);
  document.querySelector("form").submit();
}
```

Υπάρχει το Web Storage API, το οποίο είναι ένα σύνολο μηχανισμών που επιτρέπουν στα προγράμματα περιήγησης να αποθηκεύουν ζεύγη τιμών κλειδιών. Έχει σχεδιαστεί για να είναι πολύ πιο διαισθητικό από τη χρήση cookies. Τα ζεύγη τιμής κλειδιού αντιπροσωπεύουν αντικείμενα αποθήκευσης, τα οποία είναι παρόμοια με τα αντικείμενα, με τη διαφορά ότι παραμένουν ανέπαφα κατά τη διάρκεια φόρτωσης σελίδας και είναι πάντα συμβολοσειρές. Το

LocalStorage είναι μια ιδιότητα που επιτρέπει σε ιστότοπους και εφαρμογές JavaScript να αποθηκεύουν ζεύγη βασικών τιμών σε ένα πρόγραμμα περιήγησης ιστού χωρίς ημερομηνία λήξης. Αυτό σημαίνει ότι τα δεδομένα που είναι αποθηκευμένα στο πρόγραμμα περιήγησης θα παραμείνουν ακόμα και μετά το κλείσιμο του παραθύρου του προγράμματος περιήγησης.

Στην συνέχεια, ο χρήστης οδηγείται στην επόμενη σελίδα όπου συμπληρώνει τις ερωτήσεις που αφορούν την περίοδο κατά την διάρκεια του Covd-19. Ο κώδικας τότε παίρνει τα δεδομένα από την πρώτη σελίδα καθώς και από τη δεύτερη και τα καταχωρεί σε <input>μεταβλητές οι οποίες είναι κρυμμένες(hidden) (δηλαδή δεν φαίνονται στην φόρμα). Όταν ο χρήστης προχωρήσει στην σελίδα που του ανακοινώνει το τέλος της έρευνας, ο κώδικας «τραβάει» τα δεδομένα, με την εντολή JSON.parse() τα τροποποιεί ώστε να έρθουν στην αρχική τους μορφή (αντικείμενα από συμβολοσειρές), δημιουργεί μια «εισαγωγή απάντησης» (answer entry) και εκτελεί την εντολή NewData.save() που είναι μέθοδος εγγράφου του mongoose. Η μέθοδος save() είναι ασύγχρονη. Όταν δημιουργείται μια παρουσία ενός μοντέλου mongoose χρησιμοποιώντας τη μέθοδο new, η κλήση save() κάνει το mongoose να εισαγάγει ένα νέο έγγραφο. Αυτό γίνεται στον παραπάνω κώδικα με την εντολή *const Newdata = new Answer(data).* Εάν φορτωθεί ένα έγγραφο που υπάρχει ήδη από τη βάση δεδομένων και τροποποιηθεί, η μέθοδος save() ενημερώνει το υπάρχον έγγραφο. Το mongoose επικυρώνει τροποποιημένες διαδρομές πριν από την αποθήκευση οπότε αν οριστεί σε ένα πεδίο μια μη έγκυρη τιμή, θα προκαλέσει σφάλμα όταν γίνει η προσπάθεια να εκτελεστεί η μέθοδος save() για την αποθήκευση του εγγράφου.

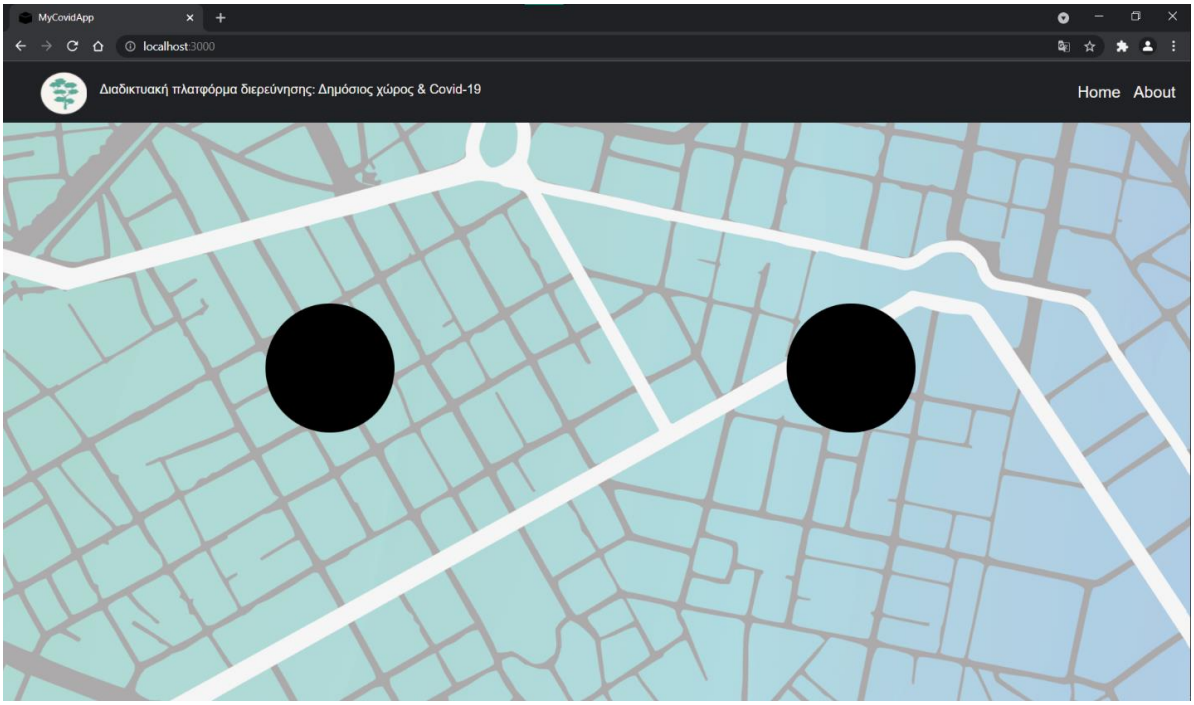
4. ΠΑΡΟΥΣΙΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

4.1 ΠΑΡΟΥΣΙΑΣΗ ΤΩΝ ΣΕΛΙΔΩΝ

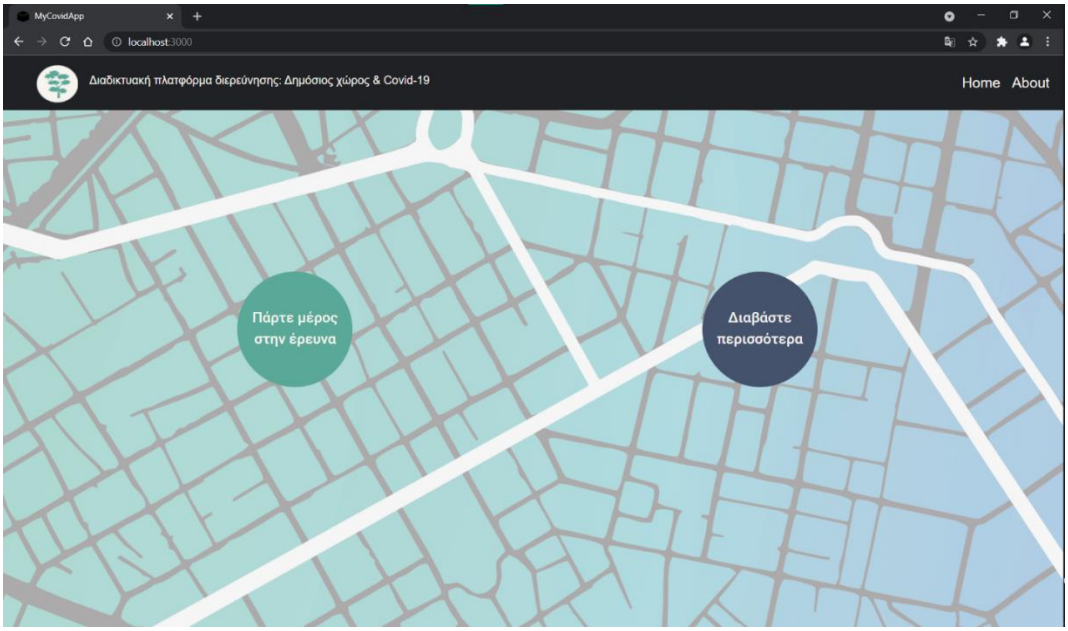
Η εφαρμογή αποτελείται συνολικά από επτά ξεχωριστές σελίδες. Όλες οι σελίδες περιέχουν, εκτός των άλλων, μια μπάρα πλοήγησης η οποία αποτελείται από μια μικρή εικόνα, σαν logo, πάνω αριστερά και δύο συνδέσμους. Οι δύο σύνδεσμοι ανακατευθύνουν τον χρήστη είτε στην αρχική σελίδα, εάν ο χρήστης έχει πλοηγηθεί σε κάποια άλλη σελίδα είτε στην About, δηλαδή τη σελίδα που περιγράφει με λίγα λόγια το θέμα της διπλωματικής.

Αρχική σελίδα:

Η αρχική σελίδα αποτελείται από την εικόνα ενός χάρτη επεξεργασμένη στο Photoshop, τη μπάρα πλοήγησης και δύο κουμπιά στο κέντρο. Η εικόνα είναι αποτύπωση ενός κεντρικού σημείου της Αθήνας γύρω από την Ομόνοια. Το αριστερό κουμπί οδηγεί στις σελίδες με το ερωτηματολόγιο και τον χάρτη και το δεξιά στην σελίδα με τα άρθρα και σχετική βιβλιογραφία για όποιον ενδιαφέρεται να διαβάσει περισσότερα για το ευρύτερο θέμα.



Εικόνα 4.1 Αρχική σελίδα πριν περάσει ο κέρσορας πάνω από τα κουμπιά



Εικόνα 4.2 Αρχική σελίδα αφού περάσει ο κέρσορας πάνω από τα κουμπιά

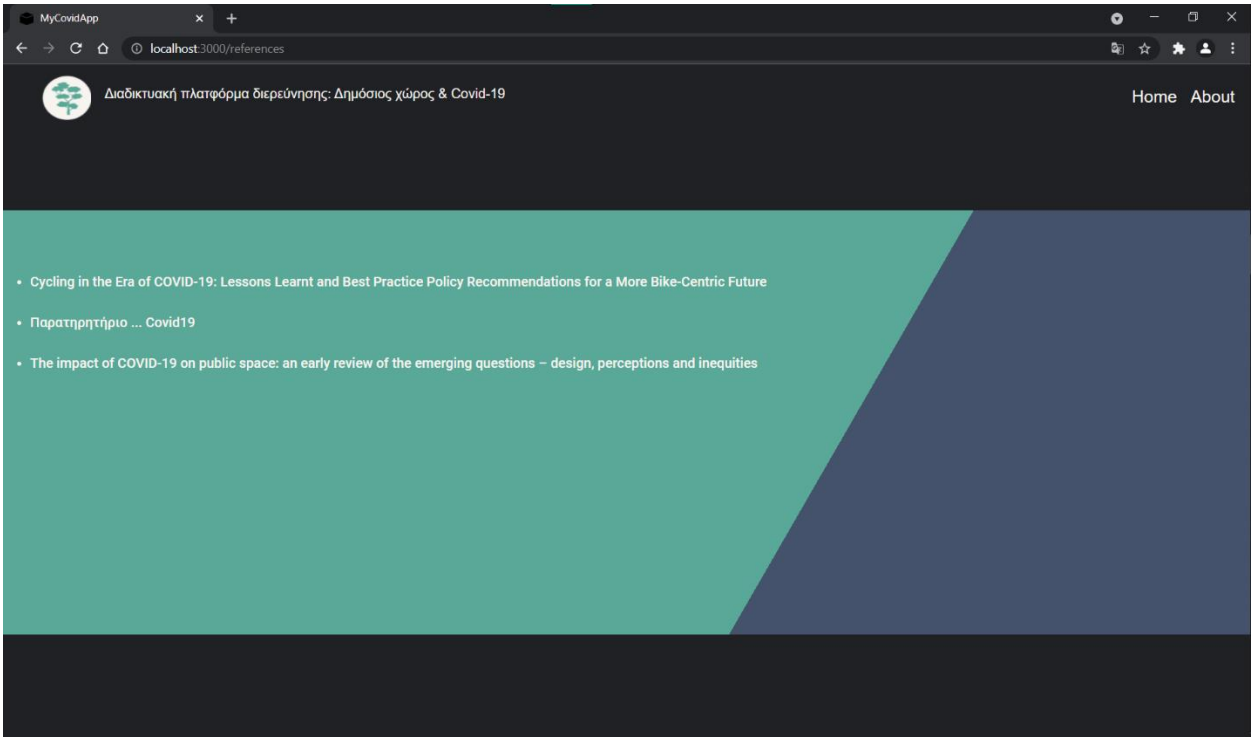
Σελίδα About:

Η σελίδα About περιλαμβάνει τον τίτλο της διπλωματικής, λίγα λόγια για το θέμα, το όνομα της συγγραφέα, το όνομα του επιβλέποντα καθηγητή το οποίο είναι και σύνδεσμος για την ιστοσελίδα του, καθώς και το όνομα της σχολής που επίσης είναι σύνδεσμος για την ιστοσελίδα της.

Εικόνα 4.3 Σελίδα About

Σελίδα Βιβλιογραφίας:

Η σελίδα αυτή περιέχει άρθρα σχετικά με τον δημόσιο χώρο και τον Covid-19, αλλά και άρθρα γενικότερα για τον δημόσιο χώρο, τη χρήση του στα αστικά κέντρα και συγκρίσεις για την Αθήνα και άλλες μεγάλες πόλεις της Ευρώπης.

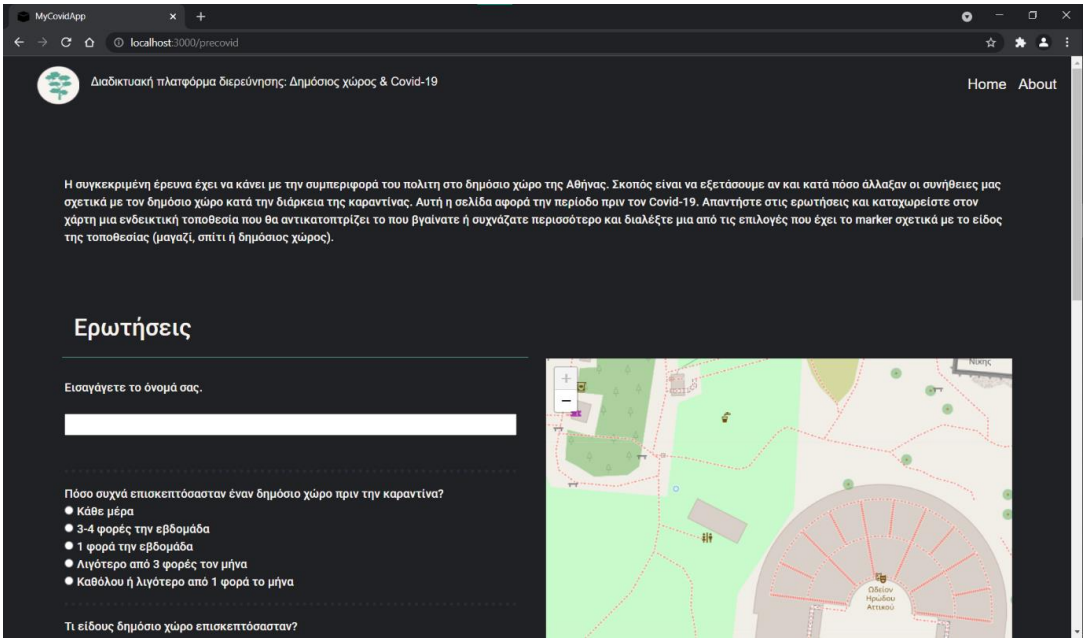


Εικόνα 4.4 Αναφορές & πηγές για περαιτέρω εμβάθυνση

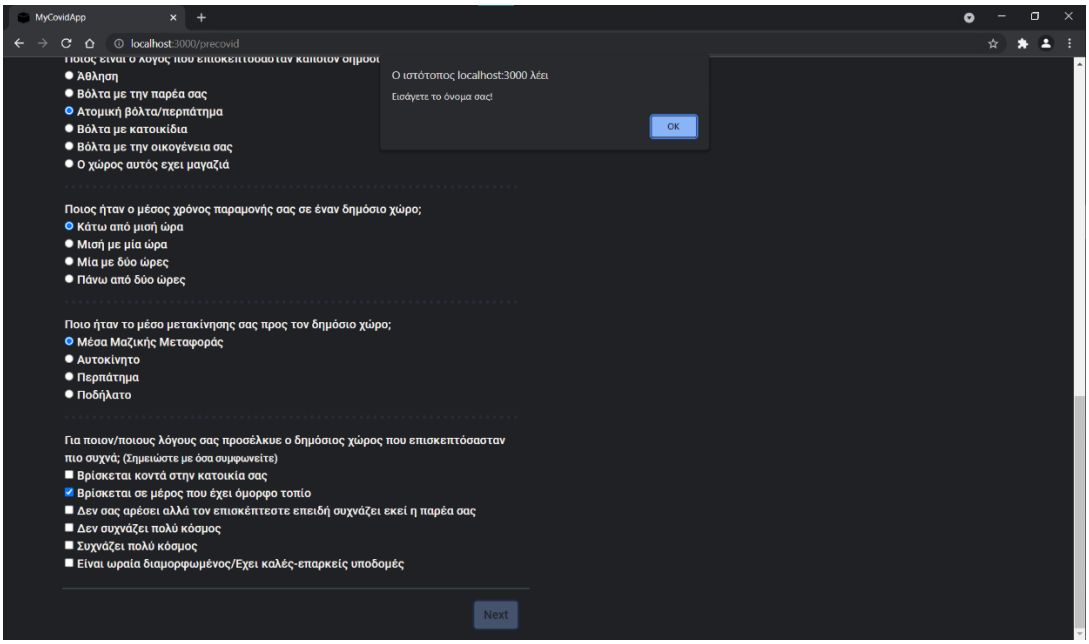
Σελίδα ερωτηματολογίου πριν τον Covid-19:

Το κύριο μέρος της σελίδας αυτής αποτελείται από το ερωτηματολόγιο και τον χάρτη. Αρχικά, ο χρήστης μπορεί να διαβάσει τις οδηγίες που υπάρχουν στο πάνω μέρος της σελίδας και να καταχωρήσει το όνομα του, σε περίπτωση που δεν το καταχωρήσει δεν μπορεί να συνεχίσει στην επόμενη σελίδα. Στη συνέχεια, θα πρέπει να απαντήσει στις ερωτήσεις και να καταχωρήσει στον χάρτη μια ενδεικτική τοποθεσία επιλέγοντας στον αναδυόμενο δείκτη και τον τύπο της τοποθεσίας (μαγαζί, σπίτι ή δημόσιος χώρος). Εάν δεν έχει απαντήσει σε κάποια ερώτηση ή δεν έχει καταχωρήσει τοποθεσία δεν μπορεί να συνεχίσει στην επόμενη σελίδα. Εφόσον ολοκληρώσει όλα

τα παραπάνω τότε μπορεί να πατήσει το “Next” και να οδηγηθεί στην επόμενη σελίδα που αφορά την περίοδο κατά τον Covid-19.



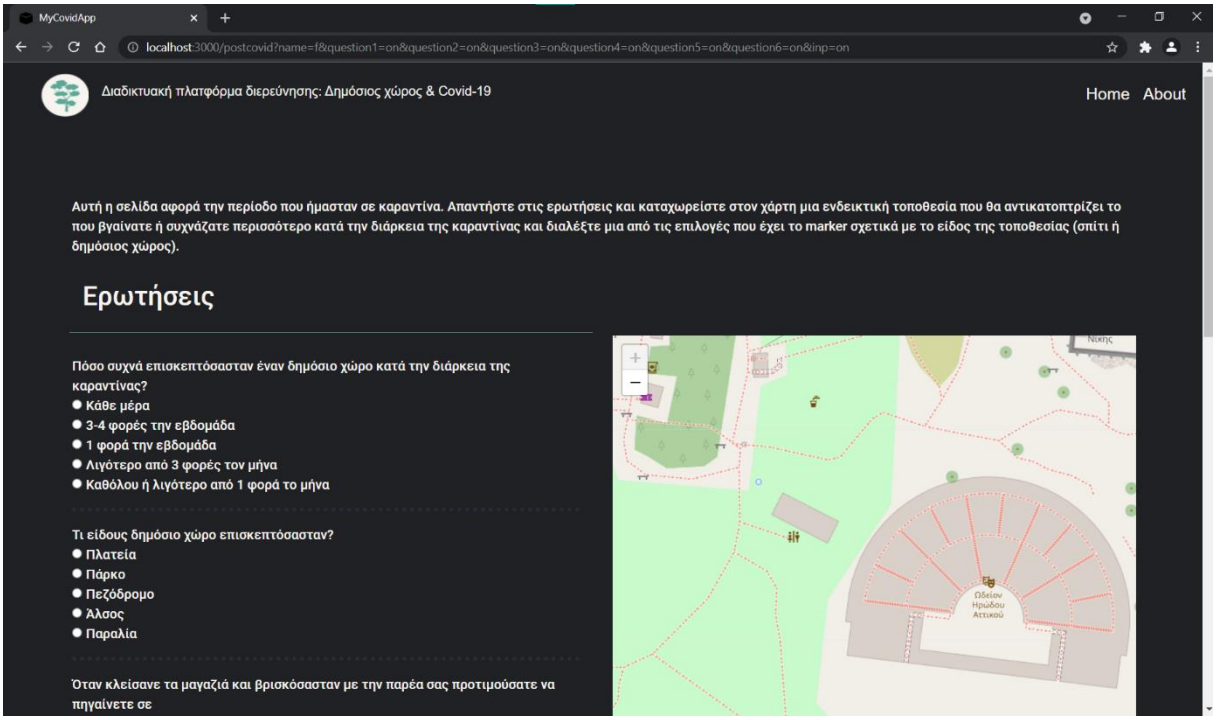
Εικόνα 4.5 Ερωτηματολόγιο πριν τον Covid-19



Εικόνα 4.6 Alert εάν είναι κενό το όνομα χρήστη

Σελίδα ερωτηματολογίου για την περίοδο του Covid-19:

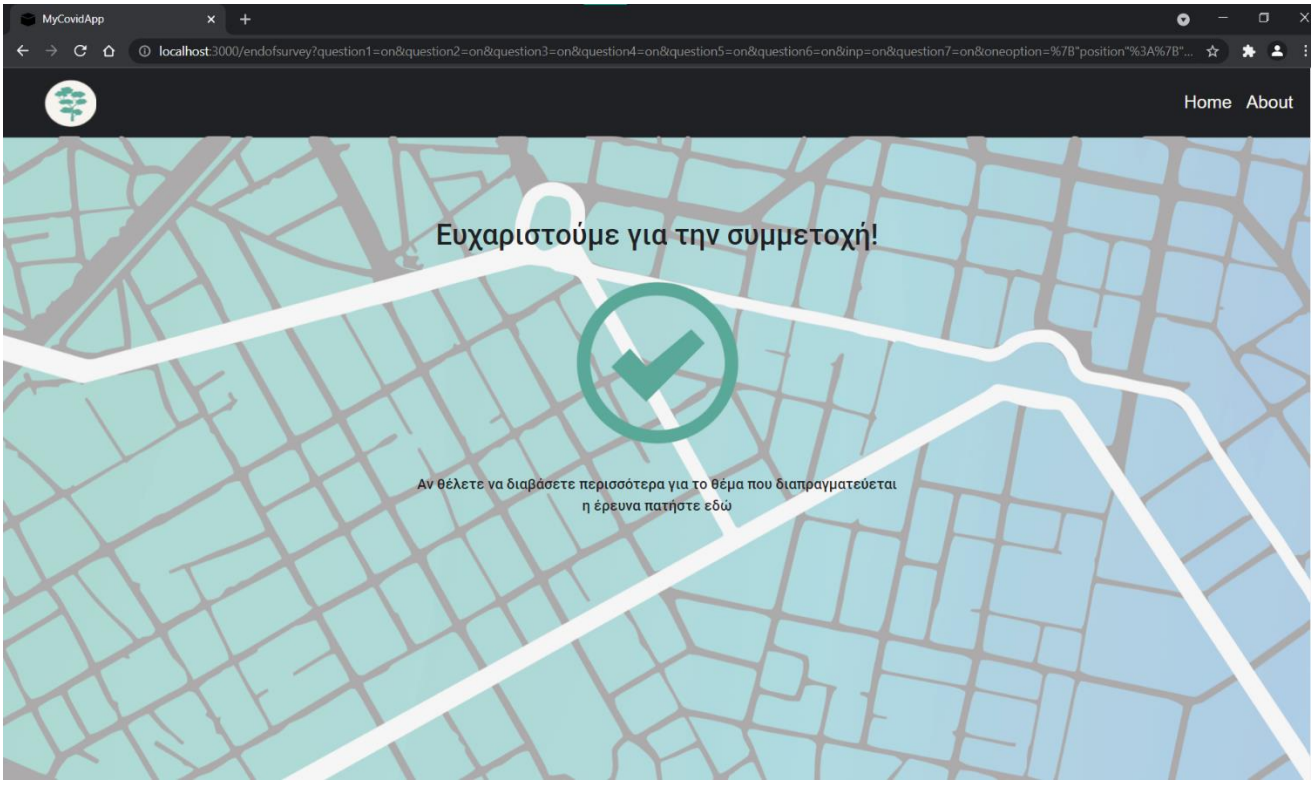
Η σελίδα αυτή περιέχει επίσης ένα ερωτηματολόγιο με αντίστοιχες ερωτήσεις προσαρμοσμένες στις συνθήκες της καραντίνας. Ο χρήστης, πάλι θα πρέπει να απαντήσει στις ερωτήσεις και να καταχωρήσει μια ενδεικτική τοποθεσία στον χάρτη, επιλέγοντας το είδος της τοποθεσίας. Όταν ολοκληρώσει τα παραπάνω πατάει το κουμπί «Next» που τον οδηγεί στη σελίδα που του ανακοινώνει ότι τελείωσε η έρευνα.



Εικόνα 4.7 Ερωτηματολόγιο για την περίοδο κατά την διάρκεια Covid-19

Σελίδα «τέλος της έρευνας»:

Η σελίδα αυτή ανακοινώνει στον χρήστη ότι τελείωσε το κομμάτι της έρευνας και ότι αν θελήσει μπορεί να διαβάσει περισσότερο για το θέμα της διπλωματικής, άρθρα που έχουν γραφτεί, έρευνες που έχουν γίνει γενικότερα πάνω στο θέμα του δημόσιου χώρου και της χρήσης του. Διαφορετικά ο χρήστης μπορεί από τη μπάρα πλοήγησης να ξαναπάει στην αρχική σελίδα ή στη σελίδα About.



Εικόνα 4.8 Τέλος της έρευνας

Σελίδα σφάλματος (error):

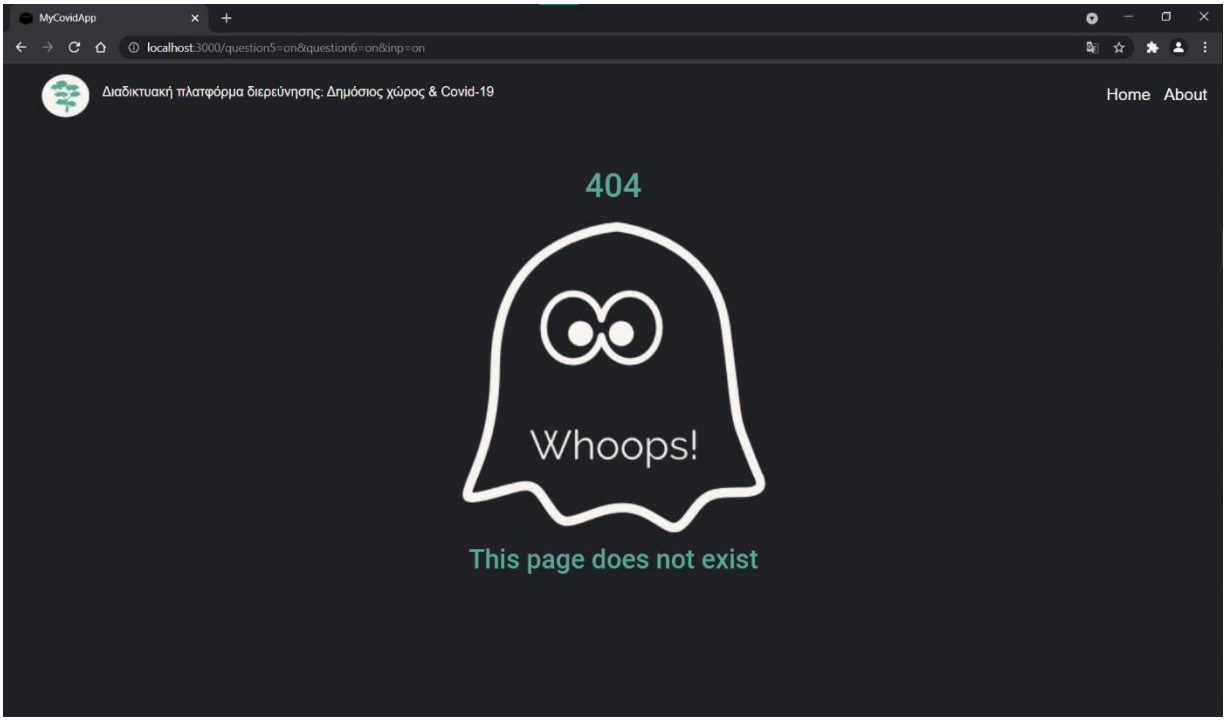
Η τελευταία σελίδα είναι η σελίδα error, δηλαδή η σελίδα που εμφανίζεται όταν ο χρήστης πληκτρολογήσει λάθος τη διεύθυνση ιστοσελίδας. Η αρχική σελίδα της εφαρμογής εμφανίζεται πληκτρολογώντας localhost:3000 και οι υπόλοιπες προσθέτοντας μετά από μία "/" το όνομα που τους αντιστοιχεί (το έχει καθορίσει ο προγραμματιστής). Για παράδειγμα, στη συγκεκριμένη εφαρμογή για να βρεθεί ο χρήστης στην σελίδα About, εκτός από το να πατήσει τον σύνδεσμο στη μπάρα πλοήγησης, μπορεί να πληκτρολογήσει στη μπάρα διεύθυνσης «localhost:3000/about». Σε περίπτωση που ο χρήστης πληκτρολογήσει κάτι λάθος, όπως localhost:3000/abot θα εμφανιστεί η παρακάτω ιστοσελίδα που τον ενημερώνει ότι αυτή η διεύθυνση δεν υπάρχει. Αυτό είναι το σφάλμα 404, δηλαδή ο διακομιστής δεν βρίσκει σελίδα με τέτοια διεύθυνση οπότε επιστρέφει το συγκεκριμένο σφάλμα.

Σε αυτό το σημείο θα αναφερθούμε σύντομα στους κωδικούς απόκρισης κατάστασης HTTP (HTTP response status codes). Οι κωδικοί απόκρισης κατάστασης απόκρισης HTTP υποδεικνύουν εάν ένα συγκεκριμένο αίτημα HTTP έχει ολοκληρωθεί με επιτυχία. Οι απαντήσεις κωδικοποιούνται με έναν αριθμό και ομαδοποιούνται στις εξής πέντε τάξεις:

1. Ενημερωτικές απαντήσεις (Informational responses) (100-199)
2. Απαντήσεις επιτυχίας (Successful responses) (200-299)
3. Ανακατευθύνσεις (Redirects) (300-399)
4. Σφάλματα πελάτη (Client errors) (400-499)
5. Σφάλματα διακομιστή (Server errors) (500-599)

Οι πιο συχνές απαντήσεις είναι:

- **200**: Το αίτημα πέτυχε (The request has succeeded)
- **404**: Δεν βρέθηκε (Not found). Ο διακομιστής δεν μπορεί να βρει τον πόρο που ζητήθηκε. Στο πρόγραμμα περιήγησης, αυτό σημαίνει ότι η διεύθυνση URL δεν αναγνωρίζεται. Σε ένα API, αυτό μπορεί επίσης να σημαίνει ότι το τελικό σημείο είναι έγκυρο, αλλά ο ίδιος ο πόρος δεν υπάρχει. Οι διακομιστές μπορούν επίσης να στείλουν αυτήν την απάντηση αντί για 403 για να αποκρύψουν την ύπαρξη ενός πόρου από μη εξουσιοδοτημένο πρόγραμμα -πελάτη. Αυτός ο κωδικός απόκρισης είναι ίσως ο πιο διάσημος λόγω της συχνής εμφάνισής του στον ιστό.
- **500**: Εσωτερικό σφάλμα διακομιστή (Internal server error). Ο διακομιστής αντιμετώπισε μια κατάσταση που δεν ξέρει πώς να χειριστεί.



Εικόνα 4.9 Σελίδα σφάλματος

4.2 ΑΠΟΚΡΙΣΙΜΟΤΗΤΑ ΓΙΑ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ

Τη σημερινή εποχή που η πλειοψηφία της πρόσβασης στο διαδίκτυο γίνεται μέσω έξυπνων κινητών συσκευών είναι σχεδόν απαραίτητο η οποιαδήποτε ιστοσελίδα ή εφαρμογή να προσαρμόσει τον κώδικα ώστε να μπορούν να προβληθούν και από τις κινητές συσκευές χωρίς να δυσκολεύουν τον χρήστη.
Επειδή υπάρχει πληθώρα κινητών συσκευών, παρακάτω παρατίθενται τα μεγέθη των οθονών σύμφωνα με τα οποία προσαρμόστηκε ο κώδικας.

Τάξη Μεγέθους	Σημείο (Breakpoint)	Είδος συσκευών	Μέγεθος παραθύρου
Μικρή	>640px	κινητά	320x569, 360x640, 480x854
Μεσαία	641-1007px	Tablets	960x540
Μεγάλη	1007px<	Επιτραπέζιοι/φορητοί υπολογιστές	1024x640, 1366x768, 1920x1080

Ο προγραμματιστής επιλέγει το μέγεθος της οθόνης σύμφωνα με την οποία αναπτύσσει την εφαρμογή του και έπειτα μπορεί να προσαρμόσει τον κώδικα εάν επιθυμεί να αλλάξει τον τρόπο που αυτή θα εμφανίζεται σε άλλες συσκευές με διαφορετικό μέγεθος οθόνης. Η προσαρμογή γίνεται μέσω του αρχείου CSS και τα media queries.

```
@media (max-width: 990px) {

…

}
```

Μέσα στο αρχείο .css, κατά προτίμηση στο τέλος του κώδικα, γράφονται οι προσαρμογές. Το συντακτικό προσαρμόζεται ως εξής: μετά το @media σε παρένθεση γράφεται το μέγεθος οθόνης για το οποίο θέλει ο προγραμματιστής να αλλάξει λειτουργικότητα/εμφάνιση ο ήδη γραμμένος κώδικας. Αυτό μπορεί να γίνει με (max-width: _px) για μεγέθη ως την τιμή που θα αναγράφεται, με (min-width:_px), δηλαδή για μεγέθη από την τιμή που αναγράφεται αλλά και με (max-width: 900px) and (min-width: 600px) για καθορισμό συγκεκριμένου διαστήματος τιμών. Για παράδειγμα, στην παρούσα εφαρμογή ο κώδικας αναπτύχθηκε με γνώμονα τις οθόνες

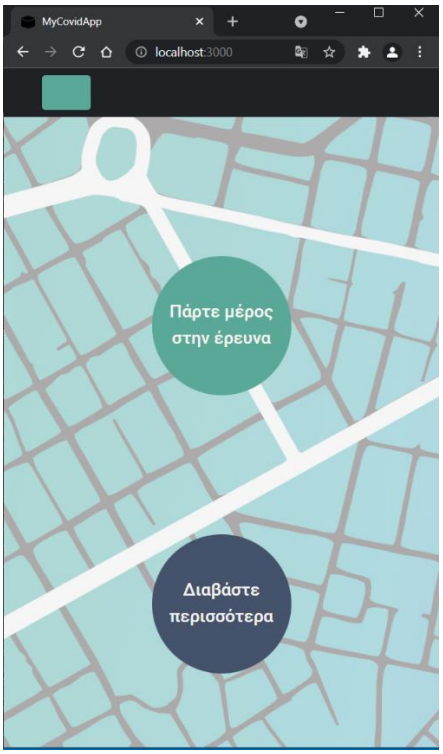
επιτραπέζιων και φορητών υπολογιστών και το μέγεθος για το οποίο έγινε προσαρμογή είναι από τα 960px και κάτω, με κάποιες ακόμα προσαρμογές στο κομμάτι: @media (min-width: 961px) and (max-width: 1366px). Ο κώδικας που γράφεται μέσα στο @media ακολουθεί το συντακτικό και τους κανόνες της CSS και χρησιμοποιώντας τα id και τις class για να στοχεύσει ο προγραμματιστής ακριβώς τι επιθυμεί να αλλάξει.
Στην εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας, ο κώδικας που καθιστά το περιεχόμενο ανταποκρίσιμο είναι ο εξής:

```
/* MEDIA QUERIES */
@media (max-width: 960px) {

  /*HOMEPAGE*/
  .about-div {
    width: 80%;

  }
  #homeButtonOne,
  #homeButtonTwo {
    display: block;
    position: relative;
    margin-left: auto;
    margin-right: auto;
    margin-top: 10rem;
  }

  /*NAVIGATION BAR*/
  .navbar-toggler {
    margin-left: 2rem;
    z-index: 2;

  }
  .navbar-collapse {
    margin-left: 2rem;
  }
  .tree-logo {
    display: none;
  }

  /*PRE&DUR COVID-19 PAGE*/
  .page-container {
    margin-right: 1rem;
    max-width: 100%;
```

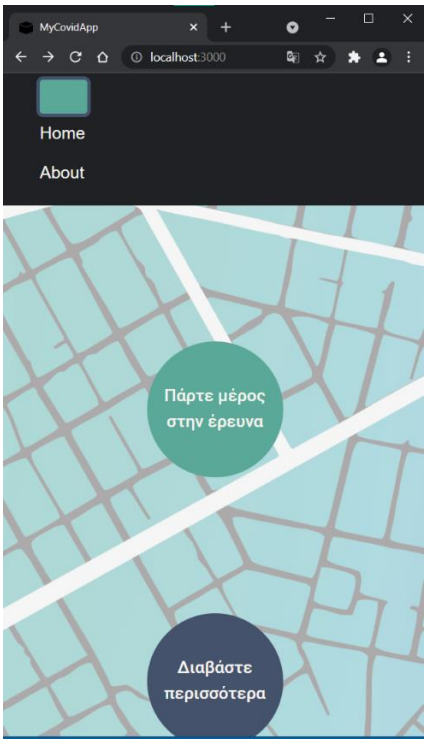
```
  }
  /* END OF SURVEY PAGE */
 .end-container {
   margin-top: 45%;
 }
  /*ERROR PAGE*/
  .error-div {
    margin-top: 20%;
  }
  .ghost-image {
    width: 60%;
  }
}

@media (min-width: 961px) and (max-width: 1366px) {
  .dotButton1,
  .dotButton2,
  .dotButtonBeforeClicked1,
  .dotButtonBeforeClicked2{
    width: 15.5%;
  }
.dotButton1,
.dotButtonBeforeClicked1 {
    margin-left: 23.3%;
  }
.dotButton2,
.dotButtonBeforeClicked2 {
    margin-left: 66.6%;
  }
}
```

Παρακάτω παρατίθενται εικόνες για τη μικρότερη δυνατή ανάλυση στην οθόνη του φορητού υπολογιστή στον οποίο αναπτύχθηκε η εφαρμογή για να αναδειχθούν οι προσαρμογές και η παρουσίαση της εφαρμογής σε κινητές συσκευές:

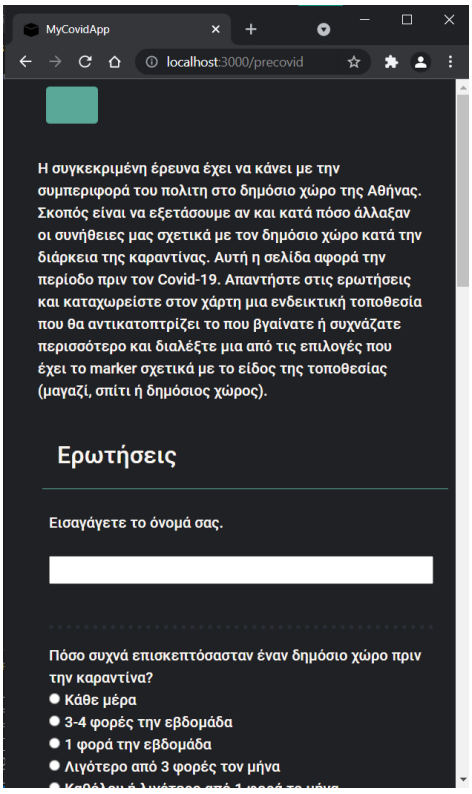
Εικόνα 4.10 Αρχική σελίδα για κινητές συσκευές



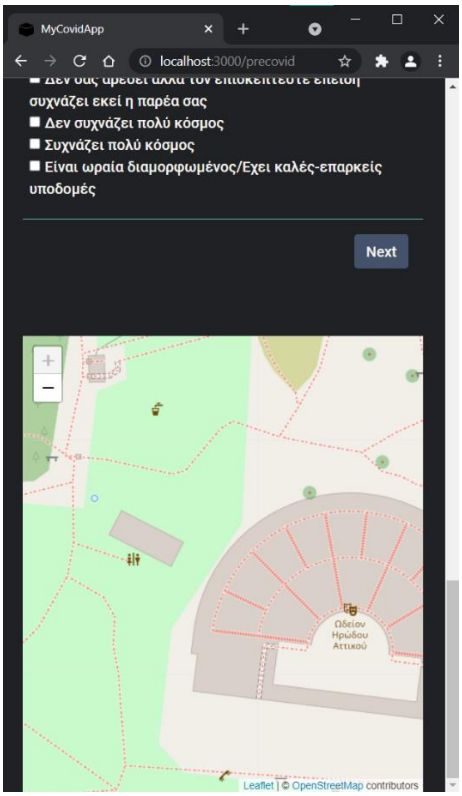
Εικόνα 4.11 Αρχική σελίδα για κινητές συσκευές λειτουργία κουμπιού μπάρας πλοήγησης



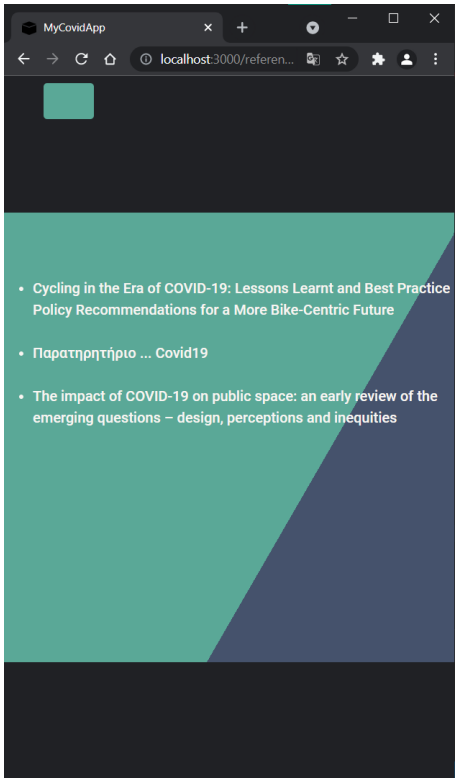
Εικόνα 4.11 Ερωτηματολόγιο προ Covid-19



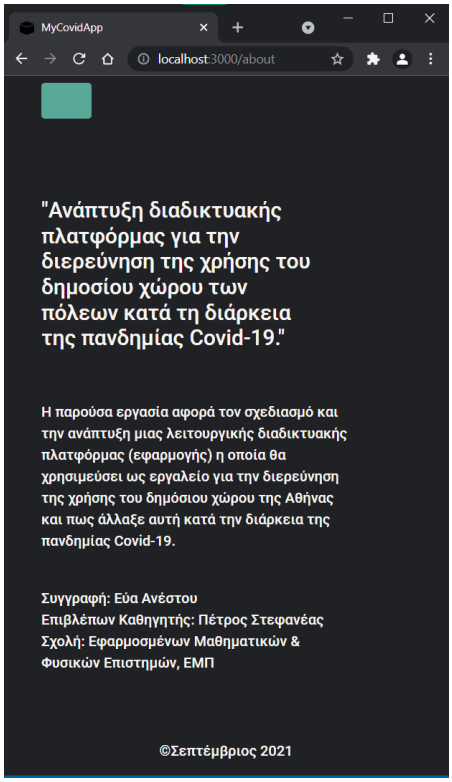
Εικόνα 4.12 Ερωτηματολόγιο προ Covid-19 (χάρτης)



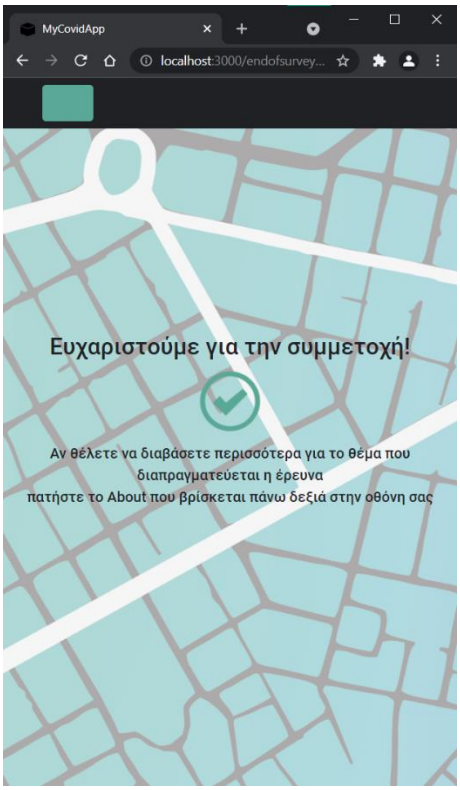
Εικόνα 4.13 Σελίδα πηγών



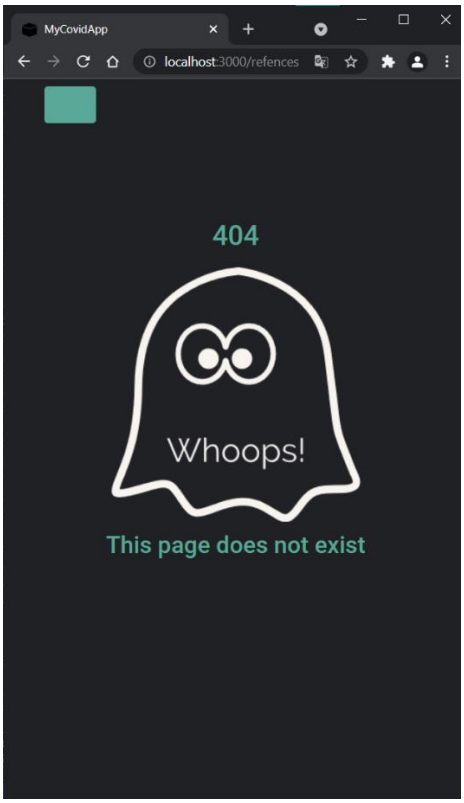
Εικόνα 4.14 Σελίδα About



Εικόνα 4.15 Τέλος έρευνας



Εικόνα 4.16 Σελίδα σφάλματος



4.3 ΜΕΤΑΦΟΡΤΩΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ

Η παρούσα διπλωματική περιλαμβάνει τη διαδικασία ανάπτυξης της εφαρμογής, τις επιλογές και την επεξήγηση των τεχνολογιών που χρησιμοποιήθηκαν και έχει σαν αποτέλεσμα μια λειτουργική εφαρμογή. Παρόλα αυτά, θεωρείται σκόπιμη η αναφορά στα αρχικά στάδια της μεταφόρτωσης της εφαρμογής στο διαδίκτυο.

Η παρούσα εφαρμογή λειτουργεί σε τοπικό επίπεδο, όπως εξηγήθηκε και στο κεφάλαιο localhost, μεταβαίνοντας στο localhost βλέπουμε τις σελίδες της εφαρμογής, χρησιμοποιούμε τον υπολογιστή του προγραμματιστή σαν διακομιστή για να διαχειριστεί την εφαρμογή και τα δεδομένα της. Αντίστοιχα, για τη βάση δεδομένων, ο υπολογιστής του προγραμματιστή λειτουργεί σαν διακομιστής, εφόσον είναι και αυτή συνδεδεμένη με τον localhost, όπως φαίνεται και στον κώδικα παρακάτω:

```
mongoose.connect("mongodb://localhost:27017/myCovidApp", { useNewUrlParser: true
,useUnifiedTopology: true})
```

Με λίγα λόγια η εφαρμογή «δεν υπάρχει έξω από τον συγκεκριμένο υπολογιστή».

Ένα βασικό ζήτημα στην διαδικασία της μεταφόρτωσης είναι αυτό του τομέα που θα φιλοξενήσει την ιστοσελίδα/εφαρμογή. Υπάρχουν πλατφόρμες στο διαδίκτυο που παρέχουν δωρεάν τομέα για συγκεκριμένο αριθμό ιστοσελίδων, για παράδειγμα μια δημοφιλής πλατφόρμα είναι το Heroku. Το επόμενο βήμα είναι οι μικρές αλλαγές που πρέπει να γίνουν για να μπορεί να μεταφορτωθεί και η βάση δεδομένων. Σε αυτό παίζει τον κύριο ρόλο το MongoDB Atlas που έχει περιγραφεί στο κεφάλαιο της MongoDB. Φτιάχνοντας λογαριασμό, ο προγραμματιστής αποκτά πρόσβαση σε μια συστάδα (cluster), το οποίο υπάρχει στο νέφος (cloud). Μέσα στο ίδιο το cluster, υπάρχει ενότητα Ασφάλεια (Security) που είναι ένα ακόμη πολύ σημαντικό ζήτημα. Εκεί, ο προγραμματιστής προσθέτει χρήστη (με την έννοια του διαχειριστή, επιλέγει να είναι διαχειριστής μέσω της επιλογής Atlas Admin) με ένα όνομα και έναν κωδικό. Όσο πιο ισχυρός είναι ο κωδικός τόσο πιο προστατευμένο είναι το cluster του χρήστη και άρα και οι βάσεις δεδομένων που φιλοξενούνται στο συγκεκριμένο cluster.

Έπειτα, στην ενότητα «Προσθήκη IP διεύθυνσης» (Add an IP address) πρέπει να επιλεγεί το κομμάτι που επιτρέπει την πρόσβαση από οπουδήποτε «Allow access from anywhere». Στη συνέχεια, ο χρήστης μπορεί να δημιουργήσει οποιαδήποτε βάση θελήσει μέσω της διεπαφής του Atlas.

Τέλος, πρέπει να γίνει η προσαρμογή στον κώδικα και η σύνδεση με τη βάση δεδομένων. Πρώτα

γίνεται η σύνδεση με το cluster στην ενότητα που λέει «Σύνδεση» και έπειτα «Σύνδεση με τον φλοιό του MongoDB». Επιλέγοντας τα προηγούμενα το Atlas παρέχει στον χρήστη έναν ομοιόμορφο εντοπιστή πόρων (URL), τον οποίο μπορεί να εισάγει στη γραμμή εντολής στον φλοιό του MongoDB. Προχωρώντας, θα πρέπει να εισάγει το όνομα και τον κωδικό που είχε καθορίσει για την ασφάλεια του cluster και η σύνδεση ολοκληρώνεται. Τώρα ο χρήστης έχει ενεργή συνεδρία με τον φλοιό του MongoDB να «τρέχει» στο σύμπλεγμα (cluster) Atlas. Το επόμενο βήμα είναι η σύνδεση μέσω της εφαρμογής. Το κομμάτι του κώδικα που μέχρι πρότινος συνέδεε την εφαρμογή με τη βάση δεδομένων:

```
mongoose.connect("mongodb://localhost:27017/myCovidApp", { useNewUrlParser: true
,useUnifiedTopology: true})
```

πλέον δεν ισχύει, διότι θα πρέπει να υπάρχει σύνδεση με τη βάση δεδομένων που δημιουργήθηκε στο Atlas και όχι στο localhost. Στην ενότητα «Σύνδεση» υπάρχει και η επιλογή «Σύνδεση με την εφαρμογή σας» και έπειτα ο χρήστης επιλέγει την «Σύνδεση SRV» και αντιγράφει τη διεύθυνση με την οποία τον προμηθεύει το Atlas. Προσαρμόζει τη διεύθυνση όπου χρειάζεται, καθώς ζητάει το όνομα και τον κωδικό του χρήστη καθώς και το όνομα της βάσης δεδομένων και το επικολλάει στο κομμάτι κώδικα που φαίνεται παραπάνω, μέσα στην παρένθεση στη θέση του “mongodb://localhost:27017/MyCovidApp”. Με αυτό τον τρόπο εγκαθιδρύεται η σύνδεση με την εφαρμογή.

Τα επόμενα βήματα για τη μεταφόρτωση στο διαδίκτυο πρέπει να γίνουν μέσω Heroku, εάν επιλεγεί η συγκεκριμένη πλατφόρμα. Το Heroku είναι μια πλατφόρμα cloud που επιτρέπει στις εταιρείες (και όχι μόνο) να δημιουργούν, να παραδίδουν, να παρακολουθούν και να κλιμακώνουν εφαρμογές.

Το Heroku είναι γνωστό για την εκτέλεση εφαρμογών σε dynos - οι οποίοι είναι μόνο εικονικοί υπολογιστές που μπορούν να τροφοδοτηθούν παραπάνω ή λιγότερο ανάλογα με το μέγεθος και τις απαιτήσεις της εφαρμογής που θα φιλοξενήσει. Αφού δημιουργήσει λογαριασμό ο χρήστης τότε υπάρχει λεπτομερής οδηγός για την εγκατάσταση του πακέτου εντολών του Heroku (ώστε να εκτελούνται εντολές μέσω του φλοιού) καθώς και όλων των βημάτων που πρέπει να ακολουθήσει ο χρήστης για να γίνει σωστά η μεταφόρτωση της εφαρμογής του, στα οποία βήματα δεν θα αναλωθεί η παρούσα εργασία.

5. ΣΥΜΠΕΡΑΣΜΑΤΑ

5.1 ΘΕΜΑ ΕΡΕΥΝΑΣ

Σε αυτό το κεφάλαιο θα γίνει αναφορά, στο επίπεδο που αναλογεί στην παρούσα πτυχιακή εργασία, στο θέμα με το οποίο ασχολείται η έρευνα της εφαρμογής. Εφόσον λείπουν ακαδημαϊκές γνώσεις σχετικές με τη διαχείριση και τη χρήση των δημοσίων χώρων, το θέμα προέκυψε κυρίως για δύο λόγους: το ενδιαφέρον για τους χώρους πρασίνου και την παρατήρηση μιας αλλαγής στην συμπεριφορά του κόσμου ως προς τη συχνότητα της επίσκεψης τους σε αυτούς τους χώρους. Βλέποντας τους ήδη υπάρχοντες χώρους πρασίνου στις γειτονιές της Αθήνας μπορεί να παρατηρηθεί ότι πολλοί από αυτούς είναι εγκαταλελειμμένοι, γεγονός το οποίο πολλές φορές αποτρέπει την επίσκεψη και την παραμονή στους συγκεκριμένους χώρους. Ενδεικτικά, στην περιοχή Υμηττού-Βύρωνα-Παγκρατίου μπορούν να αναφερθούν το πάρκο Αλεξάνδρου Άρη, η πλατεία Παπαφλέσσα και το Άλσος Παγκρατίου. Βλέποντας αυτή την αλλαγή που προέκυψε λόγω της αναγκαιότητας για επαφή, αίσθηση ελευθερίας και εξόδου από το σπίτι, προέκυψε η ιδέα για το θέμα της εφαρμογής ώστε να μπορεί να αναδειχθεί σε δεύτερο χρόνο η αξιοποίηση, η πρόσβαση και ο ανοικτός χαρακτήρας των δημόσιων χώρων. Αυτό το θέμα πολλές φορές παραγκωνίζεται από κατοίκους μεγάλων αστικών κέντρων λόγω του γρήγορου ρυθμού ζωής και της αποξένωσης από χώρους πρασίνου. Για να διερευνηθεί εάν τελικά υπήρξε αλλαγή και σε ποιο βαθμό, αναπτύχθηκαν οι συγκεκριμένες ερωτήσεις τις οποίες καλείται να απαντήσει ο χρήστης. Ενδιαφέρουν έχουν παράμετροι όπως το πόσο συχνά πήγαινε κάποιος πριν τον Covid-19 σε δημόσιους χώρους (συχνότητα ανά εβδομάδα/μήνα), τι διάρκεια είχε η επίσκεψη του (μισή ώρα, μία ώρα κ.α.), ποιος ήταν ο σκοπός της (περπάτημα, βόλτα με κατοικίδιο, συνάντηση με την παρέα κ.α.), πως έφτασε εκεί(με μέσα, με αυτοκίνητο κ.α.) κλπ. Αντίστοιχα διαμορφώθηκαν οι ερωτήσεις για την περίοδο της καραντίνας όπου μπορεί να παρατηρηθεί εκτός από τη συχνότητα, αν υπάρχουν αλλαγές στον σκοπό της επίσκεψης (π.χ. αν έγινε κυρίως χώρος κοινωνικής συναναστροφής), αλλά και ο τρόπος πρόσβασης, δηλαδή αν περιορίστηκαν οι μετακινήσεις περισσότερο στις γειτονιές ή αν προτιμήθηκε το περπάτημα προς αποφυγή των μέσων μαζικής μεταφοράς ή σαν διέξοδος από την πολύωρη παραμονή στο σπίτι.

Οι ιδέες για το πως θα μπορούσε να αξιοποιηθεί μια τέτοια εφαρμογή είναι πολλές. Στην παρούσα μορφή, θα μπορούσε να είναι μια αφορμή για να ξεκινήσει διάλογος για το πώς θα μπορούσαν να αξιοποιηθούν περισσότερο από τους πολίτες αυτοί οι πράσινοι χώροι και με ποιο τρόπο θα

μπορούσαν να διεκδικήσουν περισσότερους τέτοιους χώρους ή καλύτερη διατήρηση των ήδη υπαρχόντων. Θεωρητικά, η πλατφόρμα αυτή θα μπορούσε να εξελιχθεί παραπάνω και να επεκταθεί και για κατοίκους άλλων αστικών κέντρων στην Ευρώπη. Με αυτό τον τρόπο είναι δυνατές οι συγκρίσεις αναλογικά με τον πληθυσμό και το πλήθος δημόσιων και πράσινων χώρων, όπως και οι συγκρίσεις για το πόσο χρησιμοποιεί ο μέσος πολίτης τους δημοσίους χώρους που υπάρχουν στην πόλη του, για παράδειγμα στην Αθήνα σε σχέση με το Λονδίνο.

Με μια λίγο διαφορετική κατεύθυνση και προσθήκη ανάλογων ερωτήσεων θα μπορούσε να χρησιμοποιηθεί από κρατικούς φορείς ώστε να αναδειχθούν οι ελλείψεις και οι επιθυμίες των πολιτών σε σχέση με τους δημόσιους χώρους στις γειτονιές τους (π.χ. ανά Δήμο στην Αττική). Επίσης, θα μπορούσε η ίδια η πλατφόρμα να εμπλουτιστεί περισσότερο, ώστε να έχει τη δυνατότητα ο χρήστης να καταχωρήσει περισσότερες πληροφορίες στον χάρτη, ανάλογα πάντα και με τον τρόπο που θέλει να τη χρησιμοποιήσει ο υπεύθυνος της έρευνας και της ανάλυσης δεδομένων.

Τέλος, σε μια τελείως διαφορετική προσέγγιση στις ερωτήσεις και στο θέμα, κρατώντας το κύριο σώμα της εφαρμογής (ως τεχνολογίες και λειτουργικότητα), μπορεί να εξελιχθεί σε έρευνα που ασχολείται με το πόσο επηρέασε ο εγκλεισμός ανάλογα με τον χώρο που είχε διαθέσιμο ο κάθε πολίτης, αν ζούσε με την οικογένεια του ή μόνος του, αν είχε δικό του δωμάτιο ή όχι, αν εκμίσθωνε ή είχε δικό του σπίτι και κατά πόσο σχετίστηκε αυτό με τη συχνότητα επίσκεψης του στους δημόσιους χώρους. Ο χάρτης σε αυτή την περίπτωση θα μπορούσε να εξυπηρετεί και στην παρατήρηση του μέγεθος του χώρου (σπίτι) που αναλογεί στους πολίτες κατά μέσο όρο ανάλογα με την περιοχή της Αθήνας.

5.2 ΠΕΡΙΟΡΙΣΜΟΙ ΚΑΙ ΒΕΛΤΙΩΣΕΙΣ

Σε αυτό το κεφάλαιο θα γίνει αναφορά στις δυσκολίες που αντιμετωπίστηκαν, πώς και εάν προσπεράστηκαν και βελτιώσεις που τυχόν μπορούν να γίνουν μελλοντικά. Αρχικά, η διαδικασία ανάπτυξης μια εφαρμογής ή μιας δυναμικής ιστοσελίδας είναι πολύ πιο περίπλοκη από την αίσθηση που μπορεί να δίνεται σε κάποιον μη εξοικειωμένο με το αντικείμενο.

Οι τεχνολογίες που χρησιμοποιούνται είναι πολλές και πρέπει ο προγραμματιστής ιστού να ξέρει ποια είναι η θέση της καθεμίας και με ποιον τρόπο αλληλοσυμπληρώνονται. Οι γλώσσες προγραμματισμού που χρησιμοποιήθηκαν ίσως να μην είναι ιδιαίτερα περίπλοκες, αλλά ο προγραμματιστής χρειάζεται αρκετό χρόνο για να τις υλοποιήσει σωστά αντιληφθεί το κομμάτι της λειτουργικότητας που τους αντιστοιχεί.

Η κυριότερη δυσκολία ήταν η λήψη απόφασης για το είδος της διαδικασία που θα ακολουθούταν για την ανάπτυξη της εφαρμογής. Υπήρχαν δύο ενδεχόμενα, το πρώτο ήταν να γίνει η υλοποίηση μέσω Wordpress και η δεύτερη εξολοκλήρου με κώδικα. Μέσω συζητήσεων με έμπειρους επαγγελματίες του κλάδου υπήρχαν πολλές προτροπές να ακολουθηθεί η οδός του Wordpress. Προσπαθώντας η απόφαση που θα παρθεί να είναι η καλύτερη δυνατή σκεπτόμενη και τις γνώσεις που θα αποκτιούνταν από αυτή την διαδικασία, ο δρόμος που αποφασίστηκε να ακολουθηθεί ήταν αυτός της εξολοκλήρου ανάπτυξης του κώδικα καθότι με αυτόν τον τρόπο θα γινόταν καλύτερα κατανοητή η λειτουργία μιας εφαρμογής, από το αισθητικό και σχεδιαστικό της κομμάτι μέχρι την αποθήκευση των δεδομένων στη βάση.

Η δεύτερη μεγάλη δυσκολία που αντιμετωπίστηκε ήταν η μετάφραση των όρων της πληροφορικής στα ελληνικά. Εκτός από την έλλειψη πηγών σχετικών με τον προγραμματισμό ιστού, σε πολλές περιπτώσεις δεν υπάρχει κατάλληλο λεξιλόγιο που να μεταφράζει ακριβώς την αγγλική λέξη στα ελληνικά. Διαβάζοντας όλες τις πηγές στα αγγλικά, έπρεπε να κατανοηθεί το περιεχόμενο και στη συνέχεια να γίνει προσπάθεια μετάφρασης αλλάζοντας τους όρους στα ελληνικά χωρίς να χαθεί το νόημα και με τέτοιο τρόπο ώστε η μετάφραση να γίνεται κατανοητή από αναγνώστες χωρίς γνώσεις για το αντικείμενο. Σε αυτή τη διαδικασία βοήθησε πολύ η προσπάθεια που έχει γίνει στην σελίδα της Βικιπαίδεια να «μαζευτούν» πολλές ορολογίες ώστε να υπάρξει μια κοινή αφετηρία για μετάφραση που θα χρησιμοποιείται ευρύτερα. Για μεγαλύτερη κατανόηση δημιουργήθηκε το γλωσσάρι στην αρχή της παρούσας εργασίας

Οι βελτιώσεις που θα μπορούσαν να γίνουν στην εφαρμογή είναι οι εξής: Αρχικά, θα μπορούσαν πολλά κομμάτια της εφαρμογής να γίνουν περισσότερο διαδραστικά. Παραδείγματος χάριν, οι σελίδες με το ερωτηματολόγιο θα μπορούσαν να είναι λιγότερο στατικές και οι ερωτήσεις να εμφανίζονται π.χ. με αναδυόμενα παράθυρα. Επίσης, θα μπορούσε ο χάρτης να είναι μεγαλύτερος και να αποτελεί το κύριο μέρος της έρευνας, με τα αναδυόμενα παράθυρα που προαναφέρθηκαν να αναδύονται από τον χάρτη. Τέλος, μια πρόταση για το αισθητικό κομμάτι της εφαρμογής είναι ότι θα μπορούσε αντί για alert να υλοποιηθεί η προειδοποίηση προς τον χρήστη με modals (αναδυόμενα παράθυρα που επιδέχονται μορφοποίηση και υλοποιούνται από JavaScript) ώστε να μπορούν να επιδεχθούν μορφοποίηση για ένα πιο όμορφο αποτέλεσμα.

Όσον αφορά το λειτουργικό κομμάτι της εφαρμογής, μια σημαντική βελτίωση θα ήταν η δυνατότητα ο χρήστης να εισάγει περισσότερες από μια τοποθεσίες στον χάρτη. Η αποθήκευση των δεδομένων θα μπορούσε να παραμείνει και με τον ήδη υπάρχοντα τρόπο ή να αλλάξουν οι κωδικοποιήσεις των ερωτήσεων και των απαντήσεων στη διαδικασία αποθήκευσης ώστε να είναι πιο καθαρή η συνολική εικόνα τους. Μια πρόταση για την οπτικοποίηση των δεδομένων της έρευνας είναι η προσθήκη μιας σελίδας με στατιστικά στοιχεία από τις απαντήσεις των χρηστών π.χ. γραφήματα. Αυτή η δυνατότητα γίνεται πιο εύκολα υλοποιήσιμη εάν η εφαρμογή μεταφορτωθεί στο διαδίκτυο μέσω του MongoDB Atlas και την δυνατότητα που δίνεται μέσω των Graphs που αναφέρθηκαν στο κεφάλαιο 2.11.

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΚΕΦΑΛΑΙΟ 2

2.1

- Επίσημη Ιστοσελίδα Bootstrap
- HTML For Beginners The Easy Way: Start Learning HTML & CSS Today, David Moraes
- w3.org HTML & CSS
- Wikipedia – CSS
- Wikipedia - HTML

2.2

- Βικιπαίδεια - JavaScript
- A Brief History Of JavaScript, Sebastian Peyrott, 2017
- Wikipedia – Javascript

2.3

- Introducing JSON, json.org
- Wikipedia – JSON

2.4

- Επίσημη ιστοσελίδα Nodejs, About Nodejs
- Επίσημη ιστοσελίδα Nodejs, Introduction to NodeJs
- Package.json VS package-lock.json, DLT Labs, 2019
- Wikipedia – Node.js

2.5

2.5.1

- Επίσημη ιστοσελίδα της expressjs

2.5.2

- What is the MVC, Creating a [Node.js – Express] MVC Application, Islem Maboud, 2020

2.6

- Minification of CSS files, geeksforgeeks, Nayonika, 2019

2.7

- Getting Started with Gulp.js, Aleksandar Olic, 2020
- Introduction to Gulp, developers.google

2.8

- Wikipedia - localhost

2.9

- Επίσημη ιστοσελίδα ejs
- How to use EJS to Template Your Node Application, Chris Sev, 2020
- npm ejs

2.10

- Επίσημη ιστοσελίδα leaflet map
- Wikipedia – Leaflet

2.11

- Επίσημη Ιστοσελίδα Mongoose, Models
- Επίσημη Ιστοσελίδα MongoDB – Why Use MongoDB and When To Use It
- Επίσημη Ιστοσελίδα Mongoose, Guide
- MongoDB – Aggregation, tutorialspoint
- Wikipedia – Database
- Wikipedia – MongoDB
- What is MongoDB ecosystem – Tutorial, codezup, 2020

ΚΕΦΑΛΑΙΟ 3

3.2

- User Interface Design Basics, usability.gov

3.4

3.4.2

- Επίσημη Ιστοσελίδα Mongoose, έγγραφα (documents)
- localStorage in JavaScript: A complete guide, Nosa Osabeki, 2020

3.5

- Επίσημη Ιστοσελίδα Leaflet, Γρήγορος Οδηγός για τα πρώτα βήματα (quick-start)

ΚΕΦΑΛΑΙΟ 4

4.1

- HTTP response status codes, MDN Web Docs

4.2

- Screen Sizes & Breakpoints, Microsoft docs, 2020

4.3

- Επίσημη Ιστοσελίδα Heroku, τι είναι
- Επίσημη Ιστοσελίδα MongoDB, Atlas, Αρχικός Οδηγός (getting-started)
- Wikipedia - Heroku

6. ΠΑΡΑΡΤΗΜΑ

6.1 ΔΟΜΗ ΦΑΚΕΛΟΥ ΕΡΓΟΥ (ΦΑΚΕΛΟΙ & ΑΡΧΕΙΑ)

```
C:\Users\tokas\OneDrive\Υπολογιστής\δομη εφαρμογης>tree /f
Folder PATH listing for volume Windows-SSD
Volume serial number is 9466-A9F1
C:.
    .gitignore
    app.js
    gulpfile.js
    package-lock.json
    package.json

├───controllers
        index.js

├───models
        answer.js
        db.js

├───public
    │   favicon.ico
    │
    ├───assets
    │   ├───css
    │           leaflet.css
    │           style.css
    │
    │   ├───dist
    │           leaflet.min.css
    │           style.min.css
    │           webmap.min.js
    │
    │   └───images
                book-icon-green.png
                book-icon-pink.png
                book-icon-white.png
                book.png
                eve_map_png.png
                eve_map_png_crop.png
                Ghost-blue.png
                ghost-image.png
                Ghost-white.png
                graph-icon-green.png
                graph-icon-pink.png
                graphics.png
                LilGhost-ErrorPage.png
                map-icon-green.png
                map.png
                tick.png
                tree-logo.png
```


tree-logo.png
javascripts
buttons.js
webmap.js
routes
index.js
views
404.ejs
about.ejs
endofsurvey.ejs
index.ejs
postcovid.ejs
precovid.ejs
references.ejs
partials
head.ejs
jsFiles.ejs
navBar.ejs

ΚΩΔΙΚΑΣ:

6.2 ΦΑΚΕΛΟΣ Controllers

6.2.1 index.js

```
const mongoose = require("mongoose");
const Answer = require("../models/answer");

const dashboard = (_req, res, next) => {
  res.render("index");
};

const showReferencesPage = (_req, res) => {
  res.render("references");
};

const showAboutPage = (_req, res) => {
  res.render("about");
};

const showPrecovidPage = (_req, res) => {
  res.render("precovid");
};

const showPostCovidPage = (_req, res) => {
  res.render("postcovid");
};

const showEndOfSurveyPage = async (_req, res) => {
  const save_Data = _req.query;
  const data = { username: save_Data.username, oneoption:
JSON.parse(save_Data.oneoption), twooption:JSON.parse(save_Data.twooption)}
  const Newdata = new Answer(data)
  const result = await Newdata.save()
  if (result == "") {
      res.send({
          status: false
      })
  }
  else {
    res.render("endofsurvey");
  }
};
```

```
module.exports = { dashboard, showReferencesPage, showAboutPage,
showPrecovidPage, showPostCovidPage, showEndOfSurveyPage };
```

6.3 ΦΑΚΕΛΟΣ Models

6.3.1 answer.js

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const ObjectId = Schema.ObjectId;

// Create a schema
const answerSchema = new Schema({
  username: {
    type: String,
    required: true,
  },
  oneoption: {
    type: Object,
    required: true,
  },
  twooption: {
    type: Object,
    required: true,
  }
});

// Compile model from schema
module.exports = mongoose.model("Answers", answerSchema);
```

6.3.2 db.js

```
const mongoose = require("mongoose");

mongoose.set("useCreateIndex", true); // This prevents collection.ensureIndex
deprecation warning
mongoose.set("useFindAndModify", false); // This prevents current mongoose
deprecation warning


mongoose.connect("mongodb://localhost:27017/myCovidApp", { useNewUrlParser: true
,useUnifiedTopology: true}).then(() => {
```

```
    console.log('Database is connected');
})
```

6.4 ΦΑΚΕΛΟΣ Public

6.4.1 Φάκελος assets/css

6.4.1.1 style.css

```
html,
body {
  height: 100%;
  width: 100%;
}

body {
  background-color: #202125;
  margin: 0;
  color: #f8f5f1;
  font-family: "Montserrat", sans-serif;
  font-family: "Roboto", sans-serif;
}

.index{
  background-image: url("../images/eve_map_png_crop.png");
  background-position: center;
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: cover;
}

/* NAVIGATION BAR */
a {
  text-decoration: none;
  color: #f8f5f1;
}
a:hover {
  color: #5aa897;
}
.navbar {
  background-color: #202125;
  font-family: "Roboto Condensed", sans-serif;
  font-size: 20px;
  display: flex;
```

```
  justify-content: space-between;
}
.tree-logo {
  height: 3.75rem;
  width: 3.75rem;
  margin-left: 2rem;
}
.unordered-list {
  margin-left: auto;
}

.app-title{
  font-size: 1rem;
  padding-left: 1rem;
  padding-top: 0.5rem;

}
/* NAVBAR TOGGLER */
.navbar-toggler {
  background-color: #5aa897;
  color: #45526c;
}

/* ABOUT PAGE */
#footer {
  position: absolute;
  bottom: 0;
  width: 100%;
  height: 2.5rem;
  text-align: center;
}
.about-p,
.proCov-p {
  padding-left: 2rem;
}
.about-title {
  padding: 5rem 3rem 3rem 2rem;
}

/* HOME PAGE */

/* buttons before the user clicks on them */
.dotButtonBeforeClicked1,
.dotButtonBeforeClicked2 {
  background-color: black;
```

```
  color: black;
  border-radius: 50%;
  width: 10rem;
  height: 10rem;
  text-align: center;
  position: absolute;
}
.dotButtonBeforeClicked1 {
  margin: 14rem 0 0 20.37rem;
  padding-top: 3rem;
}
.dotButtonBeforeClicked2 {
  margin: 14rem 0 0 60.74rem;
  padding-top: 3rem;
}

/* buttons when the user clicks on them */
.dotButton1,
.dotButton2{
  position: absolute;
  color: #f8f5f1;
  border-radius: 50%;
  width: 10rem;
  height: 10rem;
  text-align: center;
  padding-top: 3rem;
}
.dotButton1 {
  background-color: #5aa897;
  margin: 14rem 0 0 20.37rem;
}
.dotButton2 {
  background-color: #45526c;
  margin: 14rem 0 0 60.74rem;
}
.dotButton1:hover {
  color: #45526c;
}
.dotButton2:hover {
  color: #5aa897;
}
.home-p {
  padding: 1.5rem 2rem 0 2rem;
}
.homeRedirLink {
```

```
  padding: 0.5rem;
}



/* MAP IMAGE */
.home-container {
  padding: 0 0 0 0;
  width: 100%;
  height: 100%;
  overflow: hidden;
}

/* ERROR PAGE & END OF SURVEY PAGE */
.end-p{
  color:#202125;
  margin-top: 2%;
  text-align: center;
}
.error-div{
  margin-top: 3rem;
}
.ghost-image {
  display: block;
  width: 30%;
  margin-left: auto;
  margin-right: auto;
}
.ghost-h2 {
  text-align: center;
  color: #5aa897;
}
.tick-h2 {
  margin-top: 7%;
  text-align: center;
  color: #202125;
}
.tick-image {
display: block;
width: 15%;
margin-left: auto;
margin-right: auto;
margin-top: 2%;
}
```

```css
/* PRE & POST COVID PAGE */

.page-container {
  margin-top: 5%;
  color: #45526c;
}
.covid-form {
  background-color: #202125;
  color: #f8f5f1;
  box-shadow: 0;
  border: 0;

}
.explanation-p {
  margin: 6% 5% 1% 5%;
}
hr {
  border-top: 5px dotted #45526c;
  background-color: #202125;
}
.modal-body {
  margin-left: -2%;
}
.modal-footer {
  border-top-color: #5aa897;
}
.modal-header {
  border-bottom-color: #5aa897;
}
.next-button {
  background-color: #45526c;
  color: #f8f5f1;
}
.question-div {
  margin: 2% auto 1% auto;
}

/* MAP CSS */
#map {
  width: auto;
  height: 500px;
  margin-top: 13%;

}
```

```css
/* REFERENCES PAGE */

.diag-div{
  margin-top: 6rem;
  background-color: #45526c;
  background-image: -webkit-linear-gradient(150deg, #45526c 35%, #5aa897 35%);
  min-height: 500px;
}

.ref-container {
  padding: 0 0 0 0;
  width: 100%;
  height: 70%;
  overflow: hidden;
}
.ref-list {
  padding-top: 3rem;
}
.ref-link{
  color: #202125;
}
.ref-link:hover {
  color: #45526c;
}
.ref-item {
  margin-top: 1.5rem;
}


/* MEDIA QUERIES */
@media (max-width: 960px) {

  /*HOMEPAGE*/
  .about-div {
    width: 80%;

  }
  #homeButtonOne,
  #homeButtonTwo {
    display: block;
    position: relative;
    margin-left: auto;
    margin-right: auto;
    margin-top: 10rem;
  }
```

```
  /*NAVIGATION BAR*/
  .navbar-toggler {
    margin-left: 2rem;
    z-index: 2;

  }
  .navbar-collapse {
    margin-left: 2rem;
  }
  .tree-logo {
    display: none;
  }

  /*PRE&DUR COVID-19 PAGE*/
  .page-container {
    margin-right: 1rem;
    max-width: 100%;
  }
  /* END OF SURVEY PAGE */
 .end-container {
   margin-top: 45%;
 }
  /*ERROR PAGE*/
  .error-div {
    margin-top: 20%;
  }
  .ghost-image {
    width: 60%;
  }
  /* REFERENCES PAGE */
  .ref-list {
    padding-top: 0.5rem;
  }
  .ref-item {
    margin-top: 1rem;
  }
}

@media (min-width: 961px) and (max-width: 1366px) {
  .dotButton1,
  .dotButton2,
  .dotButtonBeforeClicked1,
  .dotButtonBeforeClicked2{
    width: 15.5%;
```

```
  }
.dotButton1,
.dotButtonBeforeClicked1 {
    margin-left: 23.3%;
  }
.dotButton2,
.dotButtonBeforeClicked2 {
    margin-left: 66.6%;
  }
}
```

6.4.1.2 leaflet.css

Ο κύριος όγκος του κώδικα είναι από προεπιλογή του leaflet και δεν αλλάχθηκε, παρακάτω παρατίθεται το μόνο κομμάτι που προσαρμόστηκε για τις ανάγκες της συγκεκριμένης εφαρμογής:

```
.leaflet-popup-content-wrapper,
.leaflet-popup-tip {
  background: #0078a8;
  color: #202125;
  box-shadow: 0 3px 14px rgba(0, 0, 0, 0.4);
}
```

6.4.2 public/javascripts

6.4.2.1 buttons.js

```
const buttonOne = document.getElementById("homeButtonOne");
const buttonTwo = document.getElementById("homeButtonTwo");
buttonOne.addEventListener("mouseenter", addClassFirst);
buttonTwo.addEventListener("mouseenter", addClassSecond);

function addClassFirst() {
  buttonOne.classList.add("dotButton1");
  buttonOne.classList.remove("dotButtonBeforeClicked1");
}
function addClassSecond() {
  buttonTwo.classList.add("dotButton2");
  buttonTwo.classList.remove("dotButtonBeforeClicked2");
}
```

6.4.2.2 webmap.js

Επειδή το συγκεκριμένο κομμάτι κώδικα είναι από προεπιλογή από το leaflet map, ακολουθεί μόνο το κομμάτι που αλλάχθηκε για την συγκεκριμένη εφαρμογή. Ο υπόλοιπος κώδικας μπορεί να βρεθεί στην επίσημη ιστοσελίδα του leaflet: https://leafletjs.com/
καθώς και στο github: https://github.com/Leaflet/Leaflet

```
// Import the leaflet package
let L = require("leaflet");

// Creates a leaflet map binded to an html <div> with id "map"
// setView will set the initial map view to the location at coordinates
// 20 represents the initial zoom level with higher values being more zoomed in
let map = L.map("map").setView([37.9709797, 23.7241881], 20);

// Adds the basemap tiles to the web map
// Additional providers are available at: https://leaflet-
extras.github.io/leaflet-providers/preview/
L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
  maxZoom: 19,
  attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',
}).addTo(map);

// Adds a popup marker to the webmap for GGL address & option to define the
location
map.on("click", function (e) {
  let lt = String(e.latlng.lat),
    lg = String(e.latlng.lng);
  let popup = L.popup()
    .setLatLng(e.latlng)
    .setContent('<div class="custom-control custom-radio question-div"><input
type = "hidden" id = "position_x" value = '+lt+'><input type = "hidden" id =
"position_y" value = '+lg+'><label>Καταχωρήστε μια ενδεικτική τοποθεσία, και
επιλέξτε το είδος της:</label><br><label for="q8a1"><input class="custom-control-
input" type="radio" id="q8a1" name="question8" required>
κατάστημα</label><br><label for="q8a2"><input class="custom-control-input"
type="radio" id="q8a2" name="question8" required> σπίτι</label><br><label
for="q8a3"><input class="custom-control-input" type="radio" id="q8a3"
name="question8" required> δημόσιος χώρος</label><br></div>')
    .openOn(map);
});
```

6.5 ΦΑΚΕΛΟΣ Routes

6.5.1 index.js

```
const express = require("express");
const router = express.Router();
const indexController = require("../controllers/index");

router.route("/").get(indexController.dashboard);

router.route("/references").get(indexController.showReferencesPage);

router.route("/about").get(indexController.showAboutPage);

router.route("/precovid").get(indexController.showPrecovidPage);

router.route("/postcovid").get(indexController.showPostCovidPage);

router.route("/endofsurvey").get(indexController.showEndOfSurveyPage);

//exports the router we are using
module.exports = router;
```

6.6 ΦΑΚΕΛΟΣ View

6.6.1 about.ejs

```
<!DOCTYPE html>
<html lang="en">
<%- include("./partials/head.ejs") %>

<body>
  <nav class="navbar navbar-expand-lg">
    <div class="container-fluid">
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarTogglerDemo02" aria-controls="navbarTogglerDemo02" aria-
expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarTogglerDemo02">
        <img src="/assets/images/tree-logo.png" class="tree-logo" />
```

```
        <ul class="unordered-list ml-auto navbar-nav mb-2 mb-lg-0">
          <li class="nav-item">
            <a class="nav-link" aria-current="page" href="/">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/about">About</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
  <main role="main" class="container-fluid">
    <!-- TITLE & ABOUT PARAGRAPH -->
    <div class="about-div">
      <h3 class="about-title">"Ανάπτυξη διαδικτυακής πλατφόρμας για την
διερεύνηση της χρήσης του δημοσίου χώρου των πόλεων κατά τη διάρκεια της
πανδημίας Covid-19."</h3>
      <p class="about-p">
        Η παρούσα εργασία αφορά τον σχεδιασμό και την ανάπτυξη μιας λειτουργικής
διαδικτυακής πλατφόρμας (εφαρμογής) η οποία θα χρησιμεύσει ως εργαλείο
        για την διερεύνηση της χρήσης του δημόσιου χώρου της Αθήνας και πως
άλλαξε αυτή κατά την διάρκεια της πανδημίας Covid-19.
      </p>
      <br>
      <p class="about-p">Συγγραφή: Εύα Ανέστου<br>
        <a href="http://semfe.ntua.gr/el/faculty-members/item/40-
stefaneas">Επιβλέπων Καθηγητής: Πέτρος Στεφανέας </a><br>
        <a href="http://semfe.ntua.gr/el/">Σχολή: Εφαρμοσμένων Μαθηματικών &
Φυσικών Επιστημών, ΕΜΠ</a>
      </p>
    </div>
  </main>
  <footer id="footer">
    &copyΣεπτέμβριος 2021
  </footer>
  <%- include("./partials/jsFiles.ejs") %>
</body>

</html>
```

6.6.2 index.ejs

```
<!DOCTYPE html>
<html lang="en">
<%- include('./partials/head.ejs') %>
```

```
<body>
  <main role="main" class="container-fluid home-container index">
    <!-- MAP IMAGE -->
    <%- include('./partials/navBar.ejs') %>

    <div class="circle1">
      <a href="/precovid" type="button" id="homeButtonOne" class="btn btn-lg
dotButtonBeforeClicked1"> Πάρτε μέρος στην έρευνα </a>
    </div>
    <div class="circle2">
      <a href="/references" type="button" id="homeButtonTwo" class="btn btn-lg
dotButtonBeforeClicked2"> Διαβάστε περισσότερα </a>
    </div>
  </main>
  <script src="../javascripts/buttons.js"></script>
  <%- include("./partials/jsFiles.ejs") %>
</body>

</html>
```

6.6.3 precovid.ejs

```
<!DOCTYPE html>
<html lang="en">
<%- include("./partials/head.ejs") %>

<body>
  <%- include("./partials/navBar.ejs") %>
  <main role="main" class="container-fluid">
    <!-- Εξήγηση για τον χρήστη -->
    <p class="explanation-p">Η συγκεκριμένη έρευνα έχει να κάνει με την
συμπεριφορά του πολιτη στο δημόσιο χώρο της
      Αθήνας.
      Σκοπός είναι να εξετάσουμε αν και κατά πόσο άλλαξαν οι συνήθειες μας
σχετικά με τον δημόσιο χώρο κατά την
      διάρκεια της καραντίνας.
      Αυτή η σελίδα αφορά την περίοδο πριν τον Covid-19. Απαντήστε στις ερωτήσεις
και καταχωρείστε στον χάρτη
      μια ενδεικτική τοποθεσία που θα αντικατοπτρίζει το που βγαίνατε ή συχνάζατε
περισσότερο και διαλέξτε μια από τις
      επιλογές που έχει το marker σχετικά με το είδος της τοποθεσίας (μαγαζί,
σπίτι ή δημόσιος χώρος).</p>
    <div class="container page-container">
      <div class="row">
        <div class="col-sm">
          <form action="/postcovid" method="GET">
```

```
            <div class="modal-content covid-form">
              <div class="modal-header">
                <h2 class="modal-title"> Ερωτήσεις</h2>
              </div>

              <!-- Ερωτήσεις -->
              <div class="modal-body">
                <!-- Input Type Radio Button -->
                <div class="custom-control custom-radio question-div">
                  <label>
                    Εισαγάγετε το όνομά σας.
                  </label><br><br>
                  <label for="name" style="width: 100%;">
                    <input class="custom-control-input" style="width: 100%;"
type="email" id="name" name="name" required>
                  </label><br>
                </div>
                <br>
                <hr>
                <div class="custom-control custom-radio question-div">
                  <label>
                    Πόσο συχνά επισκεπτόσασταν έναν δημόσιο χώρο πριν την
καραντίνα?
                  </label><br>
                  <label for="q1a1">
                    <input class="custom-control-input" type="radio" id="q1a1"
name="question1" required> Κάθε μέρα
                  </label><br>
                  <label for="q1a2">
                    <input class="custom-control-input" type="radio" id="q1a2"
name="question1" required> 3-4 φορές
                    την εβδομάδα
                  </label><br>
                  <label for="q1a3">
                    <input class="custom-control-input" type="radio" id="q1a3"
name="question1" required> 1 φορά την
                    εβδομάδα
                  </label><br>
                  <label for="q1a4">
                    <input class="custom-control-input" type="radio" id="q1a4"
name="question1" required> Λιγότερο
                    από 3 φορές τον μήνα
                  </label><br>
                  <label for="q1a5">
```

```
                    <input class="custom-control-input" type="radio" id="q1a5"
name="question1" required> Καθόλου ή
                    λιγότερο από 1 φορά το μήνα
                  </label><br>
                </div>

                <hr>

                <div class="custom-control custom-radio question-div">
                  <label>
                    Τι είδους δημόσιο χώρο επισκεπτόσασταν?
                  </label><br>
                  <input class="custom-control-input" type="radio" id="q2a1"
name="question2" required> Πλατεία
                  <label for="q2a1"></label><br>
                  <label for="q2a2">
                    <input class="custom-control-input" type="radio" id="q2a2"
name="question2" required> Πάρκο
                  </label><br>
                  <label for="q2a3">
                    <input class="custom-control-input" type="radio" id="q2a3"
name="question2" required> Πεζόδρομο
                  </label><br>
                  <label for="q2a4">
                    <input class="custom-control-input" type="radio" id="q2a4"
name="question2" required> Άλσος
                  </label><br>
                  <label for="q2a5">
                    <input class="custom-control-input" type="radio" id="q2a5"
name="question2" required> Παραλία
                  </label><br>
                </div>

                <hr>

                <div class="custom-control custom-radio question-div">
                  <label>
                    Όταν βρισκόσασταν με την παρέα σας προτιμούσατε να πηγαίνετε
σε
                  </label><br>
                  <label for="q3a1">
                    <input class="custom-control-input" type="radio" id="q3a1"
name="question3" required> Δημόσιο χώρο (Πάρκο, Πλατεία κλπ)
                  </label><br>
                  <label for="q3a2">
```

```
                    <input class="custom-control-input" type="radio" id="q3a2"
name="question3" required> Μαγαζί
                  </label><br>
                  <label for="q3a3">
                    <input class="custom-control-input" type="radio" id="q3a3"
name="question3" required> Μαγαζί
                    που έχει τραπεζοκαθίσματα πάνω σε δημόσιο χώρο
                  </label><br>
                  <label for="q4a4">
                    <input class="custom-control-input" type="radio" id="q3a4"
name="question3" required> σπίτι
                    πάνω σε πλατεία
                  </label><br>
                </div>

                <hr>

                <div class="custom-control custom-radio question-div">
                  <label>
                    Ποιος είναι ο λόγος που επισκεπτόσασταν κάποιον δημόσιο χώρο;
                  </label><br>
                  <label for="q4a1">
                    <input class="custom-control-input" type="radio" id="q4a1"
name="question4" required> Άθληση
                  </label><br>
                  <label for="q4a2">
                    <input class="custom-control-input" type="radio" id="q4a2"
name="question4" required>
                    Βόλτα με την παρέα σας
                  </label><br>
                  <label for="q4a3">
                    <input class="custom-control-input" type="radio" id="q4a3"
name="question4" required> Ατομική
                    βόλτα/περπάτημα
                  </label><br>
                  <label for="q4a4">
                    <input class="custom-control-input" type="radio" id="q4a4"
name="question4" required> Βόλτα με
                    κατοικίδια
                  </label><br>
                  <label for="q4a5">
                    <input class="custom-control-input" type="radio" id="q4a5"
name="question4" required> Βόλτα με
                    την οικογένεια σας
                  </label><br>
```

```
                  <label for="q4a6">
                    <input class="custom-control-input" type="radio" id="q4a6"
name="question4" required> Ο χώρος αυτός εχει
                    μαγαζιά
                  </label><br>
                </div>

                <hr>

                <div class="custom-control custom-radio question-div">
                  <label>
                    Ποιος ήταν ο μέσος χρόνος παραμονής σας σε έναν δημόσιο χώρο;
                  </label><br>
                  <label for="q5a1">
                    <input class="custom-control-input" type="radio" id="q5a1"
name="question5" required> Κάτω από
                    μισή ώρα
                  </label><br>
                  <label for="q5a2">
                    <input class="custom-control-input" type="radio" id="q5a2"
name="question5" required> Μισή με
                    μία ώρα
                  </label><br>
                  <label for="q5a3">
                    <input class="custom-control-input" type="radio" id="q5a3"
name="question5" required> Μία με δύο
                    ώρες
                  </label><br>
                  <label for="q5a4">
                    <input class="custom-control-input" type="radio" id="q5a4"
name="question5" required> Πάνω από
                    δύο ώρες
                  </label><br>
                </div>

                <hr>

                <div class="custom-control custom-radio question-div">
                  <label>
                    Ποιο ήταν το μέσο μετακίνησης σας προς τον δημόσιο χώρο;
                  </label><br>
                  <label for="q6a1">
                    <input class="custom-control-input" type="radio" id="q6a1"
name="question6" required> Μέσα
                    Μαζικής Μεταφοράς
```

```
                </label><br>
                <label for="q6a2">
                  <input class="custom-control-input" type="radio" id="q6a2"
name="question6" required> Αυτοκίνητο
                </label><br>
                <label for="q6a3">
                  <input class="custom-control-input" type="radio" id="q6a3"
name="question6" required> Περπάτημα
                </label><br>
                <label for="q6a4">
                  <input class="custom-control-input" type="radio" id="q6a4"
name="question6" required> Ποδήλατο
                </label><br>
              </div>

              <hr>

              <!-- Input Type Checkbox -->
              <div class="custom-control question-div">
                <label>Για ποιον/ποιους λόγους σας προσέλκυε ο δημόσιος χώρος
που επισκεπτόσασταν πιο συχνά;
                  <small>(Σημειώστε με όσα συμφωνείτε)</small>
                </label><br>
                <label for="inp-1">
                  <input class="custom-control-input" type="checkbox"
name="inp"> Βρίσκεται κοντά στην κατοικία
                  σας</label><br>
                <label for="inp-2">
                  <input class="custom-control-input" type="checkbox"
name="inp"> Βρίσκεται σε μέρος που έχει
                  όμορφο τοπίο</label><br>
                <label for="inp-3">
                  <input class="custom-control-input" type="checkbox"
name="inp"> Δεν σας αρέσει αλλά τον
                  επισκέπτεστε επειδή συχνάζει εκεί η παρέα σας </label><br>
                <label for="inp-4">
                  <input class="custom-control-input" type="checkbox"
name="inp"> Δεν συχνάζει πολύ
                  κόσμος</label><br>
                <label for="inp-5">
                  <input class="custom-control-input" type="checkbox"
name="inp"> Συχνάζει πολύ κόσμος</label><br>
                <label for="inp-6">
                  <input class="custom-control-input" type="checkbox"
name="inp"> Είναι ωραία
```

```
                    διαμορφωμένος/Έχει καλές-επαρκείς υποδομές</label><br>
                </div>
              </div>
            </div>
            <div class="modal-footer">
              <button type="button" id="nextButtonNamePre" class="btn next-
button" onclick="precovid()">Next</button>
            </div>
        </div>
        </form>

        <!-- MAP DIV -->
        <div class="col-sm map-div">
          <div id="map"></div>
          <script src="javascripts/webmap.js"></script>
          <script>
            document.getElementById('nextButtonNamePre').onclick = function() {
              var name = document.getElementById("name").value
              if (name == "") {
                alert("Εισάγετε το όνομα σας!")
                return
              }
              var value1 = "";
              var i = 1
              var markedCheckbox1 =
document.querySelectorAll('input[name="question1"]');
              for (var checkbox1 of markedCheckbox1) {
                if (checkbox1.checked == true) {
                  value1 = i;
                }
                i++;
              }
              console.log(value1)
              if (value1 == "") {
                alert("Ξέχασες να απαντήσεις στην πρώτη ερώτηση!")
                return
              }

              var value2 = "";
              var i = 1
              var markedCheckbox2 =
document.querySelectorAll('input[name="question2"]');
              for (var checkbox2 of markedCheckbox2) {
                if (checkbox2.checked == true) {
                  value2 = i;
```

```
                }
                i++;
              }
              if (value2 == "") {
                alert("Ξέχασες να απαντήσεις στην δεύτερη ερώτηση!")
                return
              }


              var value3 = "";
              var i = 1
              var markedCheckbox3 =
document.querySelectorAll('input[name="question3"]');
              for (var checkbox3 of markedCheckbox3) {
                if (checkbox3.checked == true) {
                  value3 = i;
                }
                i++;
              }
              if (value3 == "") {
                alert("Ξέχασες να απαντήσεις στην τρίτη ερώτηση!")
                return
              }

              var value4 = "";
              var i = 1;
              var markedCheckbox4 =
document.querySelectorAll('input[name="question4"]');
              for (var checkbox4 of markedCheckbox4) {
                if (checkbox4.checked == true) {
                  value4 = i;
                }
                i++;
              }
              if (value4 == "") {
                alert("Ξέχασες να απαντήσεις στην τέταρτη ερώτηση!")
                return
              }

              var value5 = "";
              var i = 1;
              var markedCheckbox5 =
document.querySelectorAll('input[name="question5"]');
              for (var checkbox5 of markedCheckbox5) {
                if (checkbox5.checked == true) {
```

```
                value5 = i;
              }
              i++;
            }
            if (value5 == "") {
              alert("Ξέχασες να απαντήσεις στην πέμπτη ερώτηση!")
              return
            }

            var value6 = "";
            var i = 1;
            var markedCheckbox6 =
document.querySelectorAll('input[name="question6"]');
            for (var checkbox6 of markedCheckbox6) {
              if (checkbox6.checked == true) {
                value6 = i;
              }
              i++;
            }
            if (value6 == "") {
              alert("Ξέχασες να απαντήσεις στην έκτη ερώτηση!")
              return
            }

            var value = []
            var i = 1;
            var markedCheckbox =
document.querySelectorAll('input[type="checkbox"]:checked');
            for (var checkbox of markedCheckbox) {
              if (checkbox.value == "on") {
                value.push(i);
              }
              i++;
            }
            if (value == "") {
              alert("Ξέχασες να απαντήσεις στην τελευταία ερώτηση!")
              return
            }

            var position = {}
            if (document.getElementById("position_x") == undefined) {
              alert("Εισάγετε τοποθεσία στον χάρτη!")
              return
            } else {
              position_x = document.getElementById("position_x").value
```

```
                position_y = document.getElementById("position_y").value
                position = {
                  x: position_x,
                  y: position_y
                }
              }
              var value8 = "";
              var i = 1;
              var markedCheckbox8 =
document.querySelectorAll('input[name="question8"]');
              for (var checkbox8 of markedCheckbox8) {
                if (checkbox8.checked == true) {
                  value8 = i;
                }
                i++;
              }
              if (value8 == "") {
                alert("Διαλέξτε μία από τις επιλογές του marker στον χάρτη!")
                return
              }
              var Oneoption = {
                position: position,
                option1: value1,
                option2: value2,
                option3: value3,
                option4: value4,
                option5: value5,
                option6: value6,
                option8: value8,
              }
              console.log(Oneoption)
              localStorage.username = name;
              localStorage.Oneoption = JSON.stringify(Oneoption);
              document.querySelector("form").submit();
            }
          </script>
        </div>
      </div>
    </div>
  </main>

  <%- include("./partials/jsFiles.ejs") %>
</body>

</html>
```

6.6.4 postcovid.ejs

```
<!DOCTYPE html>
<html lang="en">
<%- include("./partials/head.ejs") %>

<body>
  <%- include("./partials/navBar.ejs") %>
  <main role="main" class="container-fluid">
    <!-- Εξήγηση για τον χρήστη -->
    <p class="explanation-p">
      Αυτή η σελίδα αφορά την περίοδο που ήμασταν σε καραντίνα. Απαντήστε στις
ερωτήσεις και καταχωρείστε στον χάρτη
      μια ενδεικτική τοποθεσία
      που θα αντικατοπτρίζει το που βγαίνατε ή συχνάζατε περισσότερο κατά την
διάρκεια της καραντίνας και διαλέξτε μια από τις
      επιλογές που έχει το marker σχετικά με το είδος της τοποθεσίας
      (σπίτι ή δημόσιος χώρος).</p>
    <div class="container">
      <div class="row">
        <div class="col-sm">
          <form action="/endofsurvey" method="GET">
            <div class="modal-content covid-form">
              <div class="modal-header">
                <h2>Ερωτήσεις</h2>
              </div>
              <!-- Ερωτήσεις -->
              <div class="modal-body">
                <!-- Input Type Radio Button -->
                <div class="custom-control custom-radio question-div">
                  <label>
                    Πόσο συχνά επισκεπτόσασταν έναν δημόσιο χώρο κατά την
διάρκεια της καραντίνας?
                  </label><br>
                  <label for="q1a1">
                    <input class="custom-control-input" type="radio" id="q1a1"
name="question1" required> Κάθε μέρα
                  </label><br>
                  <label for="q1a2">
                    <input class="custom-control-input" type="radio" id="q1a2"
name="question1" required> 3-4 φορές
                    την εβδομάδα
                  </label><br>
                  <label for="q1a3">
```

```
                    <input class="custom-control-input" type="radio" id="q1a3"
name="question1" required> 1 φορά την
                    εβδομάδα
                  </label><br>
                  <label for="q1a4">
                    <input class="custom-control-input" type="radio" id="q1a4"
name="question1" required> Λιγότερο
                    από 3 φορές τον μήνα
                  </label><br>
                  <label for="q1a5">
                    <input class="custom-control-input" type="radio" id="q1a5"
name="question1" required> Καθόλου ή
                    λιγότερο από 1 φορά το μήνα
                  </label><br>
                </div>

                <hr>

                <div class="custom-control custom-radio question-div">
                  <label>
                    Τι είδους δημόσιο χώρο επισκεπτόσασταν?
                  </label><br>
                  <input class="custom-control-input" type="radio" id="q2a1"
name="question2" required> Πλατεία
                  <label for="q2a1"></label><br>
                  <label for="q2a2">
                    <input class="custom-control-input" type="radio" id="q2a2"
name="question2" required> Πάρκο
                  </label><br>
                  <label for="q2a3">
                    <input class="custom-control-input" type="radio" id="q2a3"
name="question2" required> Πεζόδρομο
                  </label><br>
                  <label for="q2a4">
                    <input class="custom-control-input" type="radio" id="q2a4"
name="question2" required> Άλσος
                  </label><br>
                  <label for="q2a5">
                    <input class="custom-control-input" type="radio" id="q2a5"
name="question2" required> Παραλία
                  </label><br>
                </div>

                <hr>
```

```
                <div class="custom-control custom-radio question-div">
                  <label>
                    Όταν κλείσανε τα μαγαζιά και βρισκόσασταν με την παρέα σας
προτιμούσατε να πηγαίνετε σε
                  </label><br>
                  <label for="q3a1">
                    <input class="custom-control-input" type="radio" id="q3a1"
name="question3" required> Πλατεία
                  </label><br>
                  <label for="q3a2">
                    <input class="custom-control-input" type="radio" id="q3a2"
name="question3" required> Πάρκο
                  </label><br>
                  <label for="q3a3">
                    <input class="custom-control-input" type="radio" id="q3a3"
name="question3" required> Κάποιο
                    σπίτι
                  </label><br>
                  <label for="q3a4">
                    <input class="custom-control-input" type="radio" id="q3a4"
name="question3" required> Δεν
                    βρισκόμουν με την παρέα μου
                  </label><br>
                </div>

                <hr>

                <div class="custom-control custom-radio question-div">
                  <label>
                    Ποιος είναι ο λόγος που επισκεπτόσασταν κάποιον δημόσιο χώρο;
                  </label><br>
                  <label for="q4a1">
                    <input class="custom-control-input" type="radio" id="q4a1"
name="question4" required> Άθληση
                  </label><br>
                  <label for="q4a2">
                    <input class="custom-control-input" type="radio" id="q4a2"
name="question4" required>
                    Συναναστροφή με άλλους ανθρώπους
                  </label><br>
                  <label for="q4a3">
                    <input class="custom-control-input" type="radio" id="q4a3"
name="question4" required> Ατομική
                    βόλτα/περπάτημα
                  </label><br>
```

```
                  <label for="q4a4">
                    <input class="custom-control-input" type="radio" id="q4a4"
name="question4" required> Βόλτα με
                    κατοικίδια
                  </label><br>
                  <label for="q4a5">
                    <input class="custom-control-input" type="radio" id="q4a5"
name="question4" required> Βόλτα με
                    την οικογένεια σας
                  </label><br>
                  <label for="q4a6">
                    <input class="custom-control-input" type="radio" id="q4a6"
name="question4" required> Βόλτα με
                    τους φίλους σας
                </div>

                <hr>

                <div class="custom-control custom-radio question-div">
                  <label>
                    Ποιος ήταν ο μέσος χρόνος παραμονής σας σε έναν δημόσιο χώρο;
                  </label><br>
                  <label for="q5a1">
                    <input class="custom-control-input" type="radio" id="q5a1"
name="question5" required> Κάτω από
                    μισή ώρα
                  </label><br>
                  <label for="q5a2">
                    <input class="custom-control-input" type="radio" id="q5a2"
name="question5" required> Μισή με
                    μία ώρα
                  </label><br>
                  <label for="q5a3">
                    <input class="custom-control-input" type="radio" id="q5a3"
name="question5" required> Μία με δύο
                    ώρες
                  </label><br>
                  <label for="q5a4">
                    <input class="custom-control-input" type="radio" id="q5a4"
name="question5" required> Πάνω από
                    δύο ώρες
                  </label><br>
                </div>

                <hr>
```

```
                <div class="custom-control custom-radio question-div">
                  <label>
                    Ποιο ήταν το μέσο μετακίνησης σας προς τον δημόσιο χώρο;
                  </label><br>
                  <label for="q6a1">
                    <input class="custom-control-input" type="radio" id="q6a1"
name="question6" required> Μέσα
                    Μαζικής Μεταφοράς
                  </label><br>
                  <label for="q6a2">
                    <input class="custom-control-input" type="radio" id="q6a2"
name="question6" required> Αυτοκίνητο
                  </label><br>
                  <label for="q6a3">
                    <input class="custom-control-input" type="radio" id="q6a3"
name="question6" required> Περπάτημα
                  </label><br>
                  <label for="q6a4">
                    <input class="custom-control-input" type="radio" id="q6a4"
name="question6" required> Ποδήλατο
                  </label><br>
                </div>

                <hr>

                <!-- Input Type Checkbox -->
                <div class="custom-control question-div">
                  <label>Για ποιον/ποιους λόγους σας προσέλκυε ο δημόσιος χώρος
που επισκεπτόσασταν πιο συχνά;
                    <small>(Σημειώστε με όσα συμφωνείτε)</small>
                  </label><br>
                  <label for="inp-1">
                    <input class="custom-control-input" type="checkbox"
name="inp" required> Βρίσκεται κοντά στην
                    κατοικία σας</label><br>
                  <label for="inp-2">
                    <input class="custom-control-input" type="checkbox"
name="inp" required> Βρίσκεται σε μέρος που
                    έχει όμορφο τοπίο</label><br>
                  <label for="inp-3">
                    <input class="custom-control-input" type="checkbox"
name="inp" required> Δεν σας αρέσει αλλά τον
                    επισκέπτεστε επειδή συχνάζει εκεί η παρέα σας </label><br>
                  <label for="inp-4">
```

```
                    <input class="custom-control-input" type="checkbox"
name="inp" required> Δεν συχνάζει πολύ
                    κόσμος</label><br>
                  <label for="inp-5">
                    <input class="custom-control-input" type="checkbox"
name="inp" required> Συχνάζει πολύ
                    κόσμος</label><br>
                  <label for="inp-6">
                    <input class="custom-control-input" type="checkbox"
name="inp" required> Είναι ωραία
                    διαμορφωμένος/Έχει καλές-επαρκείς υποδομές</label><br>
                </div>

                <hr>

                <div class="custom-control custom-radio question-div">
                  <label>Η απαγόρευση κυκλοφορίας, μετά τις 21:00: επηρέασε τη
συχνότητα επίσκεψης των δημόσιων
                    χώρων ή/και το χρόνο παραμονής σας σε αυτούς;
                  </label><br>
                  <label for="q7a1">
                    <input type="radio" id="q7a1" name="question7" required> Ναι,
επηρέασε την
                    συχνότητα</label><br>
                  <label for="q7a2">
                    <input type="radio" id="q7a2" name="question7" required> Ναι,
έχει επηρέασε τον χρόνο
                    παραμονής</label><br>
                  <label for="q7a3">
                    <input type="radio" id="q7a3" name="question7" required> Ναι,
έχει επηρέασε και τα δύο παραπάνω
                  </label><br>
                  <label for="q7a4">
                    <input type="radio" id="q7a4" name="question7" required>
Όχι</label><br>
                </div>
              </div>
            </div>
            <input type="hidden" id="oneoption" name="oneoption">
            <input type="hidden" id="twooption" name="twooption">
            <input type="hidden" id="username" name="username">
            <div class="modal-footer">
              <button type="button" id="nextButtonNamePre" class="btn next-
button">Next</button>
            </div>
```

```
        </div>
        </form>


        <div class="col-sm">
          <div id="map"></div>
          <script src="javascripts/webmap.js"></script>
          <script>
            document.getElementById('nextButtonNamePre').onclick = function() {
              var value1 = "";
              var i = 1
              var markedCheckbox1 =
document.querySelectorAll('input[name="question1"]');
              for (var checkbox1 of markedCheckbox1) {
                if (checkbox1.checked == true) {
                  value1 = i;
                }
                i++;
              }
              console.log(value1)
              if (value1 == "") {
                alert("Ξέχασες να απαντήσεις στην πρώτη ερώτηση!")
                return
              }

              var value2 = "";
              var i = 1
              var markedCheckbox2 =
document.querySelectorAll('input[name="question2"]');
              for (var checkbox2 of markedCheckbox2) {
                if (checkbox2.checked == true) {
                  value2 = i;
                }
                i++;
              }
              if (value2 == "") {
                alert("Ξέχασες να απαντήσεις στην δεύτερη ερώτηση!")
                return
              }


              var value3 = "";
              var i = 1
              var markedCheckbox3 =
document.querySelectorAll('input[name="question3"]');
```

```
            for (var checkbox3 of markedCheckbox3) {
              if (checkbox3.checked == true) {
                value3 = i;
              }
              i++;
            }
            if (value3 == "") {
              alert("Ξέχασες να απαντήσεις στην τρίτη ερώτηση!")
              return
            }

            var value4 = "";
            var i = 1;
            var markedCheckbox4 =
document.querySelectorAll('input[name="question4"]');
            for (var checkbox4 of markedCheckbox4) {
              if (checkbox4.checked == true) {
                value4 = i;
              }
              i++;
            }
            if (value4 == "") {
              alert("Ξέχασες να απαντήσεις στην τέταρτη ερώτηση!")
              return
            }

            var value5 = "";
            var i = 1;
            var markedCheckbox5 =
document.querySelectorAll('input[name="question5"]');
            for (var checkbox5 of markedCheckbox5) {
              if (checkbox5.checked == true) {
                value5 = i;
              }
              i++;
            }
            if (value5 == "") {
              alert("Ξέχασες να απαντήσεις στην πέμπτη ερώτηση!")
              return
            }

            var value6 = "";
            var i = 1;
            var markedCheckbox6 =
document.querySelectorAll('input[name="question6"]');
```

```
            for (var checkbox6 of markedCheckbox6) {
              if (checkbox6.checked == true) {
                value6 = i;
              }
              i++;
            }
            if (value6 == "") {
              alert("Ξέχασες να απαντήσεις στην έκτη ερώτηση!")
              return
            }

            var value7 = "";
            var i = 1;
            var markedCheckbox7 =
document.querySelectorAll('input[name="question7"]');
            for (var checkbox7 of markedCheckbox7) {
              if (checkbox7.checked == true) {
                value7 = i;
              }
              i++;
            }
            if (value7 == "") {
              alert("Ξέχασες να απαντήσεις στην έβδομη ερώτηση!")
              return
            }
            var value = []
            var i = 1;
            var markedCheckbox =
document.querySelectorAll('input[type="checkbox"]:checked');
            for (var checkbox of markedCheckbox) {
              if (checkbox.value == "on") {
                value.push(i);
              }
              i++;
            }
            if (value == "") {
              alert("Ξέχασες να απαντήσεις στην τελευταία ερώτηση!")
              return
            }

            var position = {}
            if (document.getElementById("position_x") == undefined) {
              alert("Καταχωρήστε μια τοποθεσία στον χάρτη!")
              return
            } else {
```

```
                position_x = document.getElementById("position_x").value
                position_y = document.getElementById("position_y").value
                position = {
                  x: position_x,
                  y: position_y
                }
              }

              var value8 = "";
              var i = 1;
              var markedCheckbox8 =
document.querySelectorAll('input[name="question8"]');
              for (var checkbox8 of markedCheckbox8) {
                if (checkbox8.checked == true) {
                  value8 = i;
                }
                i++;
              }
              if (value8 == "") {
                alert("Select Option 8")
                return
              }

              var Twooption = {
                position: position,
                option1: value1,
                option2: value2,
                option3: value3,
                option4: value4,
                option5: value5,
                option6: value6,
                option7: value7,
                option8: value8
              }
              // console.log(Twooption)
              document.getElementById("username").value = localStorage.username;
              document.getElementById("oneoption").value =
localStorage.Oneoption;
              document.getElementById("twooption").value =
JSON.stringify(Twooption);
              document.querySelector("form").submit();
            }
          </script>
        </div>
      </div>
```

```
    </div>
  </main>

  <%- include("./partials/jsFiles.ejs") %>
</body>

</html>
```

6.6.5 endofsurvey.ejs

```
<!DOCTYPE html>
<html lang="en">
<%- include("./partials/head.ejs") %>

<body>
  <main role="main" class="container-fluid home-container index">
    <%- include("./partials/navBar.ejs") %>

    <div class="container">
      <h2 class="tick-h2">Ευχαριστούμε για την συμμετοχή!</h2>
      <img class="tick-image" src="../assets/images/tick.png">
      <p class="end-p">Αν θέλετε να διαβάσετε περισσότερα για το θέμα που
διαπραγματεύεται η έρευνα
        <br> πατήστε το About που βρίσκεται πάνω δεξιά στην οθόνη σας
      </p>
    </div>
  </main>
  <%- include("./partials/jsFiles.ejs") %>
</body>

</html>
```

6.6.6 references.ejs

```
<!DOCTYPE html>
<html lang="en">
<%- include("./partials/head.ejs") %>


<body>
  <%- include('./partials/navBar.ejs') %>
  <main role="main" class="container-fluid ref-container references">

    <div class="ref-div diag-div">
      <ul class="ref-list">
        <li class="ref-item">
```

```
          <a class="ref-link" href="https://www.mdpi.com/2071-
1050/13/9/4620/htm">Cycling in the Era of COVID-19: Lessons Learnt and Best
Practice Policy Recommendations for a More Bike-Centric Future
          </a>
        </li>
        <li class="ref-item">
          <a class="ref-link" href="https://www.smu.gr/covid19/">Παρατηρητήριο
... Covid19
          </a>
        </li>
        <li class="ref-item">
          <a class="ref-link"
href="https://www.tandfonline.com/doi/pdf/10.1080/23748834.2020.1780074?needAcces
s=true">The impact of COVID-19 on public space: an
            early review of the emerging questions – design,
            perceptions and inequities
          </a>
        </li>
      </ul>

    </div>
  </main>

  <%- include("./partials/jsFiles.ejs") %>
</body>

</html>
```

6.6.7 404.ejs

```
<!DOCTYPE html>
<html lang="en">
<%- include("./partials/head.ejs") %>

<body>
  <%- include("./partials/navBar.ejs") %>
  <main role="main" class="container">
    <div class="error-div">
      <h1 class="ghost-h2">404</h1>
      <img class="ghost-image" src="../assets/images/Ghost-white.png">
      <h2 class="ghost-h2">This page does not exist</h2>
    </div>
  </main>
  <%- include("./partials/jsFiles.ejs") %>
```

```
</body>

</html>
```

6.6.8 partials

6.6.8.1 head.ejs

```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no" />
  <meta name="description" content="MyCovidApp" />
  <meta name="author" content="" />
  <link rel="shortcut icon" href="favicon.ico" />

  <title>MyCovidApp</title>
  <!-- GOOGLE FONTS -->
  <link rel="preconnect" href="https://fonts.gstatic.com" />
  <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@100;300;400&family
=Roboto:wght@300;500&display=swap" rel="stylesheet" />

  <!-- FONT AWESOME -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css" />

  <!-- BOOTSTRAP -->
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-
eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"
    crossorigin="anonymous"
  />

  <!-- Custom styles for this template -->
  <link href="../assets/css/style.css" rel="stylesheet" />
  <link href="../assets/css/leaflet.css" rel="stylesheet" />
</head>
```

6.6.8.2 navbar.ejs

```
<nav class="navbar navbar-expand-lg">
  <div class="container-fluid">
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarTogglerDemo02" aria-controls="navbarTogglerDemo02" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarTogglerDemo02">
      <img src="/assets/images/tree-logo.png" class="tree-logo" />
      <p class="app-title">Διαδικτυακή πλατφόρμα διερεύνησης: Δημόσιος χώρος &
Covid-19</p>
      <ul class="unordered-list ml-auto navbar-nav mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link" aria-current="page" href="/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/about">About</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

6.6.8.3 jsFiles.ejs

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js
" integrity="sha384-
/bQdsTh/da6pkI1MST/rWKFNjaCP5gBSY4sEBT38Q/9RBh9AH40zEOg7Hlq2THRZ"
crossorigin="anonymous"></script>
```

6.7 app.js

```
const express = require("express");
const morgan = require("morgan");
const mongoose = require("mongoose");
const _ = require("lodash");
const favicon = require("serve-favicon");

//express app
const app = express();

//port
const port = process.env.PORT || 3000;
```

```
//listen to port 3000
app.listen(port, () => {
  console.log(`Listening on port ${port}`);
});

//routes
const indexRoute = require("./routes/index");
app.use("/", indexRoute);

// register view engine
app.set("view engine", "ejs");


// connect to mongodb & listen for requests
const db = require("./models/db");

// middleware & static files
app.use(express.static("public"));
app.use(express.urlencoded({ extended: true }));
app.use(morgan("dev"));


// 404 page
app.use((req, res) => {
  res.status(404).render("404", { title: "404" });
});


module.exports = app;
```

6.8 gulpfile.js

```
let gulp = require('gulp'),
    spawn = require('child_process').spawn,
    minify = require('gulp-minify'),
    cleanCSS = require('gulp-clean-css'),
    rename = require('gulp-rename'),
    node;

/**
 * $ gulp server
 * launch the server. If there's a server already running, kill it
 */
gulp.task('server', async () => {
    if (node) node.kill()
    node = spawn('node', ['app.js'], { stdio: 'inherit' })
```

```
    node.on('close', (code) => {
        if (code === 8) {
            gulp.log('Error detected, waiting for changes...');
        }
    });
})

/**
 * $ gulp minify-js
 * compress all js files and output them in the dist folder
 */
gulp.task('minify-js', async () => {
    gulp.src('public/assets/js/*.js')
        .pipe(minify({
            noSource: true,
            ext: {
                src: '-debug.js',
                min: '.min.js'
            },
            ignoreFiles: ['.min.js']
        }))
        .pipe(gulp.dest('public/assets/dist'))
});

/**
 * $ gulp minify-css
 * compress all css files and output them in the dist folder
 */
gulp.task('minify-css', async () => {
    return gulp.src('public/assets/css/*.css')
        .pipe(cleanCSS({ compatibility: 'ie8' }))
        .pipe(rename({
            suffix: '.min'
        }))
        .pipe(gulp.dest('public/assets/dist'));
});

/**
 * $ gulp build
 * prepare all assets
 */
gulp.task('build', gulp.parallel([
    'minify-css',
    'minify-js',
]));
```

```
/**
 * $ gulp
 * start the development environment
 */
gulp.task('default', gulp.series('build', 'server', watch = () => {
    gulp.watch(['./public/assets/css/*.css'], gulp.series('minify-css'));
    gulp.watch(['./public/assets/js/*.js'], gulp.series('minify-js'));
    gulp.watch(['./app.js', './**/*.js', './**/*.html'], gulp.series('server'));
}));

// clean up if an error goes unhandled.
process.on('exit', () => {
    if (node) node.kill()
});
```

6.9 Ερωτηματολόγια

6.9.1 Ερωτηματολόγιο για την περίοδο πριν τον Covid-19

1. Πόσο συχνά επισκεπτόσασταν έναν δημόσιο χώρο πριν την καραντίνα ;

- Κάθε μέρα
- 3-4 φορές την εβδομάδα
- 1 φορά την εβδομάδα
- Λιγότερο από 3 φορές την εβδομάδα
- Καθόλου ή λιγότερο από 1 φορά το μήνα

2. Τι είδους δημόσιο χώρο επισκεπτόσασταν?

- Πλατεία
- Πάρκο
- Πεζόδρομο
- Άλσος
- Παραλία

3. Όταν βρισκόσασταν με την παρέα σας προτιμούσατε να πηγαίνετε σε:

- Δημόσιο χώρο (Πάρκο, Πλατεία κλπ)
- Μαγαζί
- Μαγαζί που έχει τραπεζοκαθίσματα πάνω σε δημόσιο χώρο
  σπίτι πάνω σε πλατεία

4. Ποιος είναι ο λόγος που επισκεπτόσασταν κάποιον δημόσιο χώρο;

- Άθληση
- Βόλτα με την παρέα σας
- Ατομική βόλτα/περπάτημα
- Βόλτα με κατοικίδια
- Βόλτα με την οικογένεια σας
- Ο χώρος αυτός εχει μαγαζιά

5. Ποιος ήταν ο μέσος χρόνος παραμονής σας σε έναν δημόσιο χώρο;

- Κάτω από μισή ώρα
- Μισή με μία ώρα
- Μία με δύο ώρες
- Πάνω από δύο ώρες

6. Ποιο ήταν το μέσο μετακίνησης σας προς τον δημόσιο χώρο;

- Μέσα Μαζικής Μεταφοράς
- Αυτοκίνητο
- Περπάτημα
- Ποδήλατο

7. Για ποιον/ποιους λόγους σας προσέλκυε ο δημόσιος χώρος που επισκεπτόσασταν πιο συχνά; (Σημειώστε με όσα συμφωνείτε)

- Βρίσκεται κοντά στην κατοικία σας
- Βρίσκεται σε μέρος που έχει όμορφο τοπίο
- Δεν σας αρέσει αλλά τον επισκέπτεστε επειδή συχνάζει εκεί η παρέα σας
- Δεν συχνάζει πολύ κόσμος
- Συχνάζει πολύ κόσμος
- Είναι ωραία διαμορφωμένος/Εχει καλές-επαρκείς υποδομές

6.9.2 Ερωτηματολόγιο για την περίοδο κατά τη διάρκεια του Covid-19

1. Πόσο συχνά επισκεπτόσασταν έναν δημόσιο χώρο κατά την διάρκεια της καραντίνας?

- Κάθε μέρα
- 3-4 φορές την εβδομάδα
- 1 φορά την εβδομάδα
- Λιγότερο από 3 φορές τον μήνα
- Καθόλου ή λιγότερο από 1 φορά το μήνα

2. Τι είδους δημόσιο χώρο επισκεπτόσασταν?

- Πλατεία
- Πάρκο
- Πεζόδρομο
- Άλσος
- Παραλία

3. Όταν κλείσανε τα μαγαζιά και βρισκόσασταν με την παρέα σας προτιμούσατε να πηγαίνετε σε

- Πλατεία
- Πάρκο
- Κάποιο σπίτι
- Δεν βρισκόμουν με την παρέα μου

4. Ποιος είναι ο λόγος που επισκεπτόσασταν κάποιον δημόσιο χώρο;

- Άθληση
- Συναναστροφή με άλλους ανθρώπους
- Ατομική βόλτα/περπάτημα
- Βόλτα με κατοικίδια
- Βόλτα με την οικογένεια σας
- Βόλτα με τους φίλους σας

5. Ποιος ήταν ο μέσος χρόνος παραμονής σας σε έναν δημόσιο χώρο;

- Κάτω από μισή ώρα
- Μισή με μία ώρα
- Μία με δύο ώρες
- Πάνω από δύο ώρες

6. Ποιο ήταν το μέσο μετακίνησης σας προς τον δημόσιο χώρο;

- Μέσα Μαζικής Μεταφοράς
- Αυτοκίνητο
- Περπάτημα

- Ποδήλατο

7. Για ποιον/ποιους λόγους σας προσέλκυε ο δημόσιος χώρος που επισκεπτόσασταν πιο συχνά; (Σημειώστε με όσα συμφωνείτε)

- Βρίσκεται κοντά στην κατοικία σας
- Βρίσκεται σε μέρος που έχει όμορφο τοπίο
- Δεν σας αρέσει αλλά τον επισκέπτεστε επειδή συχνάζει εκεί η παρέα σας
- Δεν συχνάζει πολύ κόσμος
- Συχνάζει πολύ κόσμος
- Είναι ωραία διαμορφωμένος/Εχει καλές-επαρκείς υποδομές

8. Η απαγόρευση κυκλοφορίας, μετά τις 21:00: επηρέασε τη συχνότητα επίσκεψης των δημόσιων χώρων ή/και το χρόνο παραμονής σας σε αυτούς;

- Ναι, επηρέασε την συχνότητα
- Ναι, έχει επηρέασε τον χρόνο παραμονής
- Ναι, έχει επηρέασε και τα δύο παραπάνω
- Όχι