ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΛΥΣΗ JAVASCRIPT WEB FRAMEWORKS ΜΕ ΣΚΟΠΟ ΤΗ ΣΥΣΤΗΜΑΤΟΠΟΙΗΣΗ ΤΩΝ ΕΝΔΕΔΕΙΓΜΕΝΩΝ ΕΠΙΛΟΓΩΝ

Διπλωματική Εργασία

του

Κατσινάρη Κωνσταντίνου

Θεσσαλονίκη, Μήνας 2019

ΑΝΑΛΥΣΗ JAVASCRIPT WEB FRAMEWORKS ΜΕ ΣΚΟΠΟ ΤΗ ΣΥΣΤΗΜΑΤΟΠΟΙΗΣΗ ΤΩΝ ΕΝΔΕΔΕΙΓΜΕΝΩΝ ΕΠΙΛΟΓΩΝ

Κατσινάρης Κωνσταντίνος

Πτυχίο Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας, 2017

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Κασκάλης Θεόδωρος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ηη/μμ/εεεε

Κασκάλης Θεόδωρος	Γεωργιάδης Χρήστος	Ξυνόγαλος Στυλιανός
.....	.....	.....

Κατσινάρης Κωνσταντίνος

.....

Περίληψη

Με τις ραγδαίες τεχνολογικές εξέλιξης στον χώρο της τεχνολογίας και συγκεκριμένα σε αυτόν του Web Development, καθημερινά, νέα JavaScript Frameworks κάνουν την εμφάνισή τους και προσφέρουν νέες προσεγγίσεις όσων αφορά την ανάπτυξη κώδικα. Επιλέγοντας ένα δείγμα από τα περισσότερο διαδεδομένα των τελευταίων 2-3 χρόνων, η συγκεκριμένη έρευνα, σε πρώτο στάδιο, συγκρίνει και καταγράφει τα χαρακτηριστικά τους. Κύριος σκοπός είναι η κατηγοριοποίηση τους, βάσει των χαρακτηριστικών αυτών, με αποτέλεσμα την εύρεση των δυνατών και αδύνατών τους σημείων. Τελικός στόχος της έρευνας είναι η συστηματοποίηση και δημιουργία μιας εφαρμογής, η οποία λαμβάνοντας υπ' όψη τις απαιτήσεις των χρηστών που επιθυμούν την ανάπτυξη ενός διαδικτυακού έργου λογισμικού, προτείνει σε αυτούς τα καταλληλότερα Frameworks στην συγκεκριμένη περίπτωση. Κατά την διάρκεια της έρευνας δημιουργούνται πίνακες συγκρίσεων, παραθέτονται στατιστικά στοιχεία και σημαντικές πληροφορίες για τα Frameworks που λαμβάνουν μέρος. Ακόμη δημιουργούνται κατάλληλα ερωτήματα προς απάντηση για τους χρήστες της εφαρμογής μέσω των οποίων εξάγονται τελικά οι προτάσεις της. Τέλος δημιουργούνται δένδρα αποφάσεων με τα ερωτήματα αυτά και μια λειτουργική διαδικτυακή εφαρμογή η οποία υλοποιεί και παρουσιάζει τα αποτελέσματα της έρευνας.

Λέξεις Κλειδιά: JavaScript, Web Development, Framework, JS, Web, Full-Stack, Front-End, Back-End, React, Vue.js, Angular, Library, Σύγκριση, Συστηματοποίηση Επιλογών.

Πρόλογος – Ευχαριστίες

Αρχικά θέλω να ευχαριστήσω τους γονείς μου για την στήριξη κάθε μορφής που μου παρείχαν κατά την διάρκεια των σπουδών μου συνολικά, αλλά ακόμη περισσότερο για την περίοδο συγγραφής της διπλωματικής εργασίας. Χωρίς την προτροπή τους δεν θα είχα φτάσει έως εδώ, στην ολοκλήρωση της, καθώς αυτή συνέπεσε με μια τέτοια περίοδο της ζωής μου που υπηρετούσα την στρατιωτική μου θητεία.

Επίσης, ο σημαντικότερος παράγοντας επιλογής της συγκεκριμένης εργασίας και γενικότερα της έναρξης της ενασχόλησης μου με τον γενικότερο κάδο του Web Development και της JavaScript, είναι ο καθηγητής μου και υπεύθυνος της συγκεκριμένης έρευνας, κύριος Κασκάλης Θεόδωρος. Μέσα από το μεταπτυχιακό μάθημά του, αγάπησα την συγκεκριμένη γλώσσα προγραμματισμού και ελπίζω να μπορέσω να ασχοληθώ περισσότερο με αυτήν στο μέλλον. Τον ευχαριστώ λοιπόν για τις γνώσεις και τα κίνητρα που μου έδωσε για περαιτέρω προσωπική βελτίωση.

Τέλος, θα ήθελα να εκφράσω ένα συνολικό ευχαριστώ σε όλους τους παράγοντες του μεταπτυχιακού αυτού προγράμματος, καθηγητές και γραμματεία, για την βοήθειά που μου πρόσφεραν σε κάθε ζήτημα που προέκυψε κατά την διάρκεια των σπουδών μου.

.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1 Εισαγωγή

1.1 Πρόβλημα – Σημαντικότητα του θέματος

Η συγκεκριμένη μελέτη θα ασχοληθεί με έναν κλάδο ο οποίος αναπτύσσεται, εξελίσσεται και αλλάζει με ραγδαίους ρυθμούς καθημερινά, καθώς διαμορφώνονται διαφορετικές απαιτήσεις με την τεχνολογία να δημιουργεί νέες προκλήσεις. Ο ευρύτερος κλάδος της μελέτης είναι το Web Development ή Ανάπτυξη Διαδικτυακών Εφαρμογών.

Με το διαδίκτυο να κυριαρχεί πλέον στην καθημερινότητα των ανθρώπων παγκοσμίως, οι διαδικτυακές εφαρμογές και οι ιστοσελίδες γενικότερα αποτελούν το απαραίτητο μέσο για την μετάδοση της εκάστοτε πληροφορίας από τις πηγές στους χρήστες ή και το αντίθετο. Το πλήθος των ιστοσελίδων/εφαρμογών με όμοια συμφέροντα είναι τεράστιο, με αποτέλεσμα να υπάρχει υψηλός ανταγωνισμός ανάμεσα τους. Γίνεται λοιπόν αντιληπτό πως η συνολική προσέγγιση στην ανάπτυξη μιας εφαρμογής, είναι ένα από τα βασικότερα κομμάτια, έτσι ώστε ο χρήστης να επιλέξει αυτήν την εφαρμογή έναντι μιας άλλης. Ως αποτέλεσμα, προκύπτουν συγκεκριμένες απαιτήσεις οι οποίες τελικά διαμορφώνουν αυτή την επιλογή του χρήστη. Συνήθως, η εφαρμογή που θα πετύχει την ικανοποίηση του μεγαλύτερου μέρους όλων αυτών των απαιτήσεων είναι αυτή που θα υπερισχύσει.

Υπάρχουν χιλιάδες εργαλεία που βοηθούν τους προγραμματιστές στην ανάπτυξη τέτοιων εφαρμογών. Χιλιάδες διαφορετικές προσεγγίσεις και τεχνολογίες χρησιμοποιούνται στο σύνολο του διαδικτύου. Ωστόσο οι περισσότερες τεχνολογίες έχουν η κάθε μία ξεχωριστό ενδιαφέρον. Η συγκεκριμένη μελέτη εστιάζει στον τομέα της γλώσσας προγραμματισμού ανάπτυξης διαδικτυακών εφαρμογών, JavaScript. Υπάρχουν διάφορες προσεγγίσεις και υλοποιήσεις βιβλιοθηκών κώδικα JavaScript (Frameworks), οι οποίες επιτυγχάνουν την απλούστευση πολλών διαδικασιών όπως: γρηγορότερη συγγραφή κώδικα, ταχύτητα εκτέλεσης, λιγότερες γραμμές κώδικα, ευκολότερη συντήρηση κώδικα κ.α. Η χρήση τους στην δημιουργία μεγάλης κλίμακας εφαρμογών είναι αναπόσπαστο κομμάτι από την εργαλειοθήκη των προγραμματιστών. Ωστόσο το πλήθος τους είναι αυτό που δυσκολεύει τελικά την επιλογή Framework που ο καθένας θα χρησιμοποιήσει.

Η σημαντικότητα της επιλογής του καταλληλότερου Framework σε κάθε περίπτωση και πάντα ανάλογα με τις απαιτήσεις και τα χαρακτηριστικά της εφαρμογής

που υλοποιείται, παίζει σπουδαίο ρόλο και είναι το βασικότερο κομμάτι της μελέτης και ανάλυσης αυτής της έρευνας.

1.2 Σκοπός – Στόχοι

Βασικός σκοπός της έρευνας είναι να ελαττώσει τον χαοτικό αριθμό του πλήθους των διαφορετικών JavaScript Framework συγκρίνοντας ένα δείγμα αυτών, με στόχο να προτείνει κάποια εξ' αυτών για την ανάπτυξη εφαρμογών με συγκεκριμένες απαιτήσεις.

Σημαντικό να αναφερθεί και να ξεκαθαριστεί είναι το γεγονός πως η συγκεκριμένη έρευνα δεν ακολουθεί κατά γράμμα και εξ’ ολοκλήρου της δομή και τον τρόπο πραγματοποίησης μιας ακαδημαϊκής έρευνας βασισμένη σε επίσημη βιβλιογραφία. Ο τρόπος που παρουσιάζονται και αντλούνται οι πληροφορίες για την υλοποίηση της, είναι μέσω δημοφιλών και έμπιστων ανά το διαδίκτυο πηγών, των οποίων τα θέματα που θίγονται είναι καθαρά προγραμματιστικά. Πιο συγκεκριμένα ένας στόχος της έρευνας είναι να συλλέξει πληροφορίες, εμπειρίες και απόψεις προγραμματιστών στο σύνολο του διαδικτύου για τις τεχνολογίες που θα αναλυθούν. Αυτή η προσέγγιση είναι απαραίτητη καθώς το αντικείμενο και οι επιμέρους ενότητες του αναπτύσσονται και εξελίσσονται καθημερινά. Κάνοντας λόγο για τεχνολογίες διαδικτύου και το πώς αυτές λειτουργούν στην πράξη, είναι πολύ σημαντικές οι εμπειρίες των ανθρώπων που τις χειρίζονται και αναπτύσσουν έργα λογισμικού με την βοήθειά τους. Στο τέλος της έρευνας το επιθυμητό αποτέλεσμα θα είναι η αξιοποίηση αυτής της γνώσης με στόχο την καλύτερη κατανόηση των δυνατοτήτων των τεχνολογιών.

Σημαντικές λύσεις-προτάσεις από την συγκεκριμένη έρευνα θα δοθούν στους προγραμματιστές διευκολύνοντας το έργο της ανάπτυξης λογισμικού και ξεκαθαρίζοντας κάποια πράγματα ως προς τα χαρακτηριστικά της κάθε διαφορετικής τεχνολογίας.

Ακόμη, στόχος είναι να εξαχθούν λογικά συμπεράσματα και δημιουργηθούν σωστές συγκρίσεις και κατηγοριοποιήσεις ανάμεσα στις τεχνολογίες που θα εξεταστούν. Αποτέλεσμα, τα πλεονεκτήματα και οι βέλτιστοι τρόποι εφαρμογής των τεχνολογιών να είναι ξεκάθαρα.

Τέλος στόχο αποτελεί το γεγονός της δημιουργία μιας εφαρμογής με την χρήση της γνώσης που θα εξαχθεί από την έρευνα, η οποία θα συμβουλεύει τους χρήστες -

προγραμματιστές ανάλογα με τις απαιτήσεις τους, για την καταλληλότερη επιλογή της τεχνολογίας που θα χρησιμοποιήσουν για να αναπτύξουν μια διαδικτυακή εφαρμογή.

1.3 Ερωτήματα – Υποθέσεις (εάν υπάρχουν)

- Πρέπει απαραίτητα να χρησιμοποιηθεί ένα JavaScript Framework για να αναπτυχθεί μια σύγχρονη Web εφαρμογή;
- Ποιό Framework πρέπει να επιλεγεί για ένα νέο Project;
- Ποιό είναι το καλύτερο JavaScript Framework;
- Μπορεί τελικά κάποιο να θεωρηθεί το καλύτερο όλων, ή καλύτερο από άλλα;
- Υπάρχουν συγκεκριμένοι παράγοντες που καθορίζουν και χαρακτηρίζουν την σχέση μεταξύ των Frameworks;
- Υπάρχουν συγκεκριμένοι παράγοντες που καθορίζουν την κατάλληλη επιλογή ενός Framework για την υλοποίηση ενός Project.

1.4 Διάρθρωση της μελέτης

Στο πρώτο κεφάλαιο αναφέρθηκαν εισαγωγικές πληροφορίες για τον στόχο-σκοπό και την σημαντικότητα της μελέτης που πρόκειται να πραγματοποιήθηκε. Διατυπώθηκαν επίσης ερωτήματα και υποθέσεις, στα οποία αναμένεται να δώσουν απαντήσεις τα επόμενα κεφάλαια και τα συμπεράσματα της έρευνας. Στην συνέχεια στο Κεφάλαιο 2 παρουσιάζονται τα αντικείμενα προς ανάλυση και συγκεκριμένα τα JavaScript Frameworks, πραγματοποιείται η επιλογή τους και παρουσιάζονται λεπτομερώς αρχικά πληροφορίες για την φύση και τον σκοπό του καθενός. Οι συγκεκριμένες πληροφορίες παρέχονται από τις επίσημες ιστοσελίδες των ίδιων των Framework αλλά και διαμορφώνονται μέσα από απόψεις και εμπειρίες προγραμματιστών της κοινότητας. Στην συνέχεια του Κεφαλαίου 2, πραγματοποιείται η δημιουργία κριτηρίων, η ανάλυση τους, η σύγκριση των Frameworks με βάση τα κριτήρια αυτά. Όσων αφορά το Κεφάλαιο 3, εκεί δημιουργούνται το τελικά συμπεράσματα για το κάθε Framework, με βάση την σύγκριση και τις πληροφορίες που προηγήθηκαν. Δημιουργείται αρχικά μια λίστα με τις βέλτιστες περιπτώσεις χρήσης για το κάθε ένα και στην συνεχεία διαμορφώνονται τα ανάλογα ερωτήματα προς απάντηση για τον χρήστη, μέσω των οποίων θα εξαχθούν οι τελικές αποφάσεις και προτάσεις για

την επιλογή του κατάλληλου Framework. Στο τέλος του κεφαλαίου δημιουργούνται επίσης δενδρικές δομές για τις 3 κατηγορίες Framework οι οποίες δείχνουν τις πλήρης διαδρομές για την τελική επιλογή του κάθε ενός. Στην συνέχεια, στο Κεφάλαιο 4, παρουσιάζεται η υλοποίηση της Web εφαρμογής που δημιουργήθηκε βασισμένη στην έρευνα και στα δένδρα αποφάσεων που προαναφέρθηκαν. Δίνονται λεπτομέρειες για τον τρόπο υλοποίησης της, οδηγίες για τις δυνατότητες που αυτή έχει, αλλά και εικόνες οι οποίες κάνουν περισσότερο κατανοητό τον τρόπο λειτουργίας της. Στο Κεφάλαιο 5, συλλέγονται τα βασικά συμπεράσματα της έρευνας. Παρουσιάζονται οι λύσεις στα προβλήματα που αναφέρθηκαν στο Κεφάλαιο 1, καθώς και προτάσεις για μελλοντικές επεκτάσεις για την συγκεκριμένη θεματική ενότητα. Τέλος αναφέρεται όλη η βιβλιογραφία και οι πηγές από τις οποίες αντλήθηκαν πληροφορίες για την συγγραφή και την υλοποίηση της συγκεκριμένης έρευνας.

2 Συλλογή πληροφοριών και στοιχείων για τα Frameworks

2.1 Επιλογή των JavaScript Frameworks

Αρχικά η έρευνα που πραγματοποιείται στην συγκεκριμένη εργασία βασίζεται στην προγενέστερη έρευνα των R. Benitte, S. Greif ,M. Rambeau (2018), ιδρυτών και κατόχων της διαδικτυακής σελίδας *stateofjs.com* (StateofJs, 2018a). Οι συγκεκριμένοι, κάθε χρόνο πραγματοποιούνε μια μαζική έρευνα/δημοσκόπηση σε προγραμματιστές από ολόκληρο τον κόσμο, με ερωτήσεις που έχουν να κάνουν για τα διάφορα Frameworks της JavaScript. Για το έτος 2018 στην έρευνα έλαβαν μέρος 20.268 προγραμματιστές από 153 διαφορετικές χώρες. Ενώ οι Η.Π.Α. κυριαρχούν στην έρευνα όπως αναμενόταν με το 24% των ερωτηθέντων, η Γερμανία και η Αυστραλία εκπροσωπούνται αρκετά καλά, με πάνω από το 5% των ερωτηθέντων. Φυσικά, θα πρέπει να αναφερθεί ότι παρόλο που έγιναν πολλές προσπάθειες να συγκεντρωθεί ένα μεγάλο σύνολο από προγραμματιστές για να λάβουν μέρος στην έρευνα, το κοινό εξακολουθεί να είναι μόνο ένα μικρό δείγμα της συνολικής κοινότητας JavaScript. Το κοινό της έρευνας συγκεντρώθηκε με τους εξής τρόπους (Greif, 2018):

- Email: 2033 συμμετέχοντες (10.04%)
- Twitter: 2062 συμμετέχοντες (10.18%)
- Reddit: 1043 συμμετέχοντες (5.15%)

- Slack: 557 συμμετέχοντες (2.75%)
- JavaScript Weekly: 529 συμμετέχοντες (2.61%)
- Hacker News: 468 συμμετέχοντες (2.31%)
- Medium: 285 συμμετέχοντες (1.41%)
- Facebook: 140 συμμετέχοντες (0.69%)
- Other/Unknown: 13272 συμμετέχοντες (65.54%)

Με βάση τα αποτελέσματα της έρευνας του stateofjs.com που προκύπτουν, επιλέχθηκαν τα πιο πολυαναφερόμενα Framework από τους προγραμματιστές, ως την βάση για την υλοποίηση της εργασίας. Με αυτό το κριτήριο επιλέχθηκαν προς ανάλυση 11 Front-End Frameworks, 4 Data-Layer Frameworks και 6 Back-End Frameworks. Οι ονομασίες τους παρουσιάζονται ανά κατηγορία στους παρακάτω πίνακες:

Πίνακας 2-1: Front End JavaScript Frameworks που συμμετέχουν στην έρευνα

React
Angular
Vue
Preact
Ember
Polymer
Svelte
Aurelia
Hyperapp
Backbone
Mithril

Πίνακας 2-2: Data Layer JavaScript Frameworks που συμμετέχουν στην έρευνα

Redux
GraphQL
Apollo
MobX

Πίνακας 2-3: Back End JavaScript Frameworks που συμμετέχουν στην έρευνα

Express
Next
Koa
Meteor
Sails
Feathers

2.2 Πληροφορίες και χαρακτηριστικά των JavaScript Frameworks

Σε αυτό το σημείο πραγματοποιείται συλλογή πληροφοριών από τα επίσημα Documentations των Frameworks, καθώς και από πηγές μέσα σε προγραμματιστικά φόρουμ με συμπεράσματα/απόψεις προγραμματιστών στο σύνολο του διαδικτύου. Επίσης, θα παρουσιαστούν και τα τρία αγαπημένα χαρακτηριστικά του κάθε Framework, σύμφωνα με τις απαντήσεις των χρηστών που συμπεριέλαβε η έρευνα του *stateofjs.com* η οποία παρουσιάστηκε παραπάνω. Με την συλλογή των εν λόγω πληροφοριών και την μελέτη της βιβλιογραφίας, μπορούν να εξαχθούν συμπεράσματα για την δημιουργία μιας αποτελεσματικότερης κατηγοριοποίησης και ομαδοποίησης των Frameworks.

2.2.1 Front End Frameworks

Όσων αφορά το Front-End κομμάτι, ξεχωρίζουν με βάση τις προηγούμενες συγκρίσεις και δεδομένα, τα τρία μεγάλα Frameworks της κατηγορίας (React, Angular, Vue.Js). Ωστόσο και τα υπόλοιπα που αναφέρονται παραπάνω χρησιμεύουν για την επίτευξη διαφορετικών στόχων, πολλές φορές με διαφορετικές, αποτελεσματικότερες και καινοτόμες μεθόδους (Hannah, 2018). Γι αυτό τον λόγο, η διαδικασία εύρεσης πληροφοριών για τα χαρακτηριστικά του καθενός ξεχωριστά, είναι ιδιαιτέρως σημαντική.

2.2.1.1 React

Είναι στην πραγματικότητα μια βιβλιοθήκη και όχι ένα Framework. Η ίδια η εταιρεία τονίζει πως η βιβλιοθήκη τους είναι κατάλληλη για την δημιουργία διεπαφών χρήστη (React, 2019).Υποστηρίζεται από την κοινότητα του Facebook πρωταρχικά,

αλλά είναι η πιο δημοφιλής βιβλιοθήκη/framework, στην γενικότερη κοινότητα των προγραμματιστών, σύμφωνα και με την προαναφερθείσα έρευνα του *stateofjs.com*. Η React επιτρέπει στους προγραμματιστές να δημιουργούν μεγάλες εφαρμογές ιστού, οι οποίες μπορούν να ανανεώνουν τα δεδομένα τους, χωρίς να φορτώνουν ξανά τη σελίδα από την αρχή, δηλαδή σε πραγματικό χρόνο. Ο κύριος σκοπός της React είναι να είναι γρήγορη και απλή. Λειτουργεί μόνο σε διεπαφές χρήστη σε εφαρμογή. Αυτό αντιστοιχεί στο View Layer του MVC μοντέλου (React, 2019). Η React έχει συμβάλλει σε μεγάλο βαθμό στην δημοφιλία του συναρτησιακού προγραμματισμού, χωρίς ωστόσο να είναι 100% συναρτησιακή (Tinning, 2016).

Βασικά χαρακτηριστικά:

- Virtual DOM.
- Το μεγαλύτερο μερίδιο της αγοράς.
- Υποστήριξη από το ίδιο το Facebook.
- Πολλές πηγές για εκμάθηση διαδικτυακά.
- Ύπαρξη συμπληρωματικών βιβλιοθηκών για γρηγορότερη και αποτελεσματικότερη ανάπτυξη εφαρμογών.
- Πολύ καλή επιλογή για Cross-Platform ανάπτυξη με την ύπαρξη της React Native.
- Ειδικεύεται στην δημιουργία Single Page εφαρμογών.
- Χρήση JavaScript για την διαχείριση/δημιουργία και της HTML πλευράς της εφαρμογής.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018b):

1. Κομψό στυλ προγραμματισμού και μοτίβων.
2. Πλούσιο οικοσύστημα εργαλείων.
3. Καλά εδραιωμένη επιλογή.

2.2.1.2 Angular

Ανήκει στα τρία μεγαλύτερα και δημοφιλή Framework. Είναι η δεύτερη και βελτιωμένη έκδοση της AngularJS και έχει αναπτυχθεί και υποστηρίζεται από την ίδια την Google (AngularJS, 2016). Πρόκειται για ένα πλήρως εξοπλισμένο Framework που παρέχει προεπιλεγμένες ρυθμίσεις για την ανάκτηση δεδομένων, τη διαχείριση

καταστάσεων και τη γλώσσα ανάπτυξης (Angular, 2019a). Το κυριότερο χαρακτηριστικό της Angular είναι η χρήση της TypeScript ως γλώσσα ανάπτυξης. Αυτό την έχει κάνει κατάλληλη για προγραμματιστές που προέρχονται από παραδοσιακές αντικειμενοστραφείς γλώσσες όπως Java και C#, καθώς η TypeScript εμπνέεται από αυτές (Angular, 2019b). Λαμβάνοντας υπ' όψη ότι πολλές μεγάλες εταιρείες έχουν ομάδες εξοικειωμένες με την Java και άλλες αντικειμενοστραφείς γλώσσες, μπορούμε να πούμε πως οι χρήστες τους οποίους στοχεύει η Angular είναι οι επιχειρήσεις αυτές.

Βασικά χαρακτηριστικά:

- Πλήρες πλαίσιο, με αποδεδειγμένα ισχυρά χαρακτηριστικά.
- Η TypeScript παρέχει μεγάλη ευκολία σαν γλώσσα, σε όσους έχουν background τον αντικειμενοστραφή προγραμματισμό.
- Ισχυρή υποστήριξη από την Google.
- Ύπαρξη ξεκάθαρων βέλτιστων πρακτικών.
- Αρκετά δύσκολη στην εκμάθηση της (StateofJs, 2018c).
- Όχι αρκετά ικανοποιητικά στατιστικά επιδόσεων όσων αφορά την εκκίνηση των εφαρμογών, όπως δείχνουν τα benchmarks (Meyghani, 2019)
- Μπορεί να αντιμετωπίσει μεγάλη πολυπλοκότητα που πολύ πιθανό να υπάρχει στις διεπαφές χρηστών.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018c):

1. Πλήρης λειτουργικά και πολύ ισχυρή.
2. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
3. Καλό documentation.

2.2.1.3 VueJs

Το συγκεκριμένο Framework διαφέρει από τα άλλα δύο μεγάλα της κατηγορίας (React, Angular), κυρίως στο γεγονός ότι δεν έχει την επίσημη υποστήριξη από κάποια μεγάλη εταιρεία. Αναπτύχθηκε από ένα και μόνο άτομο, και συντηρείται από ατομικές και εταιρικές δωρεές (You, 2014). Είναι ευκολότερο στην εκμάθηση του σε σχέση με την React και την Anglular, ωστόσο έχει πολλά κοινά χαρακτηριστικά και με τα δύο. Συγκριτικά με την React έχουν ομοιότητες στον χειρισμό του DOM, εκμεταλλευόμενα

το μοντέλο του Virtual DOM και δίνοντας έμφαση μόνο στο View κομμάτι αφήνοντας σε τρίτες βιβλιοθήκες τις λειτουργίες όπως Routing, HTTP Communications κτλ. Στις διαφορές μεταξύ τους συμπεριλαμβάνονται οι ελαφρώς βελτιωμένες επιδόσεις και μέγεθος της Vue έναντι της React, αλλά και ο διαφορετικός τρόπος προσέγγισης της δομής των αρχείων. Τα vue αρχεία περιλαμβάνουν 3 βασικές λειτουργίες (Templates, JavaScript, CSS), σε αντίθεση με την React που τα αρχεία της είναι καθαρά JavaScript με την εισαγωγή του JSX syntax για τα HTML elements. Όσων αφορά την Angular οι διαφορές είναι πολλές και οι ομοιότητες λίγες. Βασικές ομοιότητες έχουν κυρίως στην ταχύτητα, και στην κλιμάκωση των εφαρμογών (Vue.js, 2019).

Βασικά χαρακτηριστικά:

- Virtual DOM.
- Μικρό μερίδιο αγοράς.
- Αυξανόμενη δημοτικότητα και χρήση καθημερινά (StateofJs, 2018d).
- Καλό υλικό για την εκμάθησή της διαδικτυακά.
- Καλύτερη απόδοση συνολικά, συγκριτικά με τις δύο προηγούμενες (Krause, 2018).
- Πολύ εύκολη στην χρήση και την εκμάθηση (StateofJs, 2018d)
- Ειδικεύεται στην δημιουργία Single Page εφαρμογών.
- Στοχεύει κυρίως στο View Layer του MVC.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018d):

1. Μεγάλη ευκολία στην εκμάθησή του.
2. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
3. Καλό documentation.

2.2.1.4 Preact

Το PreactJS αναπτύχθηκε από τον Jason Miller. Το συγκεκριμένο Framework δημιουργήθηκε με στόχο την παροχή υψηλής απόδοσης, αποδοτικότητας μνήμης και στενής συμβατότητας με το API της React. Ακόμη κύριος στόχος ήταν το συνολικό μέγεθος της να είναι μικρό, κάτι το οποίο επιτεύχθηκε επιτυχώς με το συνολικό της μέγεθος τελικά να μην ξεπερνά τα 3kb (gzipped) (Budhani, 2017). Το Preact υιοθετείται ολοένα και περισσότερο στην κοινότητα με 2-3 ιστοτόπους υψηλής δημοτικότητας (Lyft,

Pepsi και Uber), που το χρησιμοποιούν ήδη με μεγάλη επιτυχία και βελτιωμένες επιδόσεις συγκριτικά με την προηγούμενη χρήση της React στις ιστοσελίδες τους. Παρόλο όμως, που το Preact έχει καλύτερη απόδοση εκκίνησης (φόρτωση σελίδας, για παράδειγμα) σε σχέση με την React, στα πιο πρόσφατα benchmarks η δεύτερη είναι πιο γρήγορη κατά την ενημέρωση του UI, αφού φορτωθεί η σελίδα (Eluwande, 2017).

Το Preact δεν έχει ως σκοπό να είναι μια ανακατασκευή της React. Υπάρχουν αρκετές διαφορές. Ωστόσο πολλές από αυτές τις διαφορές είναι ασήμαντες ή μπορούν να εξαλειφτούν πλήρως χρησιμοποιώντας το preact-compat, το οποίο είναι ένα λεπτό στρώμα πάνω από το Preact που προσπαθεί να επιτύχει 100% συμβατότητα με την React. Ο λόγος που το Preact δεν επιχειρεί να συμπεριλάβει όλα τα χαρακτηριστικά της React είναι για να παραμείνει μικρή και να εστιάσει σε συγκεκριμένα βασικά σημεία - διαφορετικά θα ήταν πιο λογικό απλά να χρησιμοποιηθεί η ίδια η React, η οποία είναι ήδη πολύ πολύπλοκη και καλά σχεδιασμένη (PreactJS, 2019). Όλα τα παραπάνω καθιστούν το Preact μια ιδανική επιλογή για κάποιον ο οποίος χρησιμοποιεί ήδη και είναι εξοικειωμένος με τις τεχνολογίες της React και η ταχύτητα εκτέλεσης, η απόδοση και το μέγεθος παίζουν σημαντικό ρόλο για την ομαλή λειτουργία της εφαρμογής του.

Βασικά χαρακτηριστικά:

- Μικρό σε μέγεθος.
- Καλές επιδόσεις.
- Πολύ καλό documentation.
- Συμβατό και με την React.
- Μικρότερη κοινότητα συγκριτικά με την React.
- Γρήγορο virtual DOM.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018e):

1. Γρήγορες επιδόσεις.
2. Απλή και μικρή σε μέγεθος.
3. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.

2.2.1.5 Ember

Το Ember αναπτύχθηκε από τον Yehuda Katz, ένας προγραμματιστής ο οποίος έχει συνεισφέρει σε πολλά έργα ανοιχτού κώδικα. Το συγκεκριμένο Framework

βασίζεται στο μοτίβο Model-View-ViewModel και έχει ένα πλούσιο σετ χαρακτηριστικών ενσωματωμένων στην βασική έκδοση. Πιο συγκεκριμένα η Ember είναι ένα front-end JavaScript Framework που έχει σχεδιαστεί για να βοηθήσει στην δημιουργία ιστοτόπων με πλούσιες και σύνθετες αλληλεπιδράσεις χρηστών. Αυτό επιτυγχάνεται παρέχοντας στους προγραμματιστές πολλά εργαλεία που είναι απαραίτητα για τη διαχείριση της πολυπλοκότητας στις σύγχρονες εφαρμογές ιστού (Ember, 2019).

Το Ember θέτει ως στόχο να προσφέρει μια ολοκληρωμένη λύση στο client-side μέρος μιας εφαρμογής. Αυτό έρχεται σε αντιδιαστολή με πολλά JavaScript Frameworks που παρέχουν λύσεις μόνο/κυρίως στο View Layer του MVC (Model-View-Controller) και προσπαθούν να αναπτυχθούν από εκεί. Το Ember κατακρίνεται συνεχώς κυρίως για το μεγάλο του μέγεθος, το οποίο έχει αρνητικό αντίκτυπο στην απόδοση. Θεωρείται επίσης ότι είναι δύσκολο στην εκμάθηση του αλλά και στο να χρησιμοποιείται με τον καλύτερο δυνατό τρόπο καθώς υπάρχουν πολύ αυστηροί κανόνες και τρόποι ανάπτυξης (Hannah, 2018)

Συμπερασματικά το Ember περιλαμβάνει ένα σύνολο εργαλείων που λειτουργούν μαζί για να παρέχουν μια ολοκληρωμένη λύση ανάπτυξης. Στόχος αυτών των εργαλείων είναι να βοηθήσουν τον προγραμματιστή συνολικά καθ' όλη την διάρκεια ανάπτυξης του έργου.

Βασικά χαρακτηριστικά:

- Ξεκάθαρες βέλτιστες πρακτικές.
- Παρέχει όλο το σύνολο των εργαλείων που χρειάζονται για την ανάπτυξη μιας πλήρους εφαρμογής.
- Μεγάλη σε μέγεθος συγκριτικά με άλλα Framework (Restuta, 2019).
- Δύσκολο στην εκμάθηση του (Conery, 2014).
- Δημοτικότητα η οποία μειώνεται συνεχώς (StateofJs, 2018f)

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018f)

1. Πλήρης λειτουργικά και πολύ ισχυρή.
2. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
3. Καλό documentation.

2.2.1.6 Polymer

Το Polymer είναι ένα Framework που υποστηρίζεται από την Google και επικεντρώνεται στην χρήση των Web Components, μια σχετικά καινούρια κατηγορία τεχνολογιών που αυτή την στιγμή δεν υποστηρίζονται 100% στα προγράμματα περιήγησης, ωστόσο καθημερινά ενσωματώνονται και σταδιακά υιοθετούνται από αυτά (Oliff, 2018). Το Polymer, μαζί με την εργαλειοθήκη Polymer App Toolbox, βοηθά τους προγραμματιστές να χρησιμοποιούν αυτές τις τεχνολογίες για την κατασκευή εφαρμογών ιστού. Σύμφωνα με την επίσημη ιστοσελίδα του Framework, παρέχει ένα σύνολο χαρακτηριστικών για τη δημιουργία προσαρμοσμένων html elements. Αυτές οι δυνατότητες έχουν σχεδιαστεί για να διευκολύνουν την ταχύτερη δημιουργία προσαρμοσμένων element που λειτουργούν σαν τα κλασικά DOM elements. Τα προσαρμοσμένα αυτά στοιχεία, μπορεί να διαχειριστούν με όλους τους γνωστούς τρόπους διαχείρισης των απλών elements όπως: να διαμορφώνονται και να ανταποκρίνονται σε αλλαγές των attributes και properties, να αλλάζει εσωτερικά η εξωτερικά το style τους κτλ. Τα βασικά εργαλεία του Polymer είναι (Polymer, 2019):

1) Προσαρμοσμένα στοιχεία: Δημιουργώντας ένα element αυτό μπορεί να συσχετιστεί κανονικά με ένα όνομα και μια κλάση. Το στοιχείο μπορεί να επανακληθεί όποια στιγμή είναι απαραίτητο.

2) Shadow DOM: Το Shadow DOM παρέχει μια τοπική, κλειστή δομή DOM για το κάθε προσαρμοσμένο element. Το Polymer μπορεί ακόμα να δημιουργήσει αυτόματα και να συμπληρώσει μια τέτοια δομή γι αυτό το element, από ένα DOM template.

3) Events: Το Polymer παρέχει μια συγκεκριμένη σύνταξη για την προσάρτηση των απαραίτητων Event Listeners στα παιδία του Shadow DOM.

4) Σύστημα δεδομένων: Το σύστημα δεδομένων του Polymer παρέχουν και πραγματοποιούν το Data binding σε attributes και properties.

Βασικά χαρακτηριστικά:

- Υποστήριξη και εκμετάλλευση των τελευταίων web τεχνολογιών.
- Ισχυρή υποστήριξη από την Google.
- Ελλιπής, προς το παρόν, υποστήριξη όλων αυτών των τεχνολογιών από τους browsers.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018g)

1. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
2. Απλό και μικρό σε μέγεθος.
3. Υποστήριξη από μια καλή ομάδα/εταιρεία.

2.2.1.7 Svelte

Το Svelte είναι μια πρωτοπόρα νέα προσέγγιση για την κατασκευή διεπαφών χρήστη. Ενώ τα παραδοσιακά Frameworks όπως η React και η Vue κάνουν το μεγαλύτερο μέρος της δουλειάς τους στο πρόγραμμα περιήγησης (web browser), η Svelte μεταφέρει αυτήν την λειτουργία, στην διαδικασία του compile που πραγματοποιείται, όταν δημιουργείται η εφαρμογή. Με λίγα λόγια η Svelte στην πραγματικότητα είναι ένας compiler. Πιο συγκεκριμένα, η Svelte δεν κάνει το compile του κώδικα κατά εκτέλεση της (run time), αλλά μετατρέπει την εφαρμογή σε JavaScript την στιγμή που αυτή δημιουργείται (build time). Αυτό σημαίνει ότι οι επιδόσεις της εφαρμογής θα είναι καλύτερες, καθώς και ότι δεν θα υπάρχουν καθυστερήσεις κατά την πρώτη φόρτωση της εφαρμογής (Svelte, 2019). Η Svelte αφαιρεί τις αφηρημένες δομές και τα abstractions για να παράγει Vanilla JavaScript που εκτελείται πολύ πιο γρήγορα, ειδικά σε κινητές συσκευές. Επειδή δεν υπάρχουν καθυστερήσεις, η Svelte μπορεί εύκολα να ενσωματωθεί σε μια υπάρχουσα εφαρμογή σταδιακά ή να μεταφερθούν σε αυτήν widgets ως αυτόνομα πακέτα που λειτουργούν οπουδήποτε. Τέλος το συγκεκριμένο Framework, αντί να χρησιμοποιεί τεχνικές όπως το virtual DOM, γράφει κώδικα που ενημερώνει λεπτομερώς το DOM όταν αλλάζει κάποια κατάσταση της εφαρμογής (Vepsäläinen, 2016). Σημαντική να αναφερθεί είναι η απλότητα λειτουργίας και χρήσης των αρχείων του Svelte. Το συγκεκριμένο Framework λειτουργεί με .svelte αρχεία τα οποία συνδυάζουν, JavaScript, CSS και HTML μέσα στο ίδιο αρχείο. Με αυτήν την προσέγγιση προκύπτουν νέα πλεονεκτήματα όπως, CSS styles τα οποία έχουν εμβέλεια μόνο στα Elements τα οποία αναφέρονται στο συγκεκριμένο αρχείο και δεν επηρεάζουν όλη την εφαρμογή κ.α.

Βασικά χαρακτηριστικά:

- Υψηλή ταχύτητα εκτέλεσης και μικρό μέγεθος (Schae, 2019).
- Ενσωματωμένες λειτουργίες Http Request, State Managment και Animations.
- Πολύ καλό documentation.

- Καινοτόμες τεχνολογίες και προσεγγίσεις.
- Μικρή, προς το παρόν, κοινότητα.
- Πολύ καινούριο σαν Framework, με αποτέλεσμα να μην υπάρχουν δεδομένα για την πλήρη αξιολόγηση της.

2.2.1.8 Aurelia

Η Aurelia, έχει δημιουργηθεί από τον Rob Eisenberg. Μπορεί να θεωρηθεί ως μια διαφορετική προσέγγιση της AngularJS όσο και του προηγούμενου Framework του Eisenberg, Durandal. Πριν από τη δημιουργία της Aurelia, ο Eisenberg ήταν μέλος της ομάδας ανάπτυξης της Google για την Anglular, φεύγοντας από αυτήν στα τέλη του 2014 καθώς διαφωνούσε με την κατεύθυνση της νέας Angular 2 (Hannah, 2018). Η Aurelia είναι ένα πλήρες Framework, παρέχοντας τις βασικότερες δυνατότητες όπως: dependency injection, templating, routing κ.α, ώστε να μην χρειάζεται να χρησιμοποιείται έξτρα ένα σύνολο από τρίτες βιβλιοθήκες για την δημιουργία μιας εφαρμογής. Εκτός από αυτό το ολοκληρωμένο σύνολο λειτουργιών, η ίδια η Aurelia παρέχει extra plugins για διεθνοποίηση, validation, virtualization UI και πολλά άλλα. Παρέχει επίσης ένα σύνολο δικών της εργαλείων για την παραγωγή και την κατασκευή εφαρμογών, ένα plugin για το πρόγραμμα περιήγησης και ένα VS Code plugin για την συγγραφή του κώδικα. Ωστόσο, οι χρήστες δεν υποχρεούνται να χρησιμοποιήσουν κάποιο από αυτά. Με τον τρόπο που η Aurelia λειτουργεί, είναι το μοναδικό Framework που δίνει τη δυνατότητα δημιουργίας components και δομών, χρησιμοποιώντας Vanilla JavaScript ή TypeScript. Το μεγαλύτερο πλεονέκτημα είναι πως γνωρίζοντας JS και HTML, υπάρχουν πολύ λίγα καινούρια πράγματα να μάθει κάποιος για να φτιάξει ακόμα και τις πιο σύνθετες εφαρμογές (Aurelia, 2019).

Βασικά χαρακτηριστικά:

- Ολοκληρωμένη λύση ανάπτυξης εφαρμογών.
- Γλώσσα ανάπτυξης Javascript ή TypeScript.
- Παρέχει όλα τα απαραίτητα εργαλεία για το σύνολο της ανάπτυξης.
- Μικρότερη κοινότητα συγκριτικά με μεγαλύτερα Frameworks.
- Ικανοποιητικό μέγεθος εφαρμογών.

2.2.1.9 HyperApp

Το συγκεκριμένο Framework είναι πολύ μικρό σε μέγεθος (μόλις 1 Kb) και ειδικεύεται και αυτό με την σειρά του στην δημιουργία διαδικτυακών διεπαφών. Το βασικό του πλεονέκτημα συγκριτικά με τα υπόλοιπα είναι ότι ακολουθεί συνολικά μια μινιμαλιστική προσέγγιση ακόμη και σε αυτά που χρειάζεται κάποιος να γνωρίζει για να ξεκινήσει την ανάπτυξη ενός project. Μεταβλητές καταστάσεις, μονόδρομη ροή δεδομένων κ.α, είναι τα χαρακτηριστικά της που συνδυάζονται σε μια πολύ μικρή βιβλιοθήκη. Ακόμη, ακολουθεί μια διαφορετική προσέγγιση, καθώς ο προγραμματιστής δηλώνει το τί θέλει να κάνει και όχι το πώς θα το κάνει. Η ίδια η εφαρμογή με την σειρά της θα βρει τον καλύτερη μέθοδο για να ενημερώσει το DOM σχετικά με τις αλλαγές. Τέτοιες διεπαφές χρήστη οδηγούν σε εφαρμογές οι οποίες είναι εύκολες στον έλεγχο για σφάλματα. Τέλος, το HyperApp προσφέρει ένα σημαντικό σύνολο από λειτουργίες όπως διαχείριση καταστάσεων και ένα σύγχρονο Virtual DOM, το οποίο υποστηρίζει fuctional components και view memorization, χωρίς την προσθήκη έξτρα βιβλιοθηκών (Bucaran, 2019).

Βασικά χαρακτηριστικά:

- Πολύ μικρό σε μέγεθος, 1kb.
- Virtual DOM.
- Έχει ένα σύνολο εργαλείων χρήσιμων στην ανάπτυξη, έτσι ώστε η ανάγκη χρήσης τρίτων βιβλιοθηκών είναι μικρή.
- Ικανοποιητικό μέγεθος της κοινότητας, που συνεχεία μεγαλώνει.

2.2.1.10 Backbone

Το Backbone δημιουργήθηκε από τον Jeremy Ashkenas το 2010, που είναι ο ίδιος ο οποίος έφτιαξε την CoffeeScript. Το συγκεκριμένο Framework στην σημερινή εποχή χρησιμοποιείται όλο και λιγότερο καθώς μετά από τόσα χρόνια έχει μεγάλους ανταγωνιστές. Πιο συγκεκριμένα η Backbone είναι μια μικρή βιβλιοθήκη που βοηθάει τους χρήστες να φτιάξουν και να ανανεώσουν εύκολα το front end κομμάτι μια εφαρμογής (Siddharth, 2011). Το συγκεκριμένο Framework είναι σημαντικό επειδή ήταν ένα από τα πρώτα που έδωσε συγκεκριμένη δομή στις front end εφαρμογές, εφαρμόζοντας το μοντέλο MVC. Σύμφωνα με το επίσημο documentation της εταιρείας: Το πιο σημαντικό πράγμα στο οποίο μπορεί να βοηθήσει το Backbone είναι η διατήρηση της επιχειρηματικής λογικής ξεχωριστά από τη διεπαφή χρήστη. Όταν τα δύο

εμπλέκονται, η αλλαγή είναι δύσκολη. Όταν η λογική δεν εξαρτάται από το περιβάλλον χρήστη, η διεπαφή γίνεται πιο εύκολη (BackboneJS, 2019). Τα τελευταία χρόνια το Backbone παρουσίασε μείωση της χρήσης του και πλέον οι λειτουργίες, οι επιδόσεις και η πολυπλοκότητα την οποία μπορούν να διαχειριστούν τα υπόλοιπα Framework, το κάνουν μια όχι και τόσο ιδανική επιλογή, αν και εξακολουθεί να διατίθεται στην τελευταία έκδοση του CMS Drupal.

Βασικά χαρακτηριστικά:

- Παρέχει συγκεκριμένη δομή για των κώδικα.
- Είναι ένα σταθερό Framework του οποίου η χρήση έχει εδραιωθεί με τον χρόνο.
- Δημοτικότητα που συνεχώς μειώνεται.
- Επιτακτικό στυλ προγραμματισμού, σε αντίθεση με όλα τα σύγχρονα Frameworks τα οποία δίνουν την δυνατότητα για δηλωτικό στυλ.

2.2.1.11 Mithril

Το Mithril είναι και αυτό από τα Frameworks με πολύ μικρό μέγεθος. Σε αντίθεση με την React και τα περισσότερα Framework, περιλαμβάνει ενσωματωμένες τις λειτουργίες Routing, και state management (MithrilJS, 2019). Θα μπορούσε να αναφερθεί πως το Mithril είναι περισσότερο μια βιβλιοθήκη/εργαλείο παρά ένα Framework. Στόχος του είναι να βοηθήσει τον χρήση στην σωστή οργάνωση του κώδικα της εφαρμογής, έτσι ώστε να είναι εύκολα συντηρήσιμος και ανανεώσιμος. Συντακτικά έχει πολλές ομοιότητες με την jQuery καθώς οι βασικές του εκφράσεις συμπεριλαμβάνουν το αντικείμενο 'm' το οποίο λειτουργεί με παρόμοιο τρόπο όπως το '$'. Με την χρήση του m στην JavaScript ο χρήσης μπορεί με γρήγορο τρόπο να δημιουργήσει σύνθετα HTML Elements, να τα διαχειριστεί αποτελεσματικά να περάσει σε αυτά δεδομένα κτλ.

Βασικά χαρακτηριστικά:

- Μικρό μέγεθος.
- Καλές επιδόσεις συγκριτικά με τα μεγάλα και δημοφιλή Frameworks.
- Ενσωματωμένες λειτουργίες για Routing και State Management.

2.2.2 Data Layer Frameworks

Τα περισσότερα από το προαναφερθείσα Front End Frameworks λειτουργούν και υλοποιούν κατά βάση το View κομμάτι των μοντέλων που υλοποιεί το καθένα, δηλαδή το σύνολο των λειτουργιών τους δεν περιέχει έξτρα συναρτήσεις για υλοποίηση διαφορετικών λειτουργιών που μπορεί να έχουν να κάνουν με το Routing, State Management, τα HTTP Communications, κ.α. Ως αποτέλεσμα, είναι πολλές φορές αναγκαία η χρήση πρόσθετων βιβλιοθηκών ή Framework τα οποία προσφέρουν λύσεις στα παραπάνω θέματα. Στο κομμάτι των δεδομένων, τα βασικότερα θέματα που χρίζουν αντιμετώπισης σε μια εφαρμογή είναι το state management (διαχείριση καταστάσεων), και ο τρόπος με τον οποίο η εφαρμογή θα επικοινωνήσει με APIs για την παροχή τους. (StateofJs, 2018h). Οι λειτουργίες αυτές μπορεί να είναι σχεδόν άχρηστες σε μια εφαρμογή μικρού μεγέθους όπου τα δεδομένα και τα χαρακτηριστικά της είναι περιορισμένα. Ωστόσο, καθώς η εφαρμογή αναπτύσσεται και μεγαλώνει σε μέγεθος και πολυπλοκότητα, η ύπαρξη ενός τρόπου για την παρακολούθηση της κατάστασης της εφαρμογής σε συγκεκριμένη στιγμή και ενός αποτελεσματικού τρόπου για την εισαγωγή δεδομένων είναι αναγκαία. Όσων αφορά τα states, γίνονται πολλές συζητήσεις για την αποτελεσματικότερη διαχείριση τους και τους τρόπους επίλυσης του προβλήματος αυτού. Ωστόσο οι περισσότεροι προγραμματιστές ξεκινούν ένα νέο project στο οποίο αμέσως κατά την εκκίνηση, του χρησιμοποιούν state management libraries όπως αυτές που θα παρουσιαστούν παρακάτω. Αυτή η τεχνική είναι τελείως λανθασμένη καθώς τα προβλήματα που επιλύουν οι συγκεκριμένες βιβλιοθήκες και οι περιπτώσεις στις οποίες η χρήση τους είναι αναγκαία, εμφανίζονται μόνο στις μεγάλου μεγέθους πολύπλοκες εφαρμογές (Wieruch, 2017).

2.2.2.1 Redux

Το Redux είναι μια βιβλιοθήκη διαχείρισης καταστάσεων για εφαρμογές JavaScript. Βοηθάει στην συγγραφή εφαρμογών που συμπεριφέρονται με συνέπεια, εκτελούνται σε διαφορετικά περιβάλλοντα (client, server και στο ίδιο το σύστημα για mobiles) και είναι εύκολο να ελεγχθούν μέσω testing. Εκτός από αυτό, παρέχει μια πολύ καλή προγραμματιστική εμπειρία, όπως η επεξεργασία κώδικα σε πραγματικό χρόνο, σε συνδυασμό με ένα debugger (Redux, 2019). Ο τρόπος λειτουργίας του Redux είναι απλός. Υπάρχει ένα κεντρικό 'κατάστημα' (store) που κρατά ολόκληρη την κατάσταση

(state) της εφαρμογής. Κάθε στοιχείο (component) μπορεί να έχει πρόσβαση στην αποθηκευμένη κατάσταση χωρίς να χρειάζεται να αντλήσει πληροφορίες και να επικοινωνήσει με άλλα στοιχεία (Ighodaro, 2018). Το Redux χρησιμοποιείται κατ' εξοχήν σε συνδυασμό με την React, ωστόσο μπορεί να χρησιμοποιηθεί για το state management σε συνεργασία με οποιοδήποτε άλλο Framework το οποίο υλοποιεί το κομμάτι view του MVC μοντέλου. Είναι μικρό (2kB, συμπεριλαμβανομένων των εξαρτήσεων), αλλά διαθέτει ένα μεγάλο οικοσύστημα πρόσθετων.

Βασικά χαρακτηριστικά:

- Προβλεπόμενες καταστάσεις (states).
- Κώδικας που συντηρείται εύκολα (Ighodaro, 2018).
- Χρήση για κατασκευή scalable εφαρμογών.
- Χρησιμοποιείται κυρίως σε μεγάλες και πολύπλοκες εφαρμογές (Educba, 2018)
- Εύκολη διαδικασία Debugging.
- Ισχυρό community πίσω από το Framework.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018i):

1. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
2. Καλά εδραιωμένη επιλογή.
3. Ισχυρό και αξιόπιστο, λιγότερο επιρρεπής σε σφάλματα κώδικας.

2.2.2.2 GraphQL

Το GraphQL είναι περισσότερο μια νέα τεχνολογία (πρότυπο) πρότυπο API, παρά Framework. Παρέχει μια πιο αποτελεσματική, ισχυρή και ευέλικτη εναλλακτική λύση από το REST (HowToGraphQL, 2019). Ο χρήστης πραγματοποιεί ερωτήματα και στην μεριά του Server αυτά εκτελούνται χρησιμοποιώντας ένα σύστημα που ορίζεται από τον χρήστη για το εκάστοτε σύνολο δεδομένων - μια ειδική χαρτογράφηση ουσιαστικά της δομής του εκάστοτε API. Το GraphQL δεν συνδέεται με κάποια συγκεκριμένη βάση δεδομένων ή μηχανή αποθήκευσης, αλλά αντιθέτως βασίζεται στον υπάρχοντα κώδικα και τα δεδομένα του χρήστη. Γι αυτό τον λόγο, μπορεί να χρησιμοποιηθεί πάνω σε οποιοδήποτε υπάρχων API, ή βάση δεδομένων είτε αυτή είναι MongoDB είτε SQL. Πιο συγκεκριμένα, ο χρήστης αρχικά περιγράφει και ορίζει τους

τύπους των δεδομένων που χρησιμοποίει. Στην συνέχεια όταν ζητήσει να του επιστραφούν συγκεκριμένα πεδία και τιμές από το API ή την βάση που χρειάζεται, το GraphQL θα του επιστρέψει τα δεδομένα σε μορφή, key και value για το πεδίο που αιτήθηκε (GraphQL, 2019).

Βασικά χαρακτηριστικά:

- Αποτελεσματικότερο API για συγκεκριμένα ερωτήματα σε σχέση με το REST.
- Εύκολη προσαρμογή στις γρήγορες εξελίξεις του development.
- Δημιουργήθηκε και υποστηρίζεται από το Facebook με ισχυρό community.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018j)

1. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
2. Αυξανόμενη δημοτικότητα.
3. Πολύ ισχυρά εργαλεία προγραμματισμού.

2.2.2.3 Apollo

Η πλατφόρμα Apollo GraphQL είναι μια υλοποίηση του GraphQL που βοηθά στην διαχείριση των δεδομένων που επιστρέφουν τα query της GraphQL, με στόχο την σωστή εμφάνισή τους στο UI της εφαρμογής. Μπορεί εύκολα να ενσωματωθεί και να τοποθετηθεί πάνω από τις υπάρχουσες υπηρεσίες που ήδη χρησιμοποιούνται σε μια εφαρμογή, συμπεριλαμβανομένων των REST API και των βάσεων δεδομένων. Είναι ουσιαστικά ο σύνδεσμος που ενώνει τον GraphQL Server με το Front-End κομμάτι. Το Apollo περιλαμβάνει δύο βιβλιοθήκες ανοιχτού κώδικα για τον πελάτη και το διακομιστή, μαζί με τα προγραμματιστικά εργαλεία που παρέχουν όλα όσα χρειάζονται για να την σωστή ανάπτυξη (Apollo, 2019).

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018k)

1. Καλό documentation.
2. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
3. Πλήρης λειτουργικά και πολύ ισχυρή.

2.2.2.4 MobX

Το MobX είναι μια απλή, κλιμακούμενη (scalable) και δοκιμασμένη λύση για το state managment. Μπορεί να χρησιμοποιηθεί ως αυτόνομη βιβλιοθήκη, αλλά οι περισσότεροι τη χρησιμοποιούν σε συνδυασμό με τη React. Ωστόσο μπορεί να χρησιμοποιηθεί και με άλλα View Frameworks (MobX, 2019). Είναι όμοια με το Redux που παρουσιάστηκε παραπάνω, ωστόσο έχουν συγκεκριμένες και σημαντικές διαφορές. Η βασική τους διαφορά είναι πως το MobX χρησιμοποιεί αντικείμενα που λειτουργούν σαν παρατηρητές, με αποτέλεσμα να αντιλαμβάνεται της αλλαγές στα δεδομένα αυτόματα, κάτι το οποίο πρέπει να γίνει χειροκίνητα στην Redux. Ακόμη, στην MobX μια κατάσταση μπορεί να ενημερωθεί με μια νέα τιμή, ενώ στην Redux η καταστάσεις παραμένουν αμετάβλητες, καθώς συνέχεια δημιουργούνται καινούριες καταστάσεις κατά την αλλαγή δεδομένων (Chandran, 2017). Τέλος η MobX είναι ευκολότερη στο να γίνει γρήγορα κατανοητή λόγω της έλλειψης χρήσης συναρτησιακού προγραμματισμού, ωστόσο σε θέματα debugging και όταν οι εφαρμογές γίνονται πολυπλοκότερες υστερεί έναντι της Redux (Educba, 2018).

Βασικά χαρακτηριστικά:

- Εύκολη στην εκμάθηση της.
- Κατάλληλη για μικρές εφαρμογές.
- Μικρό community συγκριτικά με τον ανταγωνιστή Redux.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018l):

1. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
2. Εύκολη στην εκμάθηση της.
3. Γρήγορη σε επιδόσεις.

2.2.3 Back-End Frameworks

Στην πλειοψηφία τους τα Frameworks που θα παρουσιαστούν παρακάτω, βασίζονται και εκμεταλλεύονται λειτουργικά, το Node.Js. Πιο συγκεκριμένα το Node.Js είναι ένα περιβάλλον εκτέλεσης JavaScript, στο οποίο ο χρήστης μπορεί να μεταφέρει την JavaScript σε server side επίπεδο, καθώς παραδοσιακά μέχρι τα προηγούμενα χρόνια αυτή χρησιμοποιούταν αποκλειστικά στο Front-End. Είναι σχεδιασμένη για να φτιάχνει κλιμακούμενες διαδικτυακές εφαρμογές. Η βασική της διαφορά με τις υπόλοιπες

τεχνολογίες ανάπτυξης εφαρμογών δικτύου, είναι πως οι διεργασίες της δεν στηρίζονται στα νήματα (threads) αλλά σε μια αρχιτεκτονική προγραμματισμού με βάση τα γεγονότα (events) με δυνατότητα χρήσης ασύγχρονων εισόδων/εξόδων (NodeJS, 2019).

2.2.3.1 Express

Το Express είναι ένα δωρεάν, ανοιχτού κώδικα Framework και θεωρείται ως η πιο δημοφιλής και διαδεδομένη λύση για το Node.Js. Πιο συγκεκριμένα ο χρήστης με την χρήση του Express μπορεί να δημιουργήσει εύκολα διαδικτυακές εφαρμογές καθώς είναι μικρό σε μέγεθος, ευέλικτο και παρέχει ένα ισχυρό σύνολο από λειτουργίες για web και mobile εφαρμογές. Μπορεί επίσης μπορεί να δημιουργήσει ισχυρά APIs εύκολα με μια πληθώρα λειτουργιών HTTP (Express, 2019).

Βασικά χαρακτηριστικά (tutorialspoint.com, 2019a):

- Επιτρέπει την ρύθμιση των Middlewares, έτσι ώστε να απαντούν σε HTTP αιτήματα.
- Ορίζει έναν πίνακα δρομολόγησης ο οποίος χρησιμοποιείται για την εκτέλεση διαφορετικών ενεργειών με βάση τη μέθοδο HTTP και τη διεύθυνση URL.
- Επιτρέπει τη δυναμική δημιουργία ιστοσελίδων HTML βασισμένη στο πέρασμα μεταβλητών.
- Εύκολη στην εκμάθησή της (Parody, 2019).
- Γρήγορη ανάπτυξη εφαρμογών με την χρήση της (Parody, 2019).

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018m)

1. Καλά εδραιωμένη επιλογή.
2. Εύκολη στην εκμάθησή της.
3. Πλούσιο οικοσύστημα λειτουργιών.

2.2.3.2 Next.Js

Το Next.Js θεωρείται ένα full stack JavaScript Framework που δημιουργήθηκε από τη Zeit. Κατατάχθηκε στην κατηγορία των back end Framework. Ωστόσο λειτουργικά πετυχαίνει ότι και τα περισσότερα front end Frameworks. Επιτρέπει την δημιουργία εφαρμογών οι οποίες όμως είναι server side rendered, και στατικές χρησιμοποιώντας τη React (Hayani, 2018a). Ο τρόπος λειτουργίας του, είναι παρόμοιος

με κάποιων CMS όπως το WordPress. Με τον ίδιο τρόπο που το WordPress εκτελεί PHP κώδικα για την δημιουργία των HTML αρχείων, η Next εκτελεί κώδικα JavaScript στον server, αυτός μεταφράζεται και δημιουργεί τα απαραίτητα αρχεία για την εφαρμογή (NextJS, 2019). Διαθέτει πολλά χαρακτηριστικά και πλεονεκτήματα, τα οποία το καθιστούν μια πολύ καλή επιλογή για την κατασκευή σύγχρονων εφαρμογών διαδικτύου. Δεν απαιτεί σχεδόν καμία ρύθμιση, καθώς προσφέρει λειτουργίες αυτόματου code splitting, routing κ.α. Τέλος δημιουργεί εφαρμογές έτοιμες για παραγωγή, διαθέτοντας μικρό μέγεθος και γρήγορη παραγωγή κώδικα (NextJS, 2019). Το συγκεκριμένο Framework δεν λειτουργεί βασιζόμενο στο Node.Js όπως το Express. Χρησιμοποιεί ένα αποκλειστικά δικό του σύστημα για την δημιουργία των εφαρμογών σε συνδυασμό με την χρήση της React.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018n):

1. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
2. Γρήγορες επιδόσεις.
3. Εύκολη στην εκμάθησή της.

2.2.3.3 Koa

Το Koa είναι ένα νέο Framework το οποίο σχεδιάστηκε από την ίδια ομάδα που ανέπτυξε το Express που παρουσιάστηκε παραπάνω και με την σειρά του η λειτουργία του βασίζεται στο Node.Js. Το συγκεκριμένο Framework στοχεύει να είναι μια μικρότερη σε μέγεθος, πιο 'εκφραστική', και συνολικά αξιόπιστη λύση για την ανάπτυξη εφαρμογών ιστού και APIs. Με την αξιοποίηση των λειτουργιών ασύγχρονων συναρτήσεων, το Koa επιτρέπει την εξάλειψη των callback και την σημαντική αύξηση του χειρισμού σφαλμάτων. Επίσης δεν συνδέει κανένα middleware στον πυρήνα του και παρέχει μια χρήσιμη σειρά από μεθόδους που καθιστούν τους διακομιστές γραφής γρήγορους και ευχάριστους στον χρήστη (KoaJS, 2019)

Βασικά χαρακτηριστικά (Parody, 2019):

- Μικρότερο σε μέγεθος από όλα τα υπόλοιπα Node.Js Frameworks.
- Μοντέρνο, χρησιμοποιώντας ECMAScript 6, και αξιοποιώντας όλα τα σύγχρονα Web χαρακτηριστικά.
- Εύκολος χειρισμός σφαλμάτων.

- Δεν χρησιμοποιεί callbacks, αλλά ασύγχρονη εκτέλεση.
- Κατάλληλη για μικρής πολυπλοκότητας εφαρμογές.
- Modular προσέγγιση, καθώς ο χρήστης μπορεί να συμπεριλάβει μόνο τις λειτουργίες που τον ενδιαφέρουν.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018ο):

1. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
2. Απλή και μικρή σε μέγεθος.
3. Γρήγορες επιδόσεις.

2.2.3.4 Meteor

Το Meteor είναι ένα Full-Stack MVC JavaScript Framework γραμμένο σε Node.JS για την ανάπτυξη σύγχρονων εφαρμογών ιστού και κινητής τηλεφωνίας. Περιλαμβάνει ένα σύνολο βασικών τεχνολογιών για την ανάπτυξη εφαρμογών που συνδέονται με πελάτες, ένα εργαλείο ανάπτυξης και ένα επιμελημένο σύνολο πακέτων από το Node.js και τη γενική κοινότητα της JavaScript. Το Meteor επιτρέπει την ανάπτυξη σε μία γλώσσα προγραμματισμού και συγκεκριμένα στην JavaScript, σε όλα τα περιβάλλοντα: Server εφαρμογών, πρόγραμμα περιήγησης ιστού και κινητή συσκευή. Το Meteor επικοινωνεί με δεδομένα, δηλαδή ο server στέλνει δεδομένα, όχι HTML κώδικα και αρχεία, και ο πελάτης (client) το μετατρέπει. Σημαντικό χαρακτηριστικό είναι πως ενσωματώνει στις λειτουργίες της τα καλύτερα μέρη της τεράστιας κοινότητας της JavaScript με έναν προσεκτικό και διεξοδικό τρόπο. Τέλος το Meteor παρέχει Full-Stack reactivity, επιτρέποντας στο UI να είναι σε πολύ καλή κατάσταση με ελάχιστη προγραμματιστική προσπάθεια (MeteorJS, 2019).

Βασικά χαρακτηριστικά (Wodehouse, 2016):

- Full-Stack.
- Γρήγορο σε ταχύτητα.
- Είναι reactive και δημιουργεί ενημερωμένα UIs σε πραγματικό χρόνο κάτι το οποίο είναι απαίτηση των σύγχρονων χρηστών εφαρμογών.
- Υποστηρίζεται από ένα αρκετά μεγάλο σύνολο της κοινότητας.
- Έχει πολλές βιβλιοθήκες και πακέτα για να επεκτείνει τη λειτουργικότητά του.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018p):

1. Πλήρης λειτουργικά και πολύ ισχυρή.
2. Καλό documentation.
3. Εύκολη στην εκμάθησή της.

2.2.3.5 Sails

Το Sails είναι ένα δημοφιλές MVC Framework για το Node.Js το οποίο βασίζεται στο Express, και έχει σχεδιαστεί για να μιμηθεί το γνωστό μοτίβο MVC των Frameworks όπως το Ruby on Rails, αλλά με την υποστήριξη των απαιτήσεων των σύγχρονων εφαρμογών. Το μοτίβο αυτό έχει να κάνει με APIs που βασίζονται σε δεδομένα με μια κλιμακούμενη αρχιτεκτονική προσανατολισμένη στις υπηρεσίες. Η γλώσσα ανάπτυξης είναι η JavaScript και ένα από τα θετικά της είναι πως μπορεί να συνδεθεί και να υποστηρίξει οποιαδήποτε βάση δεδομένων. Η Sails επιτρέπει στους προγραμματιστές να δημιουργούν το τελικό σημείο RESTful API χωρίς να γράφουν κώδικα. Ο αυτόματα δημιουργούμενος κώδικας μπορεί να τροποποιηθεί και να προσαρμοστεί αργότερα για να ταιριάζει σε συγκεκριμένες επιχειρηματικές ανάγκες. Τέλος το Sails έρχεται με ενσωματωμένη υποστήριξη για WebSockets με το Socket.io. Ωστόσο, επιτρέπει την χρήση οποιουδήποτε Framework για το Front-End κομμάτι (SailsJS, 2019).

Βασικά χαρακτηριστικά (Bouchefra, 2018):

- Καλή οργάνωση κώδικα.
- By default υποστήριξη των WebSockets.
- Χρήση με οποιοδήποτε βάση δεδομένων.
- Καλό documentation.
- Δύσκολο στην εκμάθηση.
- Αυτόματα δημιουργούμενος κώδικας για controllers, models και routes.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018q):

1. Εύκολη στην εκμάθησή της.
2. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
3. Πλήρης λειτουργικά και πολύ ισχυρή.

2.2.3.6 FeathersJS

Το Feathers είναι ένα πλήρους λειτουργικότητας, αλλά ταυτόχρονα μικρό σε μέγεθος JavaScript Framework για την δημιουργία εφαρμογών Ιστού. Είναι και αυτό βασισμένο και κατασκευασμένο στην Node.Js. Στον πυρήνα του, το Feathers είναι ένα σύνολο εργαλείων και ένα πρότυπο αρχιτεκτονικής που διευκολύνει τη δημιουργία κλιμακωτών REST APIs και εφαρμογών πραγματικού χρόνου. Με το Feathers, μπορούν να δημιουργηθούν πρωτότυπα μέσα σε λίγα λεπτά και εφαρμογές έτοιμες για παραγωγή μόλις μέσα σε λίγες ημέρες. Αυτό επιτυγχάνεται καθώς το Feathers λειτουργεί ως ο κώδικας που ενώνει κάποιες πολύ ισχυρές και δοκιμασμένες τεχνολογίες ανοιχτού κώδικα, προσθέτοντας μικρά abstractions και εισάγοντας μια αρχιτεκτονική εφαρμογών που είναι πιο εύκολη στην κατανόηση, τη συντήρηση και την κλιμάκωση, από την παραδοσιακή αρχιτεκτονική MVC (FeathersJs, 2019).

Βασικά χαρακτηριστικά (Nwamba, 2019):

- Συνεργάζεται και ενσωματώνεται εύκολα με κάθε Front-End Framework.
- Χρησιμοποιεί μια universal σχεδίαση, καθώς ο χρήστης μπορεί να γράψει σε JavaScript για Front-End, Node.Js Back-End και React Native για κινητές εφαρμογές.
- Χρησιμοποιεί την ECMAScript 6 ενσωματώνοντας όλες τις νέες τεχνολογίες.
- Λειτουργεί καλά με όλες τις βάσεις δεδομένων καθώς είναι data agnostic.
- Real time εφαρμογές.
- Μικρές σε μέγεθος και αξιόπιστες εφαρμογές.

Αγαπημένα χαρακτηριστικά των χρηστών (StateofJs, 2018r):

1. Κομψό στυλ προγραμματισμού και χρήσης μοτίβων.
2. Καλό documentation.
3. Απλή και μικρή σε μέγεθος.

2.3 Συγκριτική Ανάλυση των JavaScript Frameworks

Σε αυτό το στάδιο θα πραγματοποιηθεί σύγκριση όμοιων χαρακτηριστικών της κάθε ομάδας των Frameworks, έτσι ώστε να μπορέσουν να προκύψουν συγκεκριμένα

κριτήρια σύγκρισης και έπειτα κατηγοριοποίησης τους. Κατά την διαδικασία αυτή συγκεντρώνονται όλες οι παραπάνω πληροφορίες που παρουσιάστηκαν στην προηγούμενη ενότητα. Δημιουργούνται πίνακες σύγκρισης όλων αυτών των χαρακτηριστικών καθώς και λεπτομερής περιγραφή του κάθε χαρακτηριστικού. Όσων αφορά τους πίνακες, με κόκκινο πράσινο χρώμα σημειώνεται ένα Framework όταν ικανοποιεί στο μεγαλύτερο βαθμό του κάποιο κριτήριο, με πορτοκαλί χρώμα όταν ικανοποιεί σε μέτριο βαθμό το κριτήριο, και με κόκκινο όταν δεν ικανοποιεί αρκετά, σε σύγκριση πάντα με τα υπόλοιπα Framework, το κριτήριο.

2.3.1 Front-End Frameworks

Η μεγαλύτερη από τις τρείς κατηγορίες που αναλύονται σε μέγεθος αλλά και σε φήμη (Front-End), περιέχει ένα σύνολο από Frameworks τα οποία στην πλειοψηφία τους έχουν όμοια χαρακτηριστικά. Ωστόσο στις λεπτομέρειες φαίνεται πως το κάθε ένα έχει κάτι ξεχωριστό που το διαφοροποιεί από τα υπόλοιπα, με αποτέλεσμα όλα τα Frameworks να είναι διαφορετικά μεταξύ τους. Στον παρακάτω πίνακα τα κριτήρια σύγκρισης πάρθηκαν κατά βάση από την προηγούμενη ενότητα παρουσίασης των χαρακτηριστικών του κάθε ενός. Σημαντική παρατήρηση είναι πως τα κριτήρια του Πίνακα 2-4, χωρίζονται σε δύο βασικές κατηγορίες. Η πρώτη κατηγορία περιέχει τα πρώτα 12 κριτήρια σύγκρισης, τα οποία είναι κατά βάση, τεχνικά χαρακτηριστικά. Τα συγκεκριμένα χαρακτηριστικά είναι αυτά που ενδιαφέρουν και γίνονται κυρίως κατανοητά και αντιληπτά από άτομα που είναι ήδη εξοικειωμένα με των προγραμματισμό διαδικτύου και τις προγραμματιστικές έννοιες. Προγραμματιστές οι οποίοι ήδη ασχολούνται και έχουν χρησιμοποιήσει ένα η περισσότερα Frameworks στο παρελθόν θα μπορέσουν να κατανοήσουν καλύτερα την σύγκριση αυτών των κριτηρίων. Ο πίνακας για αυτή την κατηγορία συμπληρώθηκε κυρίως με βάση το documentation του κάθε Framework. Από την άλλη πλευρά υπάρχει και μια δεύτερη κατηγορία κριτηρίων, τα οποία είναι κατανοητά, ενδιαφέρουν και θα βοηθήσουν τον κάθε ένα που θέλει να ασχοληθεί με τον προγραμματισμό διαδικτύου σε όποιο επίπεδο και αν βρίσκεται. Κριτήρια όπως η δημοτικότητα, η δυσκολία εκμάθησης, αλλά και το μερίδιο της αγοράς, είναι κάποια από τα οποία απαρτίζεται η συγκεκριμένη ενότητα. Με την σύγκριση και την ανάλυση τους, το άτομο μπορεί να πάρει καταλληλότερες αποφάσεις για τις τεχνολογίες με τις οποίες θέλει να ασχοληθεί και να επενδύσει σε αυτές.

Πίνακας 2-4: Σύγκριση χαρακτηριστικών των Front-End Frameworks

Κριτήρια\Frameworks	React	Vue	Angular	Preact	Ember	Polymer	Svelte	Aurelia	Hyperapp	Backbone	Mithril
Cross-Platform	x	x	x	-	-	x	x	-	-	-	-
ECMAScript 6 +	x	x	x	x	x	x	x	x	x	x	x
TypeScript	-	x	x	-	-	x	-	x	-	-	x
Web Components	-	-	-	-	-	x	-	x	-	-	-
Server Side Rendering	Extra Library	Extra Library	Extra Library	-	Extra Library	-	Extra Library	Core Feature	Extra Library	-	Extra Library
View/Templating	x	x	x	x	x	x	x	x	x	x	x
Routing	-	-	x	-	x	-	-	x	-	x	x
HTTP Communication	-	-	x	-	x	-	x	x	-	-	x
Data/State Management	-	-	x	-	x	-	x	-	x	-	x
Animations	-	-	x	-	x	-	x	x	-	-	x
Επιδόσεις											
Μέγεθος											
Εκμάθηση											
Δημοτικότητα											
Διαδικτυακά Μαθήματα											
Κοινότητα											
Υποστήριξη από μεγάλη εταιρεία	Facebook	-	Google	-	-	Google	-	-	-	-	-
Μερίδιο Αγοράς											
Documentation											

2.3.1.1 Ανάλυση Κριτηρίων

Cross-Platform

Ο τομέας της Cross-Platform ανάπτυξης εφαρμογών απασχολεί όλο και περισσότερο την κοινότητα και τους προγραμματιστές καθώς όλες οι εταιρίες ζητούν την χρήση τεχνολογιών οι οποίες κάνουν μια εφαρμογή διαθέσιμη σε κάθε είδους πλατφόρμα και συσκευή που μπορεί να διαθέτει ο χρήστης. Παραδοσιακά η ανάπτυξη μιας εφαρμογής Web και μιας Native εφαρμογής για κινητές συσκευές, είναι μια τελείως διαφορετική υπόθεση με την χρήση τελείως διαφορετικών τεχνολογιών. Από την μια τεχνολογίες HTML5 για το Web και από την άλλη τεχνολογίες Android Development ή iOS για τις κινητές συσκευές. Την σημερινή εποχή ωστόσο πολλά Frameworks προσφέρουν τεχνολογίες και εργαλεία τα οποία βοηθούν τον προγραμματιστή με την χρήση της JavaScript να δημιουργήσει με ευκολία και Native εφαρμογές για κινητά. Πολλές φορές μπορούν ολόκληρα components τα οποία έχουν δημιουργηθεί για την Web εφαρμογή να χρησιμοποιηθούν ξανά και για την Native. Γενικότερα μια ήδη υλοποιημένη Web εφαρμογή μπορεί με λίγες τροποποιήσεις και αλλαγές να μετατραπεί σε Native (Appsee, 2019). Αυτή την στιγμή τα μεγαλύτερα JavaScript Frameworks προσφέρουν συγκεκριμένα εργαλεία το καθένα για την επίτευξη αυτού του σκοπού. Το πιο γνωστό εργαλείο είναι της React, το React Native με το οποίο ο χρήστης μπορεί να δημιουργήσει εφαρμογές για κινητές συσκευές μέσω κώδικα React. Ωστόσο αυτό μπορεί να χρησιμοποιηθεί σε συνδυασμό και με τα υπόλοιπα Frameworks άλλων εταιριών που προσφέρουν εργαλεία για Cross-Platform ανάλυση. Επίσης υπάρχουν εργαλεία για την επίτευξη αυτού του σκοπού, γνωστότερα των οποίων είναι NativeScript, Flutter και Ionic. Πιο συγκεκριμένα όσων αφορά το NativeScript, πολλά από τα γνωστά JavaScript Frameworks έχουν δημιουργήσει πλήρη συμβατότητα με αυτό (Vue, Angular, Svelte*)* (NativeScript, 2019). Με τα εργαλεία αυτά ο χρήστης μπορεί να αναπτύξει Native εφαρμογές σε JavaScript.

ECMAScript 6+

Η ECMAScript είναι η script γλώσσα που καθιερώθηκε να χρησιμοποιείται για την συγγραφή κώδικα JavaScript. Δημιουργήθηκε για την καθιέρωση μιας ενιαίας γλώσσας γραφής για την JavaScript η οποία είναι η πιο διαδεδομένη και χρησιμοποιημένη υλοποίηση της έως τώρα. Το 2015 με την άφιξη της 6ης έκδοσης της γλώσσας προστέθηκαν λειτουργίες οι οποίες πραγματοποιούν διαδικασίες γρηγορότερα και ευκολότερα (WikiPedia, 2019a). Σημαντικότερες προσθήκες είναι τα Arrow Functionς και η εισαγωγή των let και const μεταβλητών (W3Schools, 2019a). Τα συγκεκριμένα Frameworks που εξετάζονται υποστηρίζουν στο σύνολό τους την ECMAScript 6.

TypeScript

Η TypeScript δημιουργήθηκε από την Microsoft και επί της ουσίας είναι ένα αυστηρά συντακτικό υπερσύνολο της JavaScript. Το βασικό της χαρακτηριστικό είναι η προσθήκη static typing στην γλώσσα. Δημιουργήθηκε για την γρηγορότερη κατασκευή μεγάλης κλίμακας εφαρμογών και μπορεί να χρησιμοποιηθεί για την client side αλλά και server side ανάπτυξη σε Node.js. Περιέχει όλα τα χαρακτηριστικά της ECMAScript 6 και όντως ένα υπερσύνολο της JavaScript καταλαβαίνει και τον απλό JavaScript κώδικα στον οποίο γίνεται και compile στο τέλος (WikiPedia, 2019b). Η JavaScript παραδοσιακά υλοποιεί την δυναμικά με τις μεταβλητές κάτι το οποίο σημαίνει πως δεν γνωρίζει τον τύπο της κάθε μεταβλητής ή αντικειμένου μέχρι να εκτελεστεί ο κώδικας. Με την TypeScript και την στατική προσέγγιση ο προγραμματιστής μπορεί να επωφεληθεί στην φάση του debugging αλλά και στην καλύτερη οργάνωση του κώδικα. Συμπληρωματικά παρέχει μια Object Oriented προσέγγιση καθώς υποστηρίζει καλυτέρα και αποτελεσματικότερα σε σύγκριση με την απλή JavaScriptτην δημιουργία κλάσεων, interfaces και την έννοια της κληρονομικότητας (tutorialspoint.com, 2019b). Όλα αυτά τα χαρακτηριστικά και ο Object Oriented χαρακτήρας καθιστούν την γλώσσα ιδανική για κάποιον ο οποίος έχει ένα τέτοιο προγραμματιστικό υπόβαθρο, του είναι οικίες γλώσσες όπως η Java ή η C++ και θέλει να ασχοληθεί με ένα JavaScript Project.

Web Components

Είναι γνωστό πως η δημιουργία επαναχρησιμοποιήσιμου κώδικα είναι μια καλή ιδέα για τους προγραμματιστές με πολλά θετικά. Τα Web Components είναι μια σουίτα διαφορετικών τεχνολογιών που επιτρέπουν την δημιουργία επαναχρησιμοποιήσιμων προσαρμοσμένων components - με τη λειτουργικότητά τους να υπάρχει ξεχωριστά μακριά από τον υπόλοιπο κώδικα - και την χρήση τους στις εφαρμογές ιστού. Οι τεχνολογίες αυτές είναι (MDN web docs, 2019)

- Custom Elements: Ένα σύνολο από JavaScript APIs που επιτρέπουν στον χρήστη να ορίσει custom elements και τη συμπεριφορά τους, τα οποία στη συνέχεια μπορούν να χρησιμοποιηθούν όπως αυτός επιθυμεί με σε ένα User Interface.
- Shadow DOM: Ένα σύνολο από JavaScript APIs για την επισύναψη ενός shadow DOM tree σε ένα στοιχείο και τον έλεγχο της συναφούς λειτουργικότητας του. Στο συγκεκριμένο element η διαδικασία του rendering πραγματοποιείται ξεχωριστά από το κύριο έγγραφο DOM. Με αυτόν τον τρόπο, μπορούν να διατηρηθούν τα χαρακτηριστικά ενός στοιχείου ιδιωτικά, ώστε να μπορεί να εφαρμοστεί JavaScript και CSS σε αυτά, χωρίς τον φόβο σύγκρουσης με άλλα μέρη του εγγράφου.
- HTML templates: Τα στοιχεία <template> και <slot> επιτρέπουν την συγγραφή markup templates που δεν εμφανίζονται στην τελική rendered σελίδα. Αυτά μπορούν στη συνέχεια να επαναχρησιμοποιηθούν πολλές φορές ως βάση για τη δομή ενός προσαρμοσμένου στοιχείου.

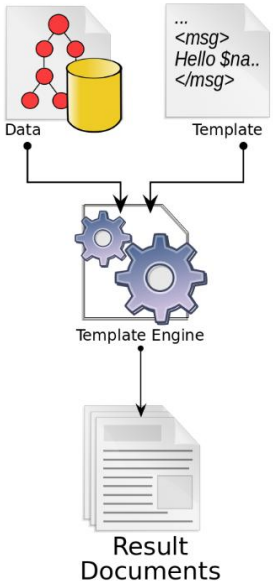
Οι συγκεκριμένες τεχνολογίες ωστόσο δεν έχουν ακόμη 100% υποστήριξη από όλους τους γνωστούς Internet Browsers (webcomponents.org, 2019), με αποτέλεσμα η χρήση τους την συγκεκριμένη χρονική περίοδο να μην είναι η σωστότερη επιλογή. Ωστόσο καθημερινά όλα και περισσότερες λειτουργίες ενσωματώνονται στους Browsers, έτσι είναι σημαντικό ένα JavaScript Framework το οποίο θα επιλέξει ο χρήστης να μπορεί να διαχειριστεί και να έχει προβλέψει για την λειτουργία των Web Components. Το πιο ισχυρό Front-End JavaScript Framework Που ενσωματώνει 100% όλες αυτές τις τεχνολογίες αλλά και περιέχει δική του βιβλιοθήκη με Components και λειτουργίες για αυτά είναι το Polymer.

Server Side Rendering

Η συγκεκριμένη τεχνική χρησιμοποιούνταν για την δημιουργία Ιστοσελίδων πριν την εμφάνιση όλων αυτών των JavaScript Framework τα οποία παρουσιάζονται στην συγκεκριμένη μελέτη. Με την ύπαρξη αυτών των Framework η διαδικασία του rendering μεταφέρθηκε από τον server στον client αφού πλέον η Google έχει την δυνατότητα να 'διαβάζει' JavaScript, δουλεύοντας αρκετά καλά τις περισσότερες φορές. Τα βασικά πλεονεκτήματα της συγκεκριμένης τεχνικής είναι, οι γρήγοροι χρόνοι rendering της σελίδας σε πραγματικό χρόνο μετά την πρώτη φόρτωση και η πλούσια διαδραστικότητα με τον χρήστη και το UI καθώς όταν πρέπει να ενημερωθεί οποιοδήποτε element της σελίδας δεν απαιτείται η πλήρης ανανέωση της σελίδας. Εκτελείται η διαδικασία του rendering μόνο στο πεδίο στο οποίο πραγματοποιήθηκε μια αλλαγή (Vega, 2017). Ωστόσο η ίδια η Google επισημαίνει πως με την συγκεκριμένη τεχνική υπάρχει η πιθανότητα να μην πραγματοποιηθεί σωστά το ranking της σελίδας με αποτέλεσμα πιθανά προβλήματα στο SEO (Manjunath, 2019). Από την άλλη, με την γνωστή διαδικασία του server side rendering επιτυγχάνεται υψηλή ταχύτητα της πρώτης φόρτωσης της σελίδας και ένας αποτελεσματικότερος τρόπος για το SEO. Ωστόσο η λύση είναι η δημιουργία νέων τεχνικών που να περιέχουν τα θετικά χαρακτηριστικά των δύο μεθόδων που προαναφέρθηκαν. Οι βασικότερες αυτών είναι: 1) η τεχνική του Pre-Rendering, με τον server να στέλνει μια 'εικόνα' της ιστοσελίδας στην μηχανή αναζήτησης έτσι ώστε να πραγματοποιηθεί σωστά η διαδικασία του ranking, αλλά ταυτόχρονα να αξιοποιηθούν και όλα τα θετικά του client side rendering, 2) Isomorphic JavaScript, με την τεχνική αυτή η οποία συνίσταται και από την ίδια την Google, ο client και η μηχανή αναζήτησης λαμβάνουν μια pre-rendered HTML μορφή της ιστοσελίδας αρχικά με αποτέλεσμα η πρώτη φόρτωση να είναι γρήγορα και το SEO να είναι εξασφαλισμένο (server side rendering). Έπειτα όλη λειτουργικότητα είναι διαστρωματωμένη επάνω σε αυτή την σελίδα έτσι ώστε να μπορούν να εκμεταλλευθούν όλα τα πλεονεκτήματα του client side rendering (Burkholder, 2019). Συμπερασματικά πολλά Framework έχουν επίσημες δικές του βιβλιοθήκες για την πραγματοποίηση αυτών των λειτουργιών, οι οποίες παίζουν σημαντικό ρόλο στην συνολική απόδοση της ιστοσελίδας.

View/Templating

Το JavaScript templating αναφέρεται στην μέθοδο δέσμευσης δεδομένων του client που εφαρμόζεται με τη γλώσσα JavaScript. Αυτή η προσέγγιση έγινε δημοφιλής χάρη στην αυξημένη χρήση της JavaScript, την αύξηση των δυνατοτήτων επεξεργασίας πελατών και την τάση να αναθέτουν υπολογισμούς στο πρόγραμμα περιήγησης του client (Tate, 2015). Ένα templating engine που χρησιμοποιείται από τα εκάστοτε Framework είναι υπεύθυνο να συνδέσει το μοντέλο δεδομένων, με τον template κώδικα και να τα εξάγει με ένα συγκεκριμένο τρόπο (Εικόνα 2-1). Παράδειγμα ο κώδικα της εικόνας 2-2.



Εικόνα 2-1: Λειτουργία Template Engine (Tate, 2015)

```
You have an input:
<h1>{{helloWorld}}></h1> // where helloWorld is an expression

You have a compiler:
The compiler takes the expressions we write in our input and turns
them into functions

You have an output:
The output is the called value from the function

<h1>Hello World</h1>
```

Εικόνα 2-2: Παράδειγμα λειτουργίας του Templating

Τα Framework που παρουσιάζονται χρησιμοποιούν στο σύνολό τους τεχνικές templating, πολλές φορές διαφοροποιημένες μεταξύ τους αλλά πετυχαίνοντας το ίδιο αποτέλεσμα.

Routing

Ένας δρομολογητής (Router) JavaScript είναι ένα βασικό συστατικό στα περισσότερα front-end Framework. Είναι το κομμάτι του λογισμικού που είναι υπεύθυνο για να οργανώσει τις καταστάσεις της εφαρμογής, και να πραγματοποιήσει τις εναλλαγές μεταξύ των διαφορετικών views. Για παράδειγμα, ο δρομολογητής θα πραγματοποιήσει την διαδικασία του rendering, αρχικά την οθόνη σύνδεσης και όταν η σύνδεση είναι επιτυχής, θα πραγματοποιήσει τη μετάβαση στην οθόνη υποδοχής του χρήστη. Με λίγα λόγια ο δρομολογητής είναι υπεύθυνος για την προσομοίωση μεταβάσεων μεταξύ εγγράφων παρακολουθώντας αλλαγές στη διεύθυνση URL. Όταν το έγγραφο επαναφορτωθεί ή η διεύθυνση URL τροποποιηθεί με κάποιο τρόπο, θα εντοπίσει αυτήν την αλλαγή και θα αποδώσει το view που σχετίζεται με τη νέα διεύθυνση URL. Οι συγκεκριμένες λειτουργίες είναι πολύ σημαντικές σε εφαρμογές οι οποίες ξεφεύγουν από το απλό μοντέλο των Single Page Applications (Esteban, 2017). Παρόλο που η ύπαρξη των λειτουργιών αυτών είναι σχεδόν αναπόσπαστο κομμάτι των σημερινών εφαρμογών, πολλά από τα Framework που παρουσιάζονται δεν περιέχουν στο βασικό τους πακέτο εγκατάστασης έναν δρομολογητή και απαιτούν την χρήση extra πακέτων.

HTTP Communication

Ακόμη ένα σημαντικό κομμάτι για την λειτουργία μιας εφαρμογής είναι η δυνατότητα του να πραγματοποιεί HTTP requests κατά τα οποία θα αιτείται ή θα αποστέλλει δεδομένα από ή σε κάποιο Server. Η ίδια η JavaScript έχει κάποιες γνωστές μεθόδους για την πραγματοποίηση αυτών των λειτουργιών, ωστόσο για τις περισσότερες απαιτείται και πάλι η ύπαρξη τρίτων πακέτων που αναλαμβάνουν το συγκεκριμένο έργο (Hayani, 2018b). Πολλά Framework έχουν ενσωματωμένες λειτουργίες και υλοποιήσεις για το HTTP Communication το οποίο πρέπει να πραγματοποιηθεί από μια εφαρμογή, αποτέλεσμα να αποδεσμεύουν τον χρήστη από την χρήση τρίτων βιβλιοθηκών.

Data/State Management

Όταν γίνεται λόγος για το state management (διαχείριση καταστάσεων), αναφερόμαστε κυρίως στο πώς διαχειριζόμαστε τα δεδομένα που κυκλοφορούν ανάμεσα στα Components μιας εφαρμογής JavaScript. Μια τυπική πρόκληση που δημιουργήθηκε

κατά την ανάπτυξη πολλών εφαρμογών είναι η διατήρηση των διαφόρων τμημάτων της εφαρμογής και της διεπαφής του χρήστη, συγχρονισμένων. Συχνά, οι αλλαγές στην κατάσταση (state) πρέπει να αντικατοπτρίζονται σε πολλαπλά Components, και καθώς μεγαλώνει η εφαρμογή, αυτή η πολυπλοκότητα αυξάνεται. Μια κοινή λύση είναι η χρήση των events έτσι ώστε διάφορα μέρη της εφαρμογής να γνωρίζουν πότε έχει αλλάξει κάτι. Μια άλλη προσέγγιση είναι να κρατείται η κατάσταση στο εσωτερικό του ίδιου του DOM ή ακόμα και να την αντιστοιχίσετε σε ένα global αντικείμενο στο παράθυρο. Σήμερα με όλα τα διαφορετικά JavaScript Frameworks έχουνε δημιουργηθεί αρκετές βιβλιοθήκες όπως το Redux το VueX και το NgRx για να διευκολυνθεί η διαδικασία της διαχείρισης καταστάσεων. Οι περισσότερες από αυτές έχουν συνδυαστεί με την ταυτόχρονη χρήση ενός συγκεκριμένου Front End Framework, Redux με React, Vuex με το Vue, NgRx με την Angular, ωστόσο η κάθε μια μπορεί να δουλέψει σε συνεργασία μετα περισσότερα Front End Frameworks (Oki, 2019). Βασική διαφορά είναι πως από τα τρία προαναφερθέντα Frameworks μόνο η Angular περιέχει στο βασικό της πακέτο αυτή την βιβλιοθήκη. Τα περισσότερα Frameworks έρχονται χωρίς κάποιο εργαλείο για το Data Management.

Animations

Η ύπαρξη των animations είναι ένα πολύ σύνηθες φαινόμενο στις σημερινές ιστοσελίδες. Με την συγκεκριμένη τεχνική, ο τρόπος παρουσίασης του περιεχόμενο της σελίδας γίνεται πιο ενδιαφέρον και ταυτόχρονα βελτιώνεται η εμπειρία του χρήστη. Ωστόσο για να επιτευχθεί κάτι τέτοιο είναι αρκετά πολύπλοκο σαν διαδικασία καθώς πρέπει να δημιουργηθούν animations τα οποία να λειτουργούν ομαλά σε όλους τους browsers (Chinnathambi, n.d.). Για τον λόγο αυτό υπάρχουν και σε αυτή την περίπτωση πολλές τρίτες βιβλιοθήκες οι οποίες δημιουργούν εύκολα και σύντομα, animations για τα διαφορετικά τμήματα της εφαρμογής. Ωστόσο πολλά σύγχρονα Front End Frameworks περιέχουν στο βασικό τους πακέτο και αυτό το χαρακτηριστικό, όντας αυτόνομα και ανεξάρτητα από τρίτες βιβλιοθήκες.

Επιδόσεις

Όσων αφορά τις επιδόσεις των εφαρμογών, για την σύγκριση τους χρησιμοποιήθηκαν οι μετρήσεις από την Ιστοσελίδα του Stefan Krause (Krause, 2018). Το συγκεκριμένο benchmark στα JavaScript Frameworks προσπαθεί να συγκρίνει την απόδοση των πλαισίων ιστού, μετρώντας τη χρονική διάρκεια εκτέλεσης για διαφορετικές λειτουργίες σε έναν μεγάλο πίνακα, όπως τη διάρκεια δημιουργίας του πίνακα, την ενημέρωση, την επιλογή ή την αφαίρεση μιας γραμμής και την προσθήκη σειρών. Τα αποτελέσματα για τα Frameworks τα οποία εξετάζει η συγκεκριμένη έρευνα παρουσιάζονται στον πίνακα 2-5 και πίνακα 2-6.

Πίνακας 2-5: Διάρκεια εκτέλεσης διάφορων διαδικασιών

Duration in milliseconds ± standard deviation (Slowdown = Duration / Fastest)

Name	polymer-v2.0.0-non-keyed	aurelia-v1.3.0-keyed	preact-v8.2.6-keyed	vue-v2.5.16-keyed	svelte-v2.9.7-keyed	angular-v6.1.0-keyed	react-v16.4.1-keyed	mithril-v1.1.1-keyed	hyperapp-v1.2.9-keyed	ember-v3.3.0-keyed
create rows Duration for creating 1000 rows after the page loaded.	204.7 ±11.9 (1.4)	182.8 ±8.7 (1.3)	174.6 ±9.3 (1.2)	182.1 ±7.6 (1.3)	220.4 ±4.9 (1.5)	185.2 ±10.2 (1.3)	180.5 ±7.3 (1.2)	170.5 ±7.6 (1.2)	145.3 ±5.1 (1.0)	406.8 ±15.8 (2.8)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	58.6 ±1.7 (1.0)	167.2 ±1.9 (2.9)	157.3 ±4.5 (2.7)	158.8 ±2.7 (2.7)	231.6 ±3.3 (4.0)	161.2 ±2.7 (2.8)	157.3 ±2.0 (2.7)	156.7 ±3.3 (2.7)	162.0 ±3.4 (2.8)	269.1 ±23.6 (4.6)
partial update Time to update the text of every 10th row (with 5 warmup iterations) for a table with 10k rows.	141.8 ±3.9 (2.1)	66.8 ±3.5 (1.0)	96.2 ±3.0 (1.4)	156.4 ±9.8 (2.3)	75.1 ±3.8 (1.1)	68.8 ±3.7 (1.0)	81.9 ±2.7 (1.2)	134.9 ±4.2 (2.0)	288.8 ±18.9 (4.3)	134.9 ±5.9 (2.0)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	10.0 ±2.2 (1.0)	9.6 ±4.2 (1.0)	10.5 ±3.5 (1.0)	10.6 ±2.0 (1.0)	10.7 ±1.0 (1.0)	7.9 ±4.3 (1.0)	10.3 ±2.1 (1.0)	8.7 ±2.5 (1.0)	16.0 ±2.2 (1.0)	8.7 ±1.9 (1.0)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	12.9 ±7.1 (1.0)	21.8 ±4.7 (1.4)	23.1 ±2.9 (1.4)	20.0 ±2.9 (1.3)	20.4 ±4.4 (1.3)	105.8 ±1.8 (6.5)	106.5 ±1.9 (6.7)	107.4 ±1.5 (6.7)	25.5 ±2.7 (1.6)	122.0 ±2.9 (7.6)
remove row Duration to remove a row. (with 5 warmup iterations).	42.2 ±1.3 (1.0)	48.0 ±1.1 (1.1)	49.3 ±0.6 (1.2)	54.2 ±2.2 (1.3)	48.2 ±1.0 (1.1)	47.1 ±3.0 (1.1)	49.6 ±0.8 (1.2)	50.2 ±2.1 (1.2)	60.1 ±4.6 (1.4)	55.7 ±1.0 (1.3)
create many rows Duration to create 10,000 rows	1,963.8 ± 41.1 (1.3)	1,739.9 ± 53.1 (1.2)	1,852.0 ± 51.6 (1.3)	1,603.2 ± 34.8 (1.1)	2,376.0 ± 40.7 (1.6)	1,693.9 ± 70.1 (1.1)	1,935.4 ± 33.6 (1.3)	1,519.1 ± 71.8 (1.0)	1,474.5 ± 35.9 (1.0)	2,931.9 ± 42.9 (2.0)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	306.1 ±3.9 (1.3)	240.3 ±6.0 (1.0)	271.7 ±3.8 (1.1)	342.5 ±6.0 (1.4)	354.4 ±11.8 (1.5)	243.3 ±6.3 (1.0)	268.6 ±6.9 (1.1)	301.1 ±11.0 (1.3)	541.9 ±23.7 (2.3)	403.7 ±32.5 (1.7)
clear rows Duration to clear the table filled with 10.000 rows.	176.1 ±3.7 (1.0)	201.3 ±6.8 (1.1)	194.1 ±2.1 (1.1)	191.9 ±6.1 (1.1)	183.5 ±4.1 (1.0)	263.9 ±3.0 (1.5)	175.4 ±4.1 (1.0)	182.7 ±1.8 (1.0)	264.1 ±5.8 (1.5)	203.1 ±3.6 (1.2)
slowdown geometric mean	1.20	1.25	1.32	1.41	1.43	1.53	1.53	1.60	1.65	2.16

Πίνακας 2-6: Σύγκριση μετρικών εκκίνησης εφαρμογής

Startup metrics (lighthouse with mobile simulation)

Name	polymer-v2.0.0-non-keyed	aurelia-v1.3.0-keyed	preact-v8.2.6-keyed	vue-v2.5.16-keyed	svelte-v2.9.7-keyed	angular-v6.1.0-keyed	react-v16.4.1-keyed	mithril-v1.1.1-keyed	hyperapp-v1.2.9-keyed	ember-v3.3.0-keyed
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	3,003.4 ± 0.6 (1.5)	3,441.1 ± 1.3 (1.8)	1,952.7 ± 0.4 (1.0)	2,252.7 ± 0.2 (1.2)	1,877.7 ± 0.7 (1.0)	3,278.5 ± 4.0 (1.7)	2,477.6 ± 0.6 (1.3)	2,027.4 ± 0.4 (1.1)	1,877.2 ± 0.2 (1.0)	5,105.5 ± 0.8 (2.7)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	115.1 ± 1.3 (7.2)	202.5 ± 5.1 (12.7)	16.0 ± 0.0 (1.0)	55.1 ± 1.5 (3.4)	16.0 ± 0.0 (1.0)	235.2 ± 8.5 (14.7)	65.6 ± 2.3 (4.1)	16.0 ± 0.0 (1.0)	16.0 ± 0.0 (1.0)	419.9 ± 6.6 (26.2)
main thread work cost total amount of time spent doing work on the main thread. includes style/layout/etc.	577.7 ± 11.0 (2.1)	778.4 ± 8.7 (2.8)	316.3 ± 61.8 (1.1)	420.6 ± 67.5 (1.5)	277.7 ± 4.5 (1.0)	679.3 ± 5.3 (2.4)	466.0 ± 3.9 (1.7)	385.3 ± 63.8 (1.4)	413.9 ± 5.6 (1.5)	884.2 ± 7.6 (3.2)
total byte weight network transfer cost (post-compression) of all the resources loaded into the page.	355,330.0 ± 0.0 (2.4)	449,530.0 ± 0.0 (3.0)	155,626.0 ± 0.0 (1.0)	215,445.0 ± 0.0 (1.4)	150,230.0 ± 0.0 (1.0)	365,497.0 ± 0.0 (2.5)	251,915.0 ± 0.0 (1.7)	173,932.0 ± 0.0 (1.2)	148,748.0 ± 0.0 (1.0)	742,719.0 ± 0.0 (5.0)

Μέγεθος

Σχετικά με το κριτήριο του μεγέθους, πραγματοποιήθηκε μια σύγκριση μεταξύ ίδιων εφαρμογών υλοποιημένων μια φορά για το κάθε Framework από αυτά που εξετάζονται. Στον πίνακα 2-7 παρουσιάζονται τα αποτελέσματα για το μέγεθος του Real Word App (RealWordApp, 2019), μια υλοποίηση ενός σύνθετου App με παρόμοια χαρακτηριστικά με αυτά των μεγάλου μεγέθους εφαρμογών οι οποίες αναπτύσσονται από εταιρίες. Στον επόμενο πίνακα 2-8 εμφανίζονται το αποτελέσματα του Todo MVC (TodoMVC, 2019), το οποίο υλοποιεί ένα απλό todo app, με τον ίδιο τρόπο, αλλά αυτή την φορά πρόκειται για μια μικρού μεγέθους εφαρμογή. Η μέτρηση του μεγέθους των εφαρμογών έγινε μόνο για την αρχική σελίδα σαν μια single page εφαρμογή για το Real World App έτσι τα αποτελέσματα για την συγκεκριμένη εφαρμογή είναι ενδεικτικά. Συνολικά οι μετρήσεις έγινε χρησιμοποιώντας τα programming tools του Google Chrome Browser. Μέσω των tools και στην καρτέλα Network ο χρήστης μπορεί να μάθει για το μέγεθος των συνολικών resources που φορτώνονται στην σελίδα.

Πίνακας 2-7: Μέγεθος Real World App (Complex App)

Front-End Frameworks	Real Words App Size
React + Redux	1 Mb
Vue.js	735 Kb
Angular2	1 Mb
Preact	-
Ember	1,2 Mb
Polymer	-
Svelte	289 Kb
Aurelia	905 Kb
Hyperapp	555 Kb
Backbone	-
Mithril	664 Kb

Πίνακας 2-8: Μέγεθος Todo MVC (Plain App)

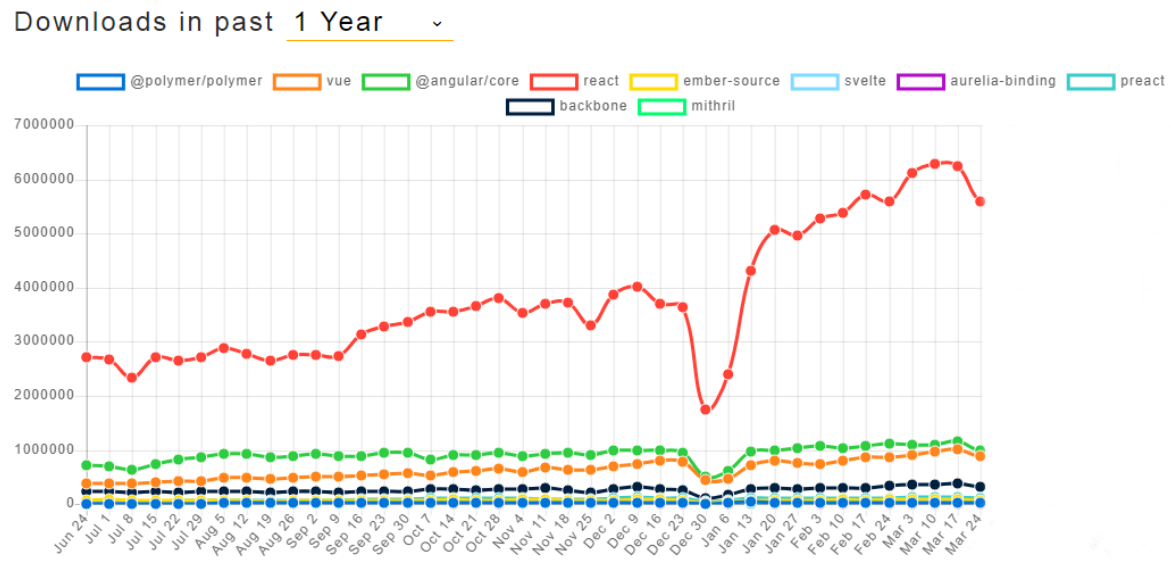
Front-End Frameworks	TodoMVC Size
React	1,2 Mb
Vue.js	302 Kb
Angular2	1,6 Mb
Preact	-
Ember	603 Kb
Polymer	271 Kb
Svelte	-
Aurelia	663 Kb
Hyperapp	-
Backbone	508 Kb
Mithril	177 Kb

Εκμάθηση

Το συγκεκριμένο κριτήριο αφορά τον χρόνο και τον κόπο που απαιτείται από έναν χρήστη για να εξοικειωθεί και να μπορεί να λύνει προβλήματα και να αναπτύσσει χωρίς πρόβλημα εφαρμογές με την χρήση του κάθε Framework. Η καμπύλη εκμάθηση είναι ο τρόπος που αυτό αναφέρεται και παρουσιάζεται μέσα στα διάφορα άρθρα στο σύνολο του διαδικτύου. Μετά από έρευνα στο documentation, άρθρα αλλά και μαθήματα διαδικτυακά τα οποία στοχεύουν στην εξοικείωση του χρήστη με το κάθε Framework, δημιουργήθηκαν τα αποτελέσματα του πίνακα 2.10. Ωστόσο και πάλι τα αποτελέσματα αυτά είναι ενδεικτικά. Για παράδειγμα εάν κάποιος έχει ήδη εξοικειωθεί με το React η μετάβαση του στο Preact θα είναι πολύ εύκολη καθώς οι διαφορές είναι ελάχιστες. Με λίγα λόγια η δυσκολία εκμάθησης του κάθε Framework εξαρτάται κυρίως από το προγραμματιστικό επίπεδο το οποίο βρίσκεται το κάθε άτομο ξεχωριστά. Γενικότερα πάντως τα αποτελέσματα που παρουσιάζονται στον πίνακα πετυχαίνουν μια πολύ αντιπροσωπευτική απεικόνιση της κατάστασης.

Δημοτικότητα

Η δημοτικότητα, έχει παρόμοια έννοια και σημασία με το επόμενο κριτήριο που είναι η κοινότητα. Αφορά το σύνολο των ατόμων, κυρίως προγραμματιστών, τα οποία γνωρίζουν απλά την ύπαρξη ή έχουν ασχοληθεί με το κάθε Framework. Για την μέτρηση αυτού του μεγέθους χρησιμοποιήθηκε η βασική πηγή της συγκεκριμένης έρευνας (StateofJs, 2018a). Τα αποτελέσματα παρουσιάζονται στον πίνακα 2. στον οποίο συμπεριλαμβάνονται τα άτομα που απάντησαν πως έχουν χρησιμοποιήσει το κάθε Framework και θα το ξαναχρησιμοποιούσαν μελλοντικά χωρίς πρόβλημα. Συμπληρωματικά παρατίθενται στοιχεία για το σύνολο των npm downloads του κάθε Framework. Με την βοήθεια του εργαλείου npmtrends.com (NPMTrends, 2019) αντιπαρατίθενται τα Downloads του δημοφιλέστερου package για το κάθε Framework. Τα αποτελέσματα συμπεριλαμβάνονται στα εικόνα 2-3.



Εικόνα 2-3: Number of Downloads, Front-End Js Frameworks (npmtrends.com, 2018-2019)

Διαδικτυακά μαθήματα

Το συγκεκριμένο κριτήριο αφορά το σύνολο του αριθμού των διαθέσιμων μαθημάτων, και διαδικτυακών πόρων που στοχεύουν στην εκμάθηση του χειρισμού του κάθε Framework. Αυτό το κριτήριο είναι ιδιαίτερα σημαντικό για κάποιον ο οποίος δεν έχει έρθει σε καμία επαφή με κάποιο από τα συγκεκριμένα Frameworks και θα χρειαστεί διαδικτυακούς πόρους οι οποίοι θα τον βοηθήσουν στην εξοικείωση του με αυτό. Πραγματοποιήθηκε έρευνα σε 2 πολύ μεγάλους σε μέγεθος ιστότοπους που φιλοξενούν διαδικτυακά μαθήματα Udemy (Udemy, 2019) και egghead.io (EggHead, 2019). +Στον πίνακα 2-9 και 2-10 ,παρουσιάζονται τα αποτελέσματα αναζήτησης για το κάθε Framework. Συμπληρωματικά μέσα από διάφορες πηγές του διαδικτύου και χρήση του Google Search επιβεβαιώθηκε η κατάσταση που παρουσιάζεται στο Udemy και Egghead, όσων αφορά το σύνολο των διαθέσιμων μαθημάτων.

Πίνακας 2-9: Σύνολο μαθημάτων στο Udemy για το κάθε Framework.

Front-End Frameworks	Αριθμός μαθημάτων Udemy
React	1732
Vue.js	504
Angular2	2065
Preact	0

Ember	18
Polymer	3
Svelte	1
Aurelia	5
Hyperapp	1
Backbone	10
Mithril	0

Πίνακας 2-10: Σύνολο μαθημάτων στο Egghead.io για το κάθε Framework.

Front-End Frameworks	Αριθμός μαθημάτων Egghead
React	~930
Vue.js	~150
Angular2	~345
Preact	11
Ember	14
Polymer	11
Svelte	15
Aurelia	0
Hyperapp	0
Backbone	0
Mithril	0

Κοινότητα

Η κοινότητα με την σειρά της είναι μια μετρική/κριτήριο που βοηθάει στην αξιολόγηση της αξιοπιστίας και της συνέχισης της εξελικτικής πορείας του κάθε Framework. Όταν αυτό χρησιμοποιείται σε μεγάλο βαθμό από άτομα της προγραμματιστικής κοινότητας, σημαίνει πως κατά πάσα πιθανότητα θα συνεχίσει να αναπτύσσεται και να ενσωματώνει νέες τεχνολογίες αλλά ταυτόχρονα θα υπάρχουν και πολλές πηγές εύρεσης πληροφοριών και επίλυσης τυχών απορίων που μπορεί να προκύψουν στον χρήστη. Για την μέτρηση του μεγέθους της κοινότητας του κάθε Framework χρησιμοποιήθηκαν αποτελέσματα από δύο μεγάλες πλατφόρμες που

σχετίζονται συνολικά με την κοινότητα των προγραμματιστών. Αρχικά μετρήθηκαν τα συνολικά αστέρια (αξιολόγηση) του καθενός στην πλατφόρμα του Github (GitHub, 2019), τα αποτελέσματα παρουσιάζονται στον πίνακα 2-11. Επίσης έγινε έρευνα και των συνολικών ετικετών (tags) του κάθε Framework, στις δημοσιεύσεις της σελίδα του StackOverflow (StackOverflow, 2019), με τα αποτελέσματα να παρουσιάζονται στον πίνακα 2-12.

Πίνακας 2-11: Github Stars of Front-End Js Frameworks (GitHub.com, 2019)

Front-End JavaScript Frameworks	GitHub Stars
React	123k
Vue.js	129k
Angular2	47.9k
Preact	21.6k
Ember	20.7k
Polymer	826
Svelte	9.2k
Aurelia	10.8k
Hyperapp	16.3k
Backbone	27.4k
Mithril	10.7k

Πίνακας 2-12: Number of Tags, Front-End Js Frameworks (StackOverflow.com, 2019)

Front-End JavaScript Frameworks	Number of Tags
React	125040
Vue.js	30465
Angular2	165342
Preact	168
Ember	23056
Polymer	8041
Svelte	87

Aurelia	3022
Hyperapp	7
Backbone	20606
Mithril	201

Υποστήριξη από μεγάλη εταιρεία

Το συγκεκριμένο κριτήριο σύγκρισης βρίσκει εφαρμογή σε πολύ λίγα Frameworks, ωστόσο είναι μια ακόμη επιβεβαίωση πως η ζωή του συγκεκριμένων τα οποία υποστηρίζονται από μεγάλες εταιρίες, δεν θα είναι μόνο λίγοι μήνες. Καθώς πολλά από τα Frameworks που παρουσιάζονται στην εργασία είναι πρόσφατα δημιουργήματα των κατασκευαστών τους, τίθεται το ερώτημα εάν θα αντέξουν στον χρόνο και αν θα μπορέσουν να προσαρμοστούν και να αναβαθμιστούν στα νέα δεδομένα που παρουσιάζονται καθημερινά στον χώρο. Έχοντας της στήριξη μιας μεγάλη εταιρείας η δουλεία στην έρευνα και την ανάπτυξη του κάθε Framework, θα είναι πολύ σημαντικότερη συγκριτικά με ένα μικρό που συντηρείται από λίγα άτομα με περιστασιακές δωρεές. Συνολικά η στήριξη μιας μεγάλης εταιρείας σε ένα Framework, του προσδίδει σταθερότητα, καλή λειτουργικότητα, όντας μια σίγουρη επιλογή η οποία σίγουρα θα διαρκέσει στον χρόνο. Οι δύο εταιρίες κολοσσοί αυτή την στιγμή, Facebook και Google, στηρίζουν τα React και Angular + Polymer αντίστοιχα.

Μερίδιο Αγοράς

Ακόμη ένα κριτήριο στην ίδια λογική με τα δύο προηγούμενα είναι αυτό του μεριδίου αγοράς. Με τον όρο θα συμπεριλάβουμε τα ποσοστά χρήσης του κάθε JavaScript Framework με βάση τις απαντήσεις που δόθηκαν στην έρευνα του 'state of js' και ταυτόχρονα τις διαθέσιμες θέσεις εργασίες που προσφέρονται σε χώρες της Ευρώπης μέσω της πιο γνωστής πλατφόρμας που αφορά τέτοια ζητήματα, LinkedIn (LinkedIn, 2019). Τα αποτελέσματα παρουσιάζονται στους πίνακες 2-13 και 2-14.

Πίνακας 2-13: Ποσοστά χρήσης Front-End Js Frameworks (StateofJs.com)

Front-End JavaScript Frameworks	Used it, would use again %
React	64.8

Vue.js	28.8
Angular2	23.9
Preact	6.2
Ember	5
Polymer	3.1
Svelte	<1
Aurelia	<1
Hyperapp	<1
Backbone	<1
Mithril	<1

Πίνακας 2-14: Διαθέσιμες θέσεις εργασίας LinkedIn

Front-End JavaScript Frameworks	Διαθέσιμες Θέσεις Εργασίας
React	82
Vue.js	51
Angular2	583
Preact	0
Ember	8
Polymer	1
Svelte	0
Aurelia	4
Hyperapp	0
Backbone	3
Mithril	0

Documentation

Το συγκεκριμένο κριτήριο σύγκρισης είναι εξίσου σημαντικό για κάποιον που ξεκινάει την εκμάθηση ενός Framework. Το Documentation του κάθε Framework είναι αυτό που θα δώσει τις περισσότερες φορές απαντήσεις στις απορίες και στα προβλήματα που θα προκύψουν κατά την διάρκεια της ανάπτυξης. Παρέχει σημαντικές λεπτομέρειες για την φύση του Framework και τον τρόπο που αυτό διαχειρίζεται κάποιες γνωστές

καταστάσεις. Η ύπαρξη λοιπόν ενός τέτοιου εμπλουτισμένου οδηγού με όσο το δυνατόν περισσότερες λεπτομέρειες, βοηθάει και ενθαρρύνει τον χρήστη έτσι ώστε να κατανοήσει πλήρως τον τρόπο που τα Framework λειτουργούν και χειρίζονται τα ζητήματα αυτά. Η πλειοψηφία των Front End Frameworks που εξετάζονται, παρέχουν πλήρη και λεπτομερή documentations τα οποία στις περισσότερες περιπτώσεις μπορούν να βοηθήσουν και να καθοδηγήσουν τον χρήστη.

2.3.2 Data Layer Frameworks

Η μικρότερη κατηγορία από τις τρείς, που οι τεχνολογίες της χρησιμοποιούνται συμπληρωματικά και σε συνδυασμό με κάποιο Front End Framework της προηγούμενης κατηγορίας. Οι τεχνολογίες που παρουσιάζονται σε αυτή την κατηγορία είναι αρκετά διαφορετικές μεταξύ τους και έτσι έχει πραγματοποιηθεί ένας διαχωρισμός στα κριτήρια τα οποία θα συγκριθούν. Τα δύο πρώτα Frameworks (Redux, MobX), είναι της ίδιας φιλοσοφίας και επιλύουν παρόμοια προβλήματα με διαφορετικούς τρόπους. Και τα δύο αφορούν το state management κομμάτι των Front End Frameworks, και προορίζονται για συνεργασία με Framework όπως το React ή το Vue τα οποία υλοποιούν λειτουργίες μόνο τους View Layer. Από την άλλη, τα δύο επόμενα Frameworks (Apollo, GraphQL), έχουν δημιουργηθεί και προσφέρουν λύσεις σε διαφορετικά προβλήματα. Λειτουργούν και αυτά σε συνεργασία με κάποια Front End τεχνολογία, αλλά είναι φτιαγμένα για να λειτουργούν μαζί, για την υλοποίηση του GraphQL, την δημιουργία δηλαδή εύκολα διαχωρίσιμων APIs. Το Apollo είναι εκεί για να υλοποιεί ουσιαστικά την σύνδεση των GraphQL τεχνολογιών με το Front End Framework που χρησιμοποιείται. Ωστόσο, με την χρήση του Apollo ο χρήστης θα μπορέσει να απολαύσει τις GraphQL τεχνολογίες που προαναφέρθηκαν και ταυτόχρονα του παρέχεται η δυνατότητα να πραγματοποιήσει την υλοποίηση του state management των Local δεδομένων, με την ίδια λογική που διαχειρίζονται τα δεδομένα που έρχονται από το API στο GraphQL. Άρα η χρήση και μιας ακόμη βιβλιοθήκης για το state management όπως το Redux ή MobX δεν είναι αναγκαία.

Ο πίνακας 2-15 που παρουσιάζεται παρακάτω, περιέχει αρχικά κοινά κριτήρια σύγκρισης που μπορούν να εφαρμοστούν επάνω σε όλα τα Framework, και έπειτα τα

τρία τελευταία κριτήρια αφορούν μόνο πιο τεχνικά χαρακτηριστικά για την αναλυτικότερη σύγκριση ανάμεσα στα, Redux και MobX.

Πίνακας 2-15: Σύγκριση χαρακτηριστικών/κριτηρίων Data-Layer Frameworks (Με κόκκινο χρώμα τα κριτήρια που δεν εφαρμόζονται σε Apollo+GraphQL)

Κριτήρια\Frameworks	**Redux**	**MobX**	**Apollo + GraphQL**
Debugging			
Developer Tools			
Scalability			
BoilerPlate Code			
Documentation			
Εκμάθηση			
Δημοτικότητα			
Κοινότητα			
Store	Μονό	Πολλαπλό	-
Δεδομένα	Plain Object	Observable Object	-
States	Αμετάβλητες	Μεταβλητές	-
Προγραμματιστικό Στυλ	Συναρτησιακό	Αντικειμενοστραφές	-

2.3.2.1 Ανάλυση Κριτηρίων

Debugging

Κατά την ανάπτυξη ενός έργου προγραμματισμού, ο κώδικας ενδέχεται να περιέχει συντακτικά ή λογικά σφάλματα. Πολλά από αυτά τα σφάλματα είναι δύσκολο να διαγνωσθούν με την πρώτη ματιά. Συχνά, όταν ο κώδικας περιέχει σφάλματα, δεν θα συμβεί τίποτα κατά την εκτέλεσή του. Είναι πολύ πιθανό να μην υπάρξουν μηνύματα σφάλματος και να μην υπάρχουν ενδείξεις για την αναζήτηση σφαλμάτων. Η αναζήτηση για (και η επίλυση) σφαλμάτων στον κώδικα προγραμματισμού καλείται σφάλμα κώδικα (W3Schools, 2019b). Η διαδικασία του debugging είναι πολύ σημαντική και τις περισσότερες φορές απαραίτητη καθώς μέχρι να επιτευχθεί το επιθυμητό αποτέλεσμα, ο

κώδικας τροποποιείται συνεχώς με αποτέλεσμα να προκύπτουν και αρκετά σφάλματα κατά την εκτέλεση. Η διαδικασία της αποσφαλμάτωσης στην JavaScript πραγματοποιείται κυρίως μέσω των Browsers που παρέχουν εργαλεία γι αυτό τον σκοπό (W3Schools, 2019b). Όταν ο κώδικας είναι πιο αυστηρός χωρίς να δημιουργούνται abstractions η διαδικασία του debugging γίνεται ευκολότερη, καθώς τα βήματα για να γίνει κάτι είναι συγκεκριμένα. Από την άλλη η ύπαρξη των abstractions και ενός πιο ελεύθερου προγραμματιστικού στυλ, έχει ως επακόλουθο την δυσχέρεια της διαδικασίας (Chandran, 2017).

Developer Tools

Τα DevTools όπως αναφέρονται πολλές φορές είναι το σύνολο των εργαλείων τα οποία παρέχονται στον προγραμματιστή με σκοπό την διευκόλυνση της διαδικασίας ανάπτυξης. Παραδοσιακά με τον όρο αυτό, συνήθως χαρακτηρίζονται εργαλεία όπως Linter, Compiler, Debugger, ωστόσο καθώς η φύση της JavaScript είναι δυναμική δημιουργείται η ανάγκη για χρήση διαφορετικών και πιο εξελιγμένων εργαλείων τα οποία θα λειτουργούν κατά την διάρκεια εκτέλεσης της εφαρμογής (Elliott, 2017). Σε έργα προγραμματισμού μεγάλης κλίμακας, τα οποία πολλές φορές αναπτύσσονται από μεγάλες ομάδες προγραμματισμού, οι χρήση τέτοιων εργαλείων είναι απαραίτητη, καθώς το κέρδος σε χρόνο και επιπλέον εργασία είναι μεγάλο. Όσων αφορά τα Framework που αναλύονται σε αυτή την ενότητα, παρέχουν ένα αξιόλογο σύνολο από εργαλεία για την επίτευξη αυτών των σκοπών. Το Redux βαδίζοντας μπροστά από τα άλλα δύο προσφέρει ένα πλούσιο οικοσύστημα από ισχυρά developer tools τα οποία υπερέχουν του MobX (Chandran, 2017; Educba, 2018). Ταυτόχρονα το Apollo Client και το GraphQL με την σειρά του παρέχει και αυτό εργαλεία τα οποία πετυχαίνουν διαφορετικούς σκοπούς, ωστόσο και αυτά πολύ αξιόλογα και χρήσιμα προς τον χρήστη που θέλει να χρησιμοποιήσει το κάθε Framework (ApolloGraphQL, 2019).

Scalability

Το συγκεκριμένο κριτήριο είναι σημαντικό και αφορά συνολικά και γενικότερα όλα τα έργα προγραμματισμού. Είναι ένα βασικό στοιχείο του επιχειρησιακού λογισμικού. Δίνοντας προτεραιότητα από την αρχή στο scalability, δημιουργείται χαμηλότερο κόστος συντήρησης, καλύτερη εμπειρία χρήστη και μεγαλύτερη ευελιξία.

Γενικότερα ένα σύστημα θεωρείται scalable όταν δεν χρειάζεται να επανασχεδιαστεί για να διατηρήσει αποτελεσματική απόδοση κατά τη διάρκεια ή μετά από μια απότομη αύξηση του φόρτου εργασίας του (conceptanext, 2019). Από την στιγμή που γίνεται λόγος για data layer frameworks, στην συγκεκριμένη ενότητα, το scalability είναι ένας παράγοντας ο οποίος επηρεάζει σε μεγάλο βαθμό την σωστή επιλογή του κατάλληλου framework. Το Redux ξεχωρίζει σε αυτό το χαρακτηριστικό. Είναι η κατάλληλη επιλογή για μια εφαρμογή η οποία θα φτιαχτεί με αυτή την προοπτική του scalability. Τα developer tools που παρέχει και αναφέρθηκαν παραπάνω καθώς και ο αυστηρός τρόπος γραφής κώδικα διευκολύνει την κατάσταση όταν η εφαρμογή μεγαλώνει σε μέγεθος (Chandran, 2017; Educba, 2018; Ravichandran, 2019; Wieruch, 2017). Οι υπηρεσίες του GraphQL επίσης παρέχουν πολλές δυνατότητες για την ανάπτυξη μια πλήρως scalable εφαρμογής.

BoilerPlate Code

Στον προγραμματισμό, ο όρος boilerplate κώδικας ή σκέτο boilerplate αναφέρεται σε τμήματα κώδικα που πρέπει να συμπεριληφθούν σε πολλά μέρη με ελάχιστες ή καθόλου τροποποιήσεις. Συχνά χρησιμοποιείται όταν γίνεται αναφορά σε γλώσσες που θεωρούνται λεπτομερείς, δηλαδή ο προγραμματιστής πρέπει να γράψει πολύ κώδικα για να κάνει ελάχιστες εργασίες (Zaveri, 2018). Ωστόσο σε μια εφαρμογή στην οποία θέλουμε να υπάρχει μικρό μέγεθος με λίγο και αποτελεσματικό κώδικα, δεν είναι θετικό ένα Framework να χρειάζεται boilerplate κώδικα για να πραγματοποιήσει συγκεκριμένες λειτουργίες. Κυρίως σε ένα Framework το οποίο προσπαθεί να πετύχει συγκεκριμένες λειτουργίες όπως αυτό της Redux. Από την άλλη το MobX παρέχει σημαντικές ελευθερίες στην συγγραφή του κώδικα, την εγκατάσταση και το σετάρισμα (Hamedani, 2019).

Documentation

Όπως αναλύθηκε παραπάνω στην ενότητα 2.3.1.1

Τα Framework που συγκρίνονται στην συγκεκριμένη εργασία παρέχουν όλα εξαιρετικό documentation το οποίο μπορεί να λειτουργήσει πλήρως αποτελεσματικά για έναν νέο χρήστη ο οποίος θέλει να μάθει τις νέες αυτές τεχνολογίες.

Εκμάθηση

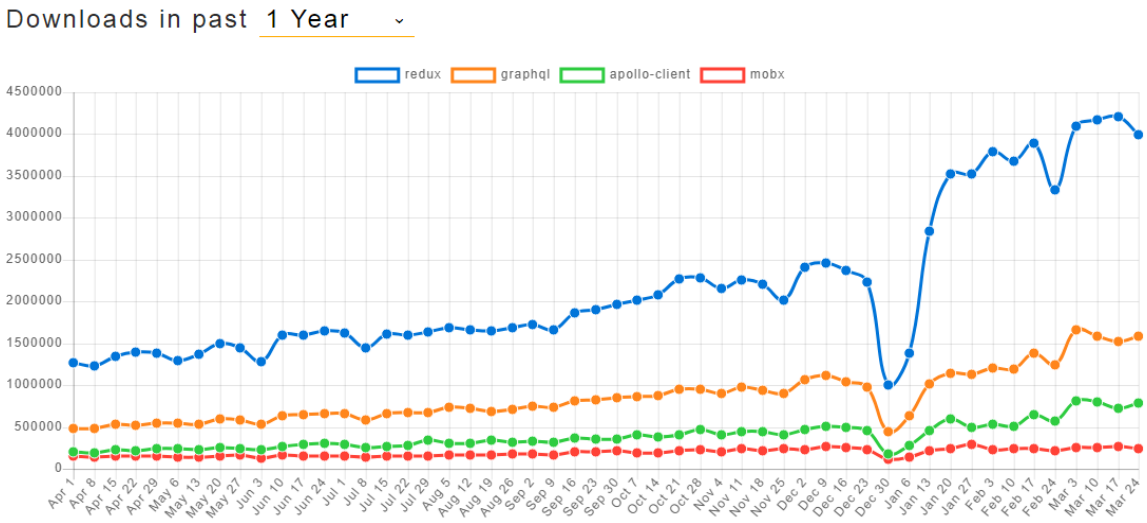
Όπως αναλύθηκε παραπάνω στην ενότητα 2.3.1.1

Το Redux και το GraphQL υστερούν σε αυτόν τον τομέα έναντι του MobX (Hamedani, 2019).

Δημοτικότητα

Όπως αναλύθηκε παραπάνω στην ενότητα 2.3.1.1

Στην Εικόνα 2-4 εμφανίζονται τα αποτελέσματα από την σελίδα του npmtrends.com (NPMTrends, 2019), μετρώντας τον συνολικό αριθμό των npm packages του κάθε Framework.



Εικόνα 2-4: Number of Downloads, Data-Layer Js Frameworks (NPMTrends, 2019)

Κοινότητα

Όπως αναλύθηκε παραπάνω στην ενότητα 2.3.1.1

Στον πίνακα 2-16 εμφανίζεται ο συνολικός αριθμός των GitHub Stars από την διαδικτυακή πλατφόρμα του GitHub.com (GitHub, 2019), που έχει το κάθε Framework, και στην συνέχεια στον πίνακα 2-17 παρουσιάζεται ο αριθμός των tags σε πόστ με την ονομασία κάποιου Framework από την σελίδα του stackoverflow.com (StackOverflow, 2019).

Πίνακας 2-16: GitHub Stars of Data-Layer Js Frameworks (GitHub.com, 2019)

Data-Layer JavaScript Frameworks	GitHub Stars
Redux	47.1k
GraphQL	11.1k
Apollo	10.2k
MobX	18.5k

Πίνακας 2-17: Number of Tags, Data-Layer Js Frameworks (StackOverflow.com, 2019)

Data-Layer JavaScript Frameworks	Number of Tags
Redux	17902
GraphQL	5515
Apollo	1552
MobX	903

Store

Το store είναι όπου διατηρούνται και αποθηκεύονται τα τοπικά δεδομένα. Περιέχει εξ ολοκλήρου τα states της εφαρμογής τα οποία διατηρεί σε ένα μεγάλο αντικείμενο json. Στο Redux, υπάρχει μόνο ένα store το οποία κρατάει την κατάσταση της εφαρμογής κατά την διάρκεια εκτέλεσής της. Η κατάσταση στο store είναι αμετάβλητη. Αυτό διευκολύνει τον χρήστη να γνωρίζει από πού να βρει τα δεδομένα / κατάσταση κάθε στιγμή. Από την άλλη η Mobx, επιτρέπει πολλαπλά stores. Τα καταστήματα αυτά μπορούνε να διαχωριστούνε με βάση την λογική της εφαρμογής, έτσι ώστε όλη η κατάσταση της εφαρμογής να μην είναι αποθηκευμένη σε ένα κατάστημα (Hamedani, 2019).

Δεδομένα

Το Redux χρησιμοποιεί απλά αντικείμενα JavaScript ως δομές δεδομένων για την αποθήκευση της κατάστασης. Κατά τη χρήση του Redux, οι ενημερώσεις των δεδομένων πρέπει να παρακολουθούνται χειροκίνητα. Το Mobx χρησιμοποιεί observable δεδομένα. Αυτό βοηθά στην αυτόματη παρακολούθηση των αλλαγών μέσω σιωπηρών συνδρομών.

Στο Mobx, οι ενημερώσεις παρακολουθούνται αυτόματα. Ως εκ τούτου, όλη η διαδικασία καθίσταται ευκολότερη για τον προγραμματιστή (Hamedani, 2019).

States

Το Redux χρησιμοποιεί αμετάβλητες καταστάσεις. Αυτό σημαίνει ότι οι καταστάσεις μπορούν μόνο να αναγνωσθούν χωρίς να υπάρχει η δυνατότητα της άμεσης αντικατάστασής τους από τον χρήστη. Στο Redux η προηγούμενη κατάσταση αντικαθίσταται από μια νέα κατάσταση. Ως αποτέλεσμα, το Redux είναι pure καθώς χρησιμοποιεί pure functions (Functions τα οποία δίνοντας του την ίδια είσοδο πάντα θα επιστρέφουν την ίδια έξοδο, χωρίς την παραγωγή side effects (Elliott, 2019)). Αυτό μπορεί να αποδειχτεί πολύ χρήσιμο όταν δημιουργηθεί η ανάγκη για επιστροφή σε μια προηγούμενη κατάσταση. Π.χ. - Μια ενέργεια αναίρεσης. Στην περίπτωση της MobX οι καταστάσεις μπορούν να αντικατασταθούν άμεσα. Μια κατάσταση μπορεί να ενημερωθεί άμεσα από τον χρήστη με τις νέες τιμές. Ως αποτέλεσμα, το MobX μπορεί να χαρακτηριστεί ως impure.

Προγραμματιστικό Στυλ

Δεδομένου ότι οι περισσότεροι από τους παραδοσιακούς προγραμματιστές Javascript είναι εξοικειωμένοι με τον αντικειμενοστραφή προγραμματισμό είναι ευκολότερο να εξοικειωθούν με το MobX καθώς χρησιμοποιεί ένα τέτοιο προγραμματιστικό στυλ. Υπάρχουν πολλά abstractions στο MobX που το κάνουν επίσης ευκολότερο. Βασικότερα χαρακτηριστικά του αντικειμενοστραφούς στυλ του είναι, η διατήρηση των states μέσα σε observable items όπως ειπώθηκε και παραπάνω. Το Redux ακολουθεί το παράδειγμα του συναρτησιακού προγραμματισμού. Για τους προγραμματιστές Javascript χωρίς εμπειρία στον συναρτησιακό προγραμματισμό, μπορεί να είναι δύσκολο να γίνουν οι αντιληπτές από τον χρήστη οι τεχνικές που χρησιμοποιεί.

2.3.3 Back-End Frameworks

Ο χώρος των Back End JavaScript Framework, αυτά που λειτουργούν επάνω στο node.js με λίγα λόγια, είναι πολύ μεγάλος. Καθώς οι back end λειτουργίες και έργα που μπορούν να αναπτυχθούν είναι πολλές, τα διάφορα Framework που υπάρχουν

υλοποιημένα σε node.js λύνουν τις περισσότερες φορές διαφορετικά και πολύ συγκεκριμένα προβλήματα. Αρχικά ο προγραμματιστής χρησιμοποιεί το node.js για να λύσει με ευκολία κάποια θέματα. Η ευκολία, σε αυτή την περίπτωση, έχει μια εξαιρετικά μεταβλητή έννοια που εξαρτάται από το μήκος του έργου, την πολυπλοκότητα, την επεκτασιμότητα (scalability) και τελικά τις τεχνικές ανάπτυξης εφαρμογών. Έτσι, τα Back End Frameworks του Node.js έρχονται σε διάφορες υλοποιήσεις και χωρίζονται σε 4 κύριες ομάδες (altexsoft.com, 2017):

- MVC Framework: Αυτά τα πλαίσια παρέχουν μεγάλη ελευθερία ανάπτυξης και ευελιξία χωρίς πρόσθετα και βασίζονται στο σχέδιο σχεδίασης Model-View-Controller.
- Full-Stack MVC Framework. Αυτά τα εργαλεία συνοδεύονται από βιβλιοθήκες, ενσωματωμένα συστήματα, template engines και άλλα πρόσθετα που επιτρέπουν στον προγραμματιστή να αυτοματοποιεί την δουλεία του, αλλά συνήθως με κόστος την ευελιξία.
- REST API Framework. Αυτά τα εργαλεία εξομαλύνουν την ανάπτυξη του διακομιστή REST API και μπορούν να χρησιμοποιηθούν σε συνδυασμό με οποιαδήποτε άλλη ομάδα πλαισίων.
- Άλλα: Υπάρχουν πολλά middleware, βιβλιοθήκες και άλλες συγκεκριμένες λύσεις που μπορούν να θεωρηθούν Framework σε κάποιο βαθμό.

Στον πίνακα 2-18 παρουσιάζεται ο πίνακας σύγκρισης των Frameworks για τα διάφορα κριτήρια.

Πίνακας 2-18: Σύγκριση χαρακτηριστικών Back-End Frameworks

Κριτήρια\Frameworks	Express	Next.Js	Koa	Meteor	Sails	FeathersJS
Routing	x	x	-	-	x	x
User authentication	-	-	-	x	-	x
Template Engine	x	-	-	x	x	x
REST API	x	-	x	-	x	x
object-rational mapping (ORM)	-	-	-	-	x	x
WebSockets	-	-	-	-	x	x
Scalability						
Εκμάθηση						
Δημοτικότητα						
Κοινότητα						
Documentation						

2.3.3.1 Ανάλυση Κριτηρίων

Routing

Όπως αναλύθηκε στην ενότητα 2.3.1.1

User authentication

Ο έλεγχος ταυτότητας χρήστη (User authentication) είναι ένα σημαντικό ζήτημα κατά τη δημιουργία μιας δυναμικής διαδικτυακής εφαρμογής. Προορίζεται για τον εντοπισμό χρηστών και την παροχή διαφορετικών δικαιωμάτων πρόσβασης και περιεχομένου ανάλογα με το αναγνωριστικό του καθ' ενός. Στις περισσότερες περιπτώσεις, η εφαρμογή παρέχει μια φόρμα σύνδεσης με ορισμένα διαπιστευτήρια για την επαλήθευση ενός χρήστη (Deutsch, 2019). Δεδομένου ότι στις μέρες μας η ασφάλεια των προσωπικών πληροφοριών του χρήστη είναι ένα από τα σημαντικότερα ζητήματα, ένα Framework πρέπει να μπορεί και να παρέχει ενσωματωμένες λύσεις στον χρήστη, που αφορούν το συγκεκριμένο πρόβλημα.

Template Engine

Όπως αναλύθηκε στην ενότητα 2.3.1.1

REST API

Το REST είναι συντομογραφία του REpresentational State Transfer. Το REST είναι αρχιτεκτονική που βασίζεται σε πρότυπα ιστού και χρησιμοποιεί πρωτόκολλο HTTP. Έχει να κάνει με resources, όπου κάθε στοιχείο είναι ένα resource και το resource είναι προσβάσιμο από μια κοινή διεπαφή χρησιμοποιώντας τις πρότυπες μεθόδους HTTP. Το REST εισήχθη για πρώτη φορά από τον Roy Fielding το 2000. Ένας REST Server παρέχει απλώς πρόσβαση στα resources του μαζί με την δυνατότητα της τροποποίησης τους χρησιμοποιώντας πρωτόκολλο HTTP. Κάθε resource προσδιορίζεται από URIs / global IDs. Το REST χρησιμοποιεί διάφορες αναπαραστάσεις για να απεικονίσει ένα resource όπως: απλό κείμενο, JSON ή XML, ωστόσο το JSON είναι το πιο δημοφιλές (tutorialspoint.com, 2019c). Οι βασικές μέθοδοι HTTP που χρησιμοποιούνται συνήθως στην αρχιτεκτονική που βασίζεται σε REST είναι:

- GET - Χρησιμοποιείται για την παροχή πρόσβασης μόνο για ανάγνωση σε ένα resource.
- PUT - Χρησιμοποιείται για τη δημιουργία ενός νέου resource.
- DELETE - Χρησιμοποιείται για την κατάργηση ενός resource.
- POST - Χρησιμοποιείται για την ενημέρωση ή τη δημιουργία ενός νέου resource.

Object-Rational Mapping (ORM)

Οι λύσεις ORM είναι χρήσιμες για τη διευκόλυνση της ανάπτυξης API δεδομένων. Στην προγραμματιστική κοινότητα, αυτή η αρχιτεκτονική δεδομένων υλοποιείται συνήθως και ελέγχεται χρησιμοποιώντας scripts των βάσεων δεδομένων, όπως αυτά της SQL. Η διαδικασίες εξόρυξης, μετασχηματισμού και φόρτωσης των δεδομένων με κώδικα, επιτρέπει σε ένα σύστημα να ενσωματώνει ευκολότερα τα δεδομένα αυτά από διαφορετικές πηγές. Οι βάσεις δεδομένων SQL με τις διάφορες εκδόσεις τους, δεδομένα NoSQL, δεδομένα συστήματος αρχείων και δεδομένα τρίτων μπορούν να ενσωματωθούν σε μια μόνο γλώσσα και να διαχειριστούν με τον όλα με τον ίδιο τρόπο, χρησιμοποιώντας το JavaScript ORM. Τέλος, ο έλεγχος δεδομένων μέσα από τον κώδικα

επιτρέπει επίσης σε ένα σύστημα να χρησιμοποιήσει τα δεδομένα κατά το χρόνο εκτέλεσης ή στη διαδικασία δημιουργίας. Συμπερασματικά, τα ORM βελτιώνουν την παραγωγικότητα των προγραμματιστών παρέχοντας ένα API υψηλού επιπέδου, σε μία γλώσσα, με λειτουργικότητα που παραδοσιακά θα απαιτούσε πολλά διαφορετικά εργαλεία και σύνολα δεξιοτήτων για να επιτευχθεί. Οι λιγότερες απαιτήσεις σε, δεξιότητες, ανάγκες εργαλείων και απαιτούμενες ώρες, διευκολύνουν την ανάπτυξη του έργου (Vandivier, 2018).

WebSockets

Τα WebSockets επιτρέπουν στο διακομιστή και τον πελάτη να στέλνουν μηνύματα μεταξύ τους ανά πάσα στιγμή, μετά από τη δημιουργία μιας σύνδεσης, χωρίς ρητό αίτημα του ενός ή του άλλου. Αυτό έρχεται σε αντίθεση με το HTTP πρωτόκολλο, το οποίο συνηθίζεται να συνδέεται με την αρχή της πρόκλησης-απόκρισης - από το οποίο πρέπει να ζητηθούν ρητά τα δεδομένα. Με περισσότερους τεχνικούς όρους, τα WebSockets επιτρέπουν μια πλήρη αμφίδρομη σύνδεση μεταξύ του πελάτη και του διακομιστή. Σε ένα σύστημα πρόκλησης-απόκρισης δεν υπάρχει κανένας τρόπος για τους πελάτες να γνωρίζουν πότε είναι διαθέσιμα νέα δεδομένα γι 'αυτά (εκτός από το να ζητούν από τον διακομιστή περιοδικά - παραπομπή ή μακρόχρονη δημοσκόπηση), με το Websockets ο διακομιστής μπορεί να προωθήσει νέα δεδομένα ανά πάσα στιγμή, η καλύτερη λύση για εφαρμογές "σε πραγματικό χρόνο". Είναι σημαντικό να σημειωθεί ότι το WebSockets μετατρέπει τη σύνδεσή τους HTTP σε μια σύνδεση WebSocket. Με άλλα λόγια, μια σύνδεση WebSocket χρησιμοποιεί μόνο HTTP για την αρχική σύζευξη (για εξουσιοδότηση και επαλήθευση), μετά την οποία χρησιμοποιείται η σύνδεση TCP για την αποστολή δεδομένων μέσω του δικού της πρωτόκολλου (hybd) (Rusendić, 2018).

Scalability

Όπως αναλύθηκε στην ενότητα 2.3.2.1

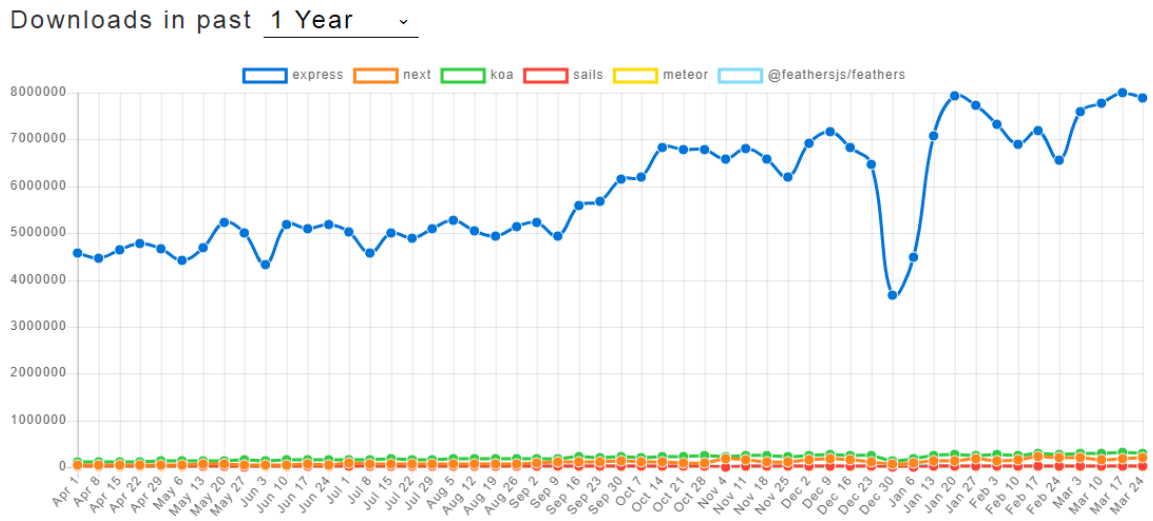
Εκμάθηση

Όπως αναλύθηκε στην ενότητα 2.3.1.1

Δημοτικότητα

Όπως αναλύθηκε στην ενότητα 2.3.1.1

Στο διάγραμμα 2-5 παρουσιάζεται το σύνολο από downloads του npm των Frameworks για την περίοδο 2018-2019, με το express να βρίσκεται στην κορυφή με μεγάλη διαφορά (NPMTrends, 2019).



Εικόνα 2-5: Number of Downloads, Back-End Js Frameworks (npmtrends.com, 2018-2019)

Κοινότητα

Όπως αναλύθηκε στην ενότητα 2.3.1.1

Στον πίνακα 2-19 εμφανίζεται ο συνολικός αριθμός των GitHub Stars από την διαδικτυακή πλατφόρμα του GitHub.com (GitHub, 2019), που έχει το κάθε Framework, και στην συνέχεια στον πίνακα 2-20 παρουσιάζεται ο αριθμός των tags σε πόστ με την ονομασία κάποιου Framework από την σελίδα του stackoverflow.com (StackOverflow, 2019).

Πίνακας 2-19: GitHub Stars of Back-End Js Frameworks (GitHub.com, 2019)

Back-End JavaScript Frameworks	GitHub Stars
Express	42.6k
Next.js	35.1k
Koa	25.1k

Meteor	40.8k
Sails	20.3k
FeathersJs	52

Πίνακας 2-20: Number of Tags, Back-End Js Frameworks (StackOverflow.com, 2019)

Back-End JavaScript Frameworks	Number of Tags
Express	50120
Next.js	704
Koa	858
Meteor	28099
Sails	6178
FeathersJs	539

Documentation

Όπως αναλύθηκε στην ενότητα 2.3.1.1

3 Μεθοδολογία

3.1 Εισαγωγή

Με βάση την παραπάνω ανάλυση των Framework, γίνεται κατανοητό πως η επιλογή του κατάλληλου προς χρήση σε κάθε περίπτωση, δεν είναι μια εύκολη απόφαση. Συνολικά και μέσα από τις αναλύσεις προκύπτει πως τα Frameworks έχουν κοινά αλλά και πολλά ξεχωριστά χαρακτηριστικά. Το γεγονός αυτό κάνει κατάλληλη την χρήση του καθενός σε συγκεκριμένες περιπτώσεις και καταστάσεις. Αρχικά η φύση του προγραμματιστικού έργου που επιθυμεί ο χρήστης να αναπτύξει είναι ένας βασικός παράγοντας που διαμορφώνει αυτή την απόφαση επιλογής. Ταυτόχρονα σημαντικοί είναι και παράγοντες όπως Performance, Κοινότητα, Δημοφιλία, Μερίδιο αγοράς κ.α., οι οποίοι τελικά και αυτοί παίζουν σημαντικότατο ρόλο και επηρεάζουν άμεσα και έμμεσα την επιλογή του κατάλληλου Framework. Στόχος στην συγκεκριμένη ενότητα της έρευνας είναι να, επισημανθούν κάποιες βέλτιστες περιπτώσεις χρήσης του κάθε Framework με την δημιουργία υποθετικών σεναρίων χρήσης. Έπειτα να δημιουργηθούν ερωτήματα προς τον χρήστη εξαγόμενα από την επιλογή των κριτηρίων που έγινε στο δεύτερο κεφάλαιο. Τέλος με την ύπαρξη των εν λόγω πληροφοριών να δημιουργηθεί μια πλήρης δενδρική δομή στην οποία θα παρουσιάζονται λεπτομερώς οι αποφάσεις και οι διαδρομές για την τελική επιλογή του κατάλληλου Framework ανά περίπτωση.

3.2 Βέλτιστες περιπτώσεις χρήσης

Σε αυτό το σημείο θα παρουσιαστούν πίνακες για την κάθε κατηγορία Framework, οι οποίοι θα εμπεριέχουν από ένα έως τρία σενάρια χρήσης για το κάθε ένα. Τα σενάρια αυτά θα είναι τα βέλτιστα σενάρια χρήσης του καθενός. Πιο συγκεκριμένα, οι αναλύσεις των προηγούμενων κεφαλαίων και τα κριτήρια που παρουσιάστηκαν θα βοηθήσουν για να βρεθούν τα σενάρια αυτά. Επίσης, τα συγκεκριμένα σενάρια θα αφορούν επιχειρήσεις μικρές ή μεγάλες καθώς και προγραμματιστές που δουλεύουν μόνοι τους. Η κάθε εταιρεία ή προγραμματιστής ανάλογα με τις απαιτήσεις της/του και τις προϋποθέσεις που έχουν θέσει οι ίδιοι ή οι πελάτες τους, μπορούν να χρησιμοποιήσουν το κατάλληλο Framework σε κάθε περίπτωση.

3.2.1 Front End JavaScript Frameworks

React	1. Μια μεσαίου/μεγάλου μεγέθους εταιρεία που θέλει να φτιάξει ένα δυνατό και σταθερό SPA στο οποίο οι αλλαγές των δεδομένων επιφέρουν αυτόματες αλλαγές και στο view layer.
	2. Μια εταιρεία η οποία θέλει να φτιάξει μια εφαρμογή η οποία πρέπει να διανεμηθεί και σε άλλες πλατφόρμες εκτός των Web Browser.
	3. Μια εταιρεία η οποία ψάχνει να προσλάβει προγραμματιστές για την δημιουργία ενός νέου project.
Vue	1. Μια Startup έχοντας προγραμματιστές που γνωρίζουν βασικές έννοιες της JavaScript χωρίς να έχουν μεγάλη εμπειρία.
	2. Μια Startup ή και μεγαλύτερη εταιρεία η οποία θέλει να δημιουργήσει ένα έργο λογισμικού σε μικρό χρονικό διάστημα.
	3. Μια εταιρεία η οποία έχει ήδη ένα υπάρχων project και θέλει να ενσωματώσει ένα Framework σε αυτό.
Angular	1. Μια μεγάλη εταιρεία η οποία λειτουργεί με μεγάλες ομάδες προγραμματιστών και θέλει να δημιουργήσει μια enterprise εφαρμογή.
	2. Μια μεγάλη εταιρεία που θέλει να δημιουργήσει ένα μεγάλης κλίμακας έργο και όλες του οι λειτουργίες να εξαρτώνται από ένα και μόνο Framework (όπως η

	Angular) έτσι ώστε να υπάρχει η μέγιστη αξιοπιστία και σταθερότητα.
	3. Μια εταιρεία η οποία δραστηριοποιείται στην δημιουργία εφαρμογών αντικειμενοστραφούς προγραμματισμού και θέλει να δημιουργήσει μια μεγάλου μεγέθους δικτυακή εφαρμογή καθώς η χρήση της TypeScript εμπεριέχει την αντικειμενοστρέφεια.
Preact	1. Μια εταιρεία η οποία έχει αναπτύξει μια εφαρμογή με το Framework της React, ωστόσο οι επιδόσεις της εφαρμογής δεν είναι οι επιθυμητές και θέλει να βελτιωθούν.
	2. Μια εταιρεία η οποία προγραμματίζει κατά βάση με την React και θέλει να δοκιμάσει μια εναλλακτική στα ίδια μέτρα, που απαιτεί λιγότερους πόρους για την εκτέλεση και αποθήκευση και έχει καλύτερες επιδόσεις.
Ember	1. Μια Startup η οποία δεν έχει εμπειρία με κάποιο συγκεκριμένο Framework και θέλει να δημιουργήσει ένα ολοκληρωμένο έργο λογισμικού με αποδοτικό και σωστά γραμμένο κώδικα. (Η Ember 'αναγκάζει' τον προγραμματιστή να χρησιμοποιήσει τις βέλτιστες πρακτικές ανάπτυξης.)
	2. Μια εταιρία η οποία θέλει να επιλέξει ένα Framework ώστε να μπορεί να δημιουργεί ολοκληρωμένα έργα λογισμικού γρήγορα καθώς συνήθως της δίνονται μικρά deadlines από τους πελάτες

	της. (Παρόλο που η Ember η πιο δύσκολη στην εκμάθησή της συγκριτικά με όλα τα Frameworks, από την στιγμή που ο προγραμματιστής εξοικειωθεί με αυτήν ο χρόνος ανάπτυξης ενός έργου ακόμη και μεγάλης κλίμακας είναι πολύ μικρότερος από τα Frameworks με όμοια χαρακτηριστικά όπως η Angular.)
Polymer	1. Μια εταιρία η οποία θέλει να ξεκινήσει να εκμεταλλεύεται τις νέες τεχνολογίες των Browsers όπως τα Web Components.
	2. Μια εταιρία που θέλει να δημιουργήσει custom elements, να εκμεταλλευτεί τις τεχνολογίες των Web Components με την χρήση του Polymer και να τα χρησιμοποιήσει υλοποιώντας μια ολοκληρωμένη εφαρμογή με κάποιο διαφορετικό Framework όπως η Angular.
Svelte	1. Μια Startup έχοντας προγραμματιστές που γνωρίζουν βασικές έννοιες της JavaScript χωρίς να έχουν μεγάλη εμπειρία.
	2. Μια εταιρεία η οποία έχει ήδη ένα υπάρχων project και θέλει να ενσωματώσει ένα Framework σε αυτό.
	3. Μια εταιρεία η οποία θέλει για οποιοδήποτε λόγο να έχει την μεγαλύτερη δυνατή ταχύτητα εκτέλεσης στην εφαρμογή της, συγκριτικά με τα υπόλοιπα Frameworks.
Aurelia	1. Μια εταιρία η οποία αναπτύσσει εφαρμογές με την Angular 1, αλλά θέλει να

	εξελιχθεί καθώς η Angular 1 είναι πλέον outdated χωρίς να λαμβάνει νέες ενημερώσεις και χωρίς να ακολουθεί τις τεχνολογικές εξελίξεις.
	2. Μια μεγάλη εταιρεία που θέλει να δημιουργήσει ένα μεγάλης κλίμακας έργο χρησιμοποιώντας μια εναλλακτική επιλογή αντί για την Angular.
Hyperapp	1. Μια εταιρία ή ένας developer που χρησιμοποιεί την React και θέλει να δοκιμάσει κάτι καινούριο στην ίδια λογική, με πολύ μικρότερο μέγεθος, καλύτερες επιδόσεις και ενσωματωμένη την λειτουργία του state management.
	2. Για έναν προγραμματιστή ο οποίος έχει γνώσεις, έχει ασχοληθεί με τον συναρτησιακό προγραμματισμό και θέλει με την ίδια λογική να αναπτύξει εφαρμογές ιστού, καθώς το Hyperapp έχει πολλά κοινά στοιχεία με αυτόν.
Backbone	1. Όσον αφορά το Backbone, η χρήση του στην αγορά περιορίζεται μόνο στις εταιρίες που την χρησιμοποιούσαν μερικά χρόνια πριν, ως μια ισχυρή και νέα τεχνολογία. Με την άνοδο και την ανάπτυξη της React όμως, οι περισσότεροι μεταφέρθηκαν σε αυτήν. Έτσι συγκριτικά με τα υπόλοιπα Framework το Backbone είναι πιο αδύναμο υποστηρίζοντας παλιότερες τεχνολογίες, και δεν βρίσκει χρήση στην εποχή συγκρίνοντας το με τα υπόλοιπα Frameworks και τεχνολογίες.

Mithril	1. Μια εταιρία ή developer που θέλει να δημιουργήσει μια δικτυακή εφαρμογή η οποία έχει γρήγορες επιδόσεις, μικρό μέγεθος, απλή δομή χωρίς να μπαίνουν όρια στον τρόπο υλοποίησης συγκεκριμένων διαδικασιών. Σημαντική είναι επίσης η χρήση της vanilla javascript καθ όλη την διάρκεια της ανάπτυξης.

3.2.2 Data Layer JavaScript Frameworks

Redux	1. Μια μεσαία/μεγάλη εταιρία η οποία αναπτύσσει εφαρμογές σε React/Vue, των οποίων το μέγεθος και η πολυπλοκότητά αυξάνεται συνεχώς.
Mobx	1. Μια μικρή επιχείρηση ή προγραμματιστή ο οποίος θέλει να χρησιμοποιήσει ένα απλό, εύχρηστο και γρήγορο εργαλείο σε συνδυασμό με το View Framework (React, Vue, κτλ) που χρησιμοποιεί, για την διαχείριση των δεδομένων της εφαρμογής του.
Apollo+GraphQL	1. Μια εταιρία που αναπτύσσει εφαρμογές σε React, χρησιμοποιεί REST API για την επικοινωνία με την βάση δεδομένων της ή με κάποια τρίτη βάση, και θέλει έναν πιο αποτελεσματικό και έξυπνο τρόπο διαχείρισης των δεδομένων.

3.2.3 Back End JavaScript Frameworks

Express	1. Μια μεσαία/μεγάλη εταιρία που

	αναπτύσσει σε React/Vue και θέλει εύκολα να προσθέσει έξτρα λειτουργικότητες σε μια πολύπλοκη εφαρμογή όπως Routing, HTML templates, διαφορετικές εξόδους για τα δεδομένα κτλ.
Next.js	1. Μια εταιρία που αναπτύσσει εφαρμογές σε React και θέλει να δημιουργήσει μια, της οποίας πρωταρχικός στόχος θα είναι η γρήγορη 1η φόρτωση της σελίδας και το καλύτερο δυνατό SEO.
	2. Ένας προγραμματιστής που αναπτύσσει εφαρμογές σε React και θέλει να δοκιμάσει τα πλεονεκτήματα του Server Side Rendering
Koa	1. Έναν προγραμματιστή ή μικρή εταιρία που θέλει να δημιουργήσει ένα μικρής κλίμακας έργο και δεν χρειάζεται όλα τα modules και bundles που υπάρχουν ενσωματωμένα στο Express. Το Koa παρέχει ένα ιδιαίτερο modularity δίνοντας την επιλογή στον χρήστη να επιλέξει μόνο τα modules που θέλει να ενσωματώσει στο project.
Meteor	1. Μια οποιαδήποτε εταιρία ή προγραμματιστής που θέλει να χρησιμοποιήσει μόνο ένα εργαλείο Full Stack για την δημιουργία μια πλήρους εφαρμογής (Front End, Back End, Database κτλ).
	2. Μια οποιαδήποτε εταιρία η προγραμματιστής που θέλει να δημιουργήσει ένα Full Stack

	ολοκληρωμένο έργο σε πολύ μικρό χρονικό διάστημα.
Sails	1. Μια εταιρία ή προγραμματιστής που είναι εξοικειωμένος με τεχνολογίες όπως η Ruby on Rails αλλά θέλει να δοκιμάσει να προγραμματίσει στην ίδια λογικά με JavaScript.
	2. Μια εταιρία η προγραμματιστής που θέλει ένα Full Stack εργαλείο για την ανάπτυξη μιας πλήρους εφαρμογής με διαφορετικά features σε μικρό χρονικό διάστημα.
Feathers.js	1. Μια εταιρία ή προγραμματιστής που προγραμματίζει σε με οποιοδήποτε Front End Framework και μαζί με αυτό, θέλει ένα δυνατό εργαλείο με πολυάριθμες επιλογές για το Back End των εφαρμογών.
	2. Μια εταιρία ή προγραμματιστής που τα Projects που δημιουργεί απαιτούν την χρήση διαφορετικών ειδών βάσεων δεδομένων κάθε φορά.

3.3 Δημιουργία και κατηγοριοποίηση ερωτημάτων προς τους χρήστες

Τα ερωτήματα που θα συνταχθούν και θα αποτελούν τους κόμβους του δέντρου αποφάσεων είναι πολύ σημαντικά για την σωστή λήψη απόφασης και έγκυρης τελικής πρότασης στον χρήστη. Με βάση τα κριτήρια σύγκρισης του κεφαλαίου 2 αλλά και τις βέλτιστες περιπτώσεις χρήσης που παρουσιάστηκαν για το κάθε ένα Framework στην προηγούμενη ενότητα, θα δημιουργηθούν και θα κατηγοριοποιηθούν τα ερωτήματα με στόχο την ορθή δημιουργία του δένδρου αποφάσεων, στο τέλος του κεφαλαίου. Συνολικά διακρίνονται 4 βασικές κατηγορίες στις οποίες μπορούν να κατανεμηθούν όλα τα ερωτήματα που προκύπτουν από τις παραπάνω αναλύσεις. Αρχικά η κατηγορία των

'Γενικών Ερωτημάτων' είναι μια κατηγορία που περιέχει ερωτήματα για το Background και την φύση του προγραμματιστή/εταιρείας που θέλει να αναπτύξει μια εφαρμογή. Έπειτα ακολουθεί η κατηγορία 'Ερωτήματα για τους στόχους και την φύση του Project' στην οποία υπάρχουν ερωτήσεις για το στυλ του Project προς υλοποίηση, το μέγεθος του, κάποιες βασικές τεχνολογίες που πρέπει να χρησιμοποιηθούν αλλά και τον σκοπό και τους στόχους που έχει αυτό. Τρίτη έρχεται η κατηγορία με όνομα 'Γενικά ερωτήματα για συγκεκριμένα Frameworks και τα χαρακτηριστικά τους' η οποία έχει ως στόχο να ελέγξει εάν ο χρήστης χρησιμοποιεί ήδη συγκεκριμένε τεχνολογίες, καθώς και να ρωτήσει τον χρήστη για την σημαντικότατα που ο ίδιος θεωρεί πως έχουν συγκεκριμένα χαρακτηριστικά αυτών. Τέλος παρουσιάζονται κάποια καθαρά 'Τεχνικά Ερωτήματα' τα οποία θα βοηθήσουν για την ορθότερη τελική απόφαση και πρόταση τεχνολογιών στον χρήστη. Όσων αφορά την ιεράρχηση τους, όλα τα ερωτήματα έχουν την ίδια σημαντικότητα. Κανένα ερώτημα δεν μπορεί να θεωρηθεί περισσότερο σημαντικό από τα υπόλοιπα εξ αρχής καθώς ο κάθε χρήστης έχει διαφορετικές απαιτήσεις και στην περίπτωση που έχει κάθε φορά, θεωρεί διαφορετικά πράγματα σημαντικά. Συμπερασματικά, δεν υπάρχουν κόμβοι οι οποίοι πρέπει να είναι ψηλότερα στο δέντρο και άλλοι που πρέπει να είναι χαμηλότερα.

3.3.1 Γενικά Ερωτήματα

1. Είστε σε άμεση αναζήτηση προγραμματιστών (σε περίπτωση εταιρείας);
2. Θέλετε να ξεκινήσετε άμεσα με την ανάπτυξη της εφαρμογής;
3. Έχετε μικρά deadlines στον χρόνο παράδοσης των Project;
4. Είστε εταιρεία και δουλεύετε σε μεγάλες ομάδες προγραμματιστών;
5. Είστε, εσείς ή οι προγραμματιστές σας, εξοικειωμένοι πλήρως με τις έννοιες και την χρήση της JavaScript;
6. Έχετε ήδη αναπτύξει μια εφαρμογή και θέλετε να ενσωματώσετε κάποιο Framework σε αυτή;
7. Θέλετε να χρησιμοποιήσετε μόνο vanilla JavaScript για την ανάπτυξη της εφαρμογής;

8. Επιθυμείτε την χρήση ενός Full Stack εργαλείου για την ανάπτυξη της εφαρμογής σας;
9. Επιθυμείτε να μάθετε ένα Framework σε μικρό χρονικό διάστημα (Εκμάθηση);
10. Θεωρείτε σημαντικό παράγοντα επιλογής την δημοτικότητα των Frameworks;
11. Θεωρείτε σημαντικό παράγοντα επιλογής το σύνολο των διαθέσιμων διαδικτυακών μαθημάτων στο διαδίκτυο για το κάθε Framework;
12. Θεωρείτε σημαντικό παράγοντα επιλογής το μέγεθος της κοινότητας του κάθε Framework;
13. Θεωρείτε σημαντικό παράγοντα επιλογής την υποστήριξη ενός Framework από μια μεγάλη εταιρία;
14. Θεωρείτε σημαντικό παράγοντα επιλογής το μερίδιο αγοράς που έχει το κάθε Framework;

3.3.2 Ερωτήματα για τους στόχους και την φύση του Project

1. Ποιο θεωρείτε πως θα είναι το μέγεθος του συνόλου των δεδομένων που θα διαχειρίζεται η εφαρμογή (State Management);
2. Είναι βασικός στόχος του Project η ανάπτυξη της εφαρμογής και για άλλες πλατφόρμες εκτός του Web Browser, όπως native εφαρμογές για κινητές συσκευές/tablets κτλ (Cross Platform);
3. Θα θέλατε το Framework που θα χρησιμοποιήσετε να έχει ίδια λογική ανάπτυξης εφαρμογών με τον αντικειμενοστραφή προγραμματισμό (TypeScript);
4. Έχετε ως πρωταρχικό στόχο την επίτευξη του καλύτερου δυνατού SEO (Server Side Rendering);
5. Σκοπεύετε να αναπτύξετε μια multipage εφαρμογή με πολλές ξεχωριστές σελίδες (Routing);
6. Επιθυμείτε οι επιδόσεις της εφαρμογής να είναι οι καλύτερες δυνατές μεταξύ όλων των Framework σύγκρισης;

7. Είναι πρωταρχικός σας στόχος η επίτευξη του μικρότερου δυνατού μεγέθους για την εφαρμογή σας;
8. Έχετε γνώσεις συναρτησιακού προγραμματισμού και θέλετε να αναπτύξετε διαδικτυακές εφαρμογές με την ίδια λογική;
9. Θεωρείται ότι το μέγεθος της εφαρμογής που θα αναπτύξετε θα είναι μικρό;

3.3.3 Γενικά Ερωτήματα για συγκεκριμένα Frameworks και τα χαρακτηριστικά τους

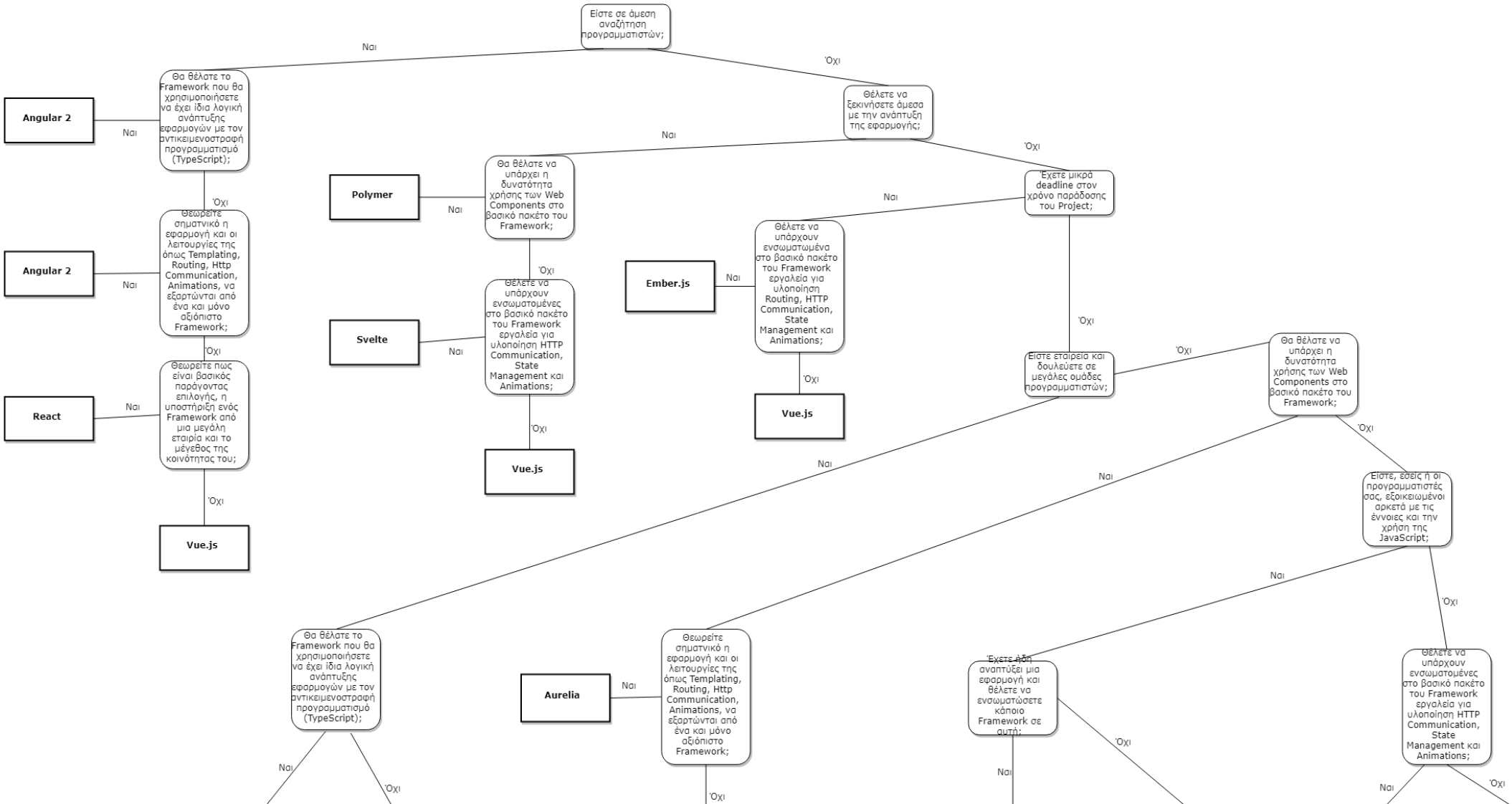
1. Προγραμματίζετε σε React αλλά θέλετε να δοκιμάστε μια εναλλακτική με βελτιωμένες επιδόσεις και μέγεθος;
2. Χρησιμοποιείτε Angular 1 ή Angular 2 και θέλετε να δοκιμάσετε μια διαφορετική εναλλακτική σε παρόμοια λογική ανάπτυξης;

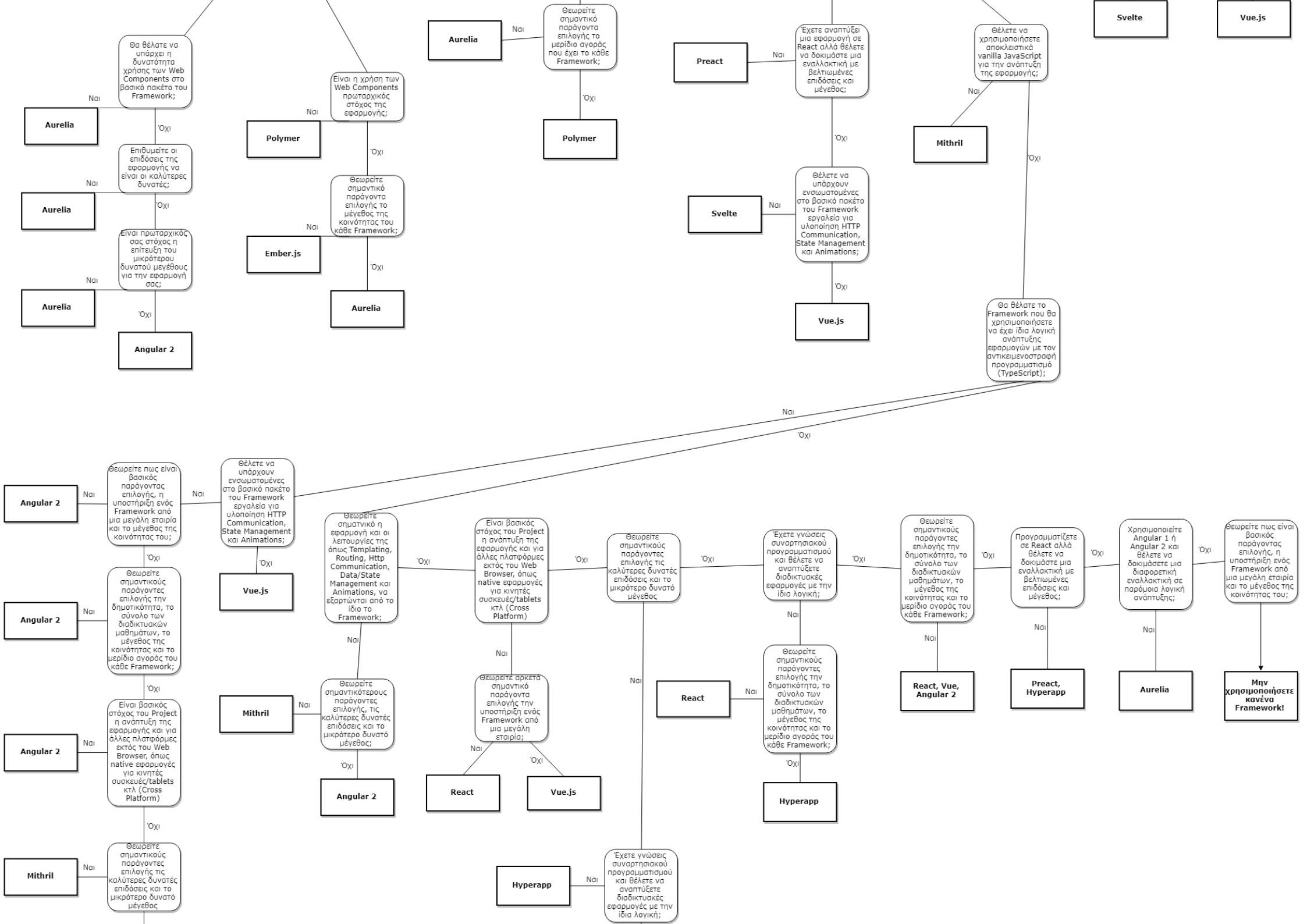
3.3.4 Τεχνικά ερωτήματα για την εφαρμογή

1. Είστε ικανοποιημένος με την λειτουργία των REST APIs ή ψάχνετε έναν τρόπο για αποτελεσματικότερη διαχείριση δεδομένων;
2. Σας ενδιαφέρει η ενσωμάτωση και χρήση τεχνολογιών των Web Components στην εφαρμογή που θέλετε να αναπτύξετε;
3. Σας ενοχλεί η χρήση μεγάλου μέρους Boilerplated κώδικα;
4. Θέλετε η εφαρμογή να έχει υλοποιημένες μεθόδους για την άμεση και εύκολη δημιουργία user authentication;
5. Θέλετε να δημιουργείτε εύκολα και γρήγορα REST APIs για την εφαρμογή;
6. Σας ενδιαφέρουν περισσότερο τεχνολογίες όπως ORM, auto generated APIs και WebSockets;

3.4 Δημιουργία πλήρους Δέντρου/Χάρτη Αποφάσεων

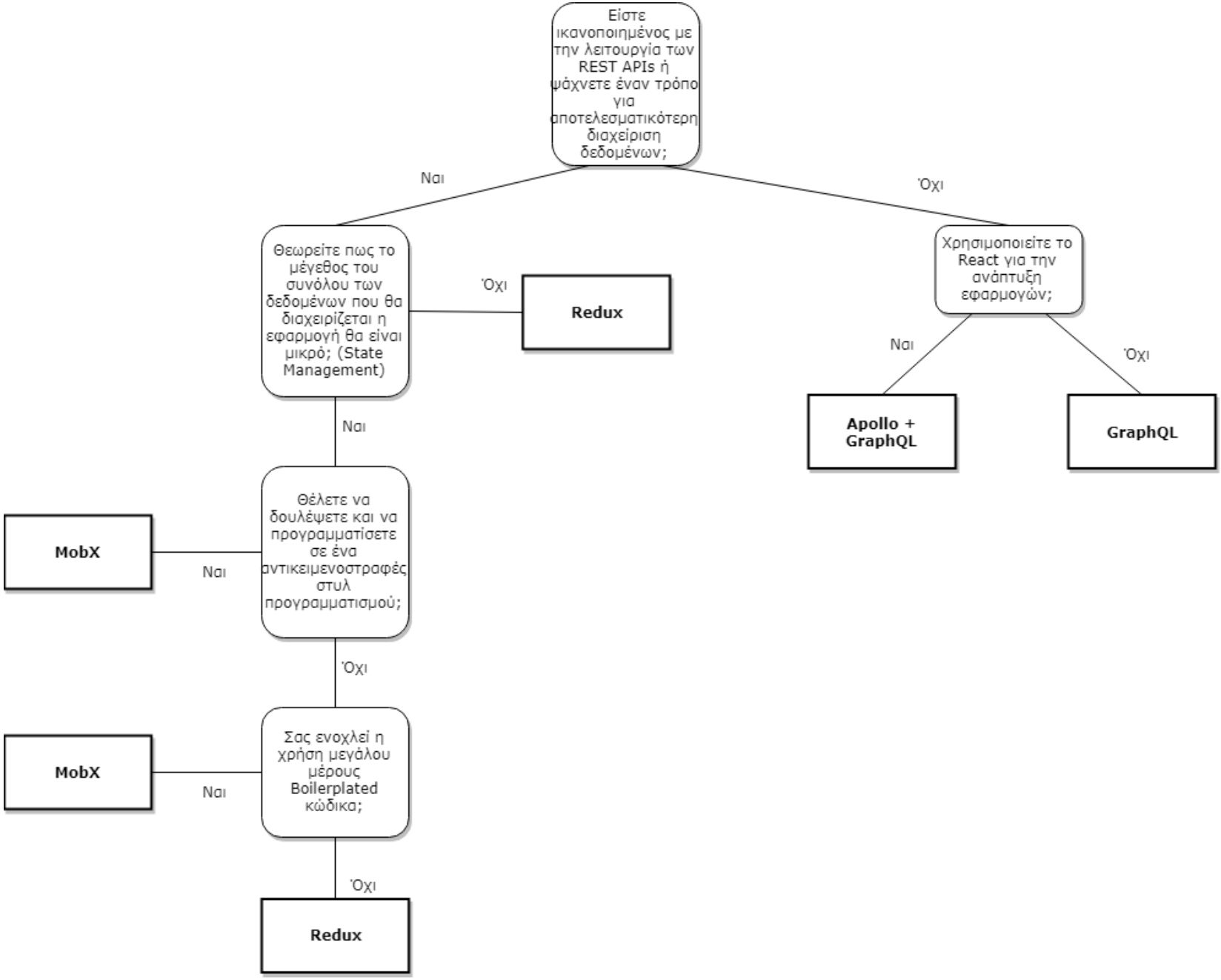
3.4.1 Front-End Frameworks



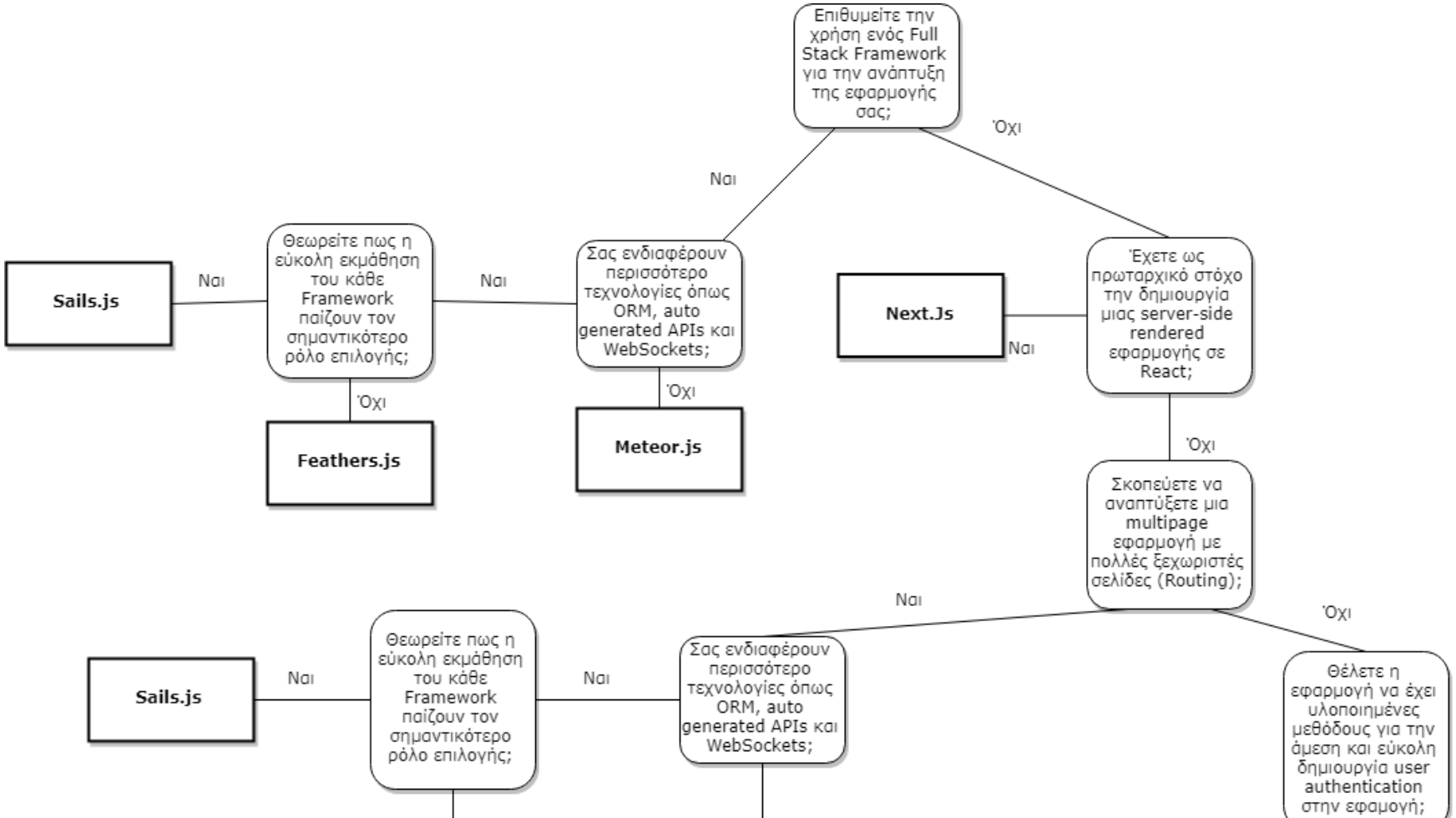

Θα θέλατε να υπάρχει η δυνατότητα χρήσης των Web Components στο βασικό πακέτο του Framework;
Ναι
Aurelia
Όχι
Επιθυμείτε οι επιδόσεις της εφαρμογής να είναι οι καλύτερες δυνατές;
Ναι
Aurelia
Όχι
Είναι πρωταρχικός σας στόχος η επίτευξη του μικρότερου δυνατού μεγέθους για την εφαρμογή σας;
Ναι
Aurelia
Όχι
Angular 2
Είναι η χρήση των Web Components πρωταρχικός στόχος της εφαρμογής;
Ναι
Polymer
Όχι
Θεωρείτε σημαντικό παράγοντα επιλογής το μέγεθος της κοινότητας του κάθε Framework;
Ναι
Ember.js
Όχι
Aurelia
Θεωρείτε σημαντικό παράγοντα επιλογής το μερίδιο αγοράς που έχει το κάθε Framework;
Ναι
Aurelia
Όχι
Polymer
Έχετε αναπτύξει μια εφαρμογή σε React αλλά θέλετε να δοκιμάστε μια εναλλακτική με βελτιωμένες επιδόσεις και μέγεθος;
Ναι
Preact
Όχι
Θέλετε να υπάρχουν ενσωματομένες στο βασικό πακέτο του Framework εργαλεία για υλοποίηση HTTP Communication, State Management και Animations;
Ναι
Svelte
Όχι
Vue.js
Θέλετε να χρησιμοποιήσετε αποκλειστικά vanilla JavaScript για την ανάπτυξη της εφαρμογής;
Ναι
Mithril
Όχι
Svelte
Vue.js
Θα θέλατε το Framework που θα χρησιμοποιήσετε να έχει ίδια λογική ανάπτυξης εφαρμογών με τον αντικειμενοστραφή προγραμματισμό (TypeScript);
Ναι
Όχι
Angular 2
Θεωρείτε πως είναι βασικός παράγοντας επιλογής, η υποστήριξη ενός Framework από μια μεγάλη εταιρία και το μέγεθος της κοινότητας του;
Ναι
Angular 2
Θεωρείτε σημαντικούς παράγοντες επιλογής την δημοτικότητα, το σύνολο των διαδικτυακών μαθημάτων, το μέγεθος της κοινότητας και το μερίδιο αγοράς του κάθε Framework;
Ναι
Angular 2
Είναι βασικός στόχος του Project η ανάπτυξη της εφαρμογής και για άλλες πλατφόρμες εκτός του Web Browser, όπως native εφαρμογές για κινητές συσκευές/tablets κτλ (Cross Platform)
Ναι
Mithril
Θεωρείτε σημαντικούς παράγοντες επιλογής τις καλύτερες δυνατές επιδόσεις και το μικρότερο δυνατό μέγεθος
Ναι
Θέλετε να υπάρχουν ενσωματομένες στο βασικό πακέτο του Framework εργαλεία για υλοποίηση HTTP Communication, State Management και Animations, να εξαρτώνται από το ίδιο το Framework;
Ναι
Vue.js
Θεωρείτε σημαντικό η εφαρμογή και οι λειτουργίες της όπως Templating, Routing, Http Communication, Data/State Management και Animations, να εξαρτώνται από το ίδιο το Framework;
Ναι
Θεωρείτε σημαντικότερους παράγοντες επιλογής, τις καλύτερες δυνατές επιδόσεις και το μικρότερο δυνατό μέγεθος;
Ναι
Angular 2
Είναι βασικός στόχος του Project η ανάπτυξη της εφαρμογής και για άλλες πλατφόρμες εκτός του Web Browser, όπως native εφαρμογές για κινητές συσκευές/tablets κτλ (Cross Platform)
Ναι
Θεωρείτε αρκετά σημαντικό παράγοντα επιλογής την υποστήριξη ενός Framework από μια μεγάλη εταιρία;
Ναι
React
Όχι
Vue.js
Θεωρείτε σημαντικούς παράγοντες επιλογής τις καλύτερες δυνατές επιδόσεις και το μικρότερο δυνατό μέγεθος
Ναι
Hyperapp
Ναι
Έχετε γνώσεις συναρτησιακού προγραμματισμού και θέλετε να αναπτύξετε διαδικτυακές εφαρμογές με την ίδια λογική;
Ναι
React
Ναι
Θεωρείτε σημαντικούς παράγοντες επιλογής την δημοτικότητα, το σύνολο των διαδικτυακών μαθημάτων, το μέγεθος της κοινότητας και το μερίδιο αγοράς του κάθε Framework;
Ναι
Hyperapp
Όχι
Έχετε γνώσεις συναρτησιακού προγραμματισμού και θέλετε να αναπτύξετε διαδικτυακές εφαρμογές με την ίδια λογική;
Ναι
Θεωρείτε σημαντικούς παράγοντες επιλογής την δημοτικότητα, το σύνολο των διαδικτυακών μαθημάτων, το μέγεθος της κοινότητας και το μερίδιο αγοράς του κάθε Framework;
Ναι
React, Vue, Angular 2
Όχι
Προγραμματίζετε σε React αλλά θέλετε να δοκιμάστε μια εναλλακτική με βελτιωμένες επιδόσεις και μέγεθος;
Ναι
Preact, Hyperapp
Όχι
Χρησιμοποιείτε Angular 1 ή Angular 2 και θέλετε να δοκιμάσετε μια διαφορετική εναλλακτική σε παρόμοια λογική ανάπτυξης;
Ναι
Aurelia
Όχι
Θεωρείτε πως είναι βασικός παράγοντας επιλογής, η υποστήριξη ενός Framework από μια μεγάλη εταιρία και το μέγεθος της κοινότητας του;
Μην Χρησιμοποιήσετε κανένα Framework!

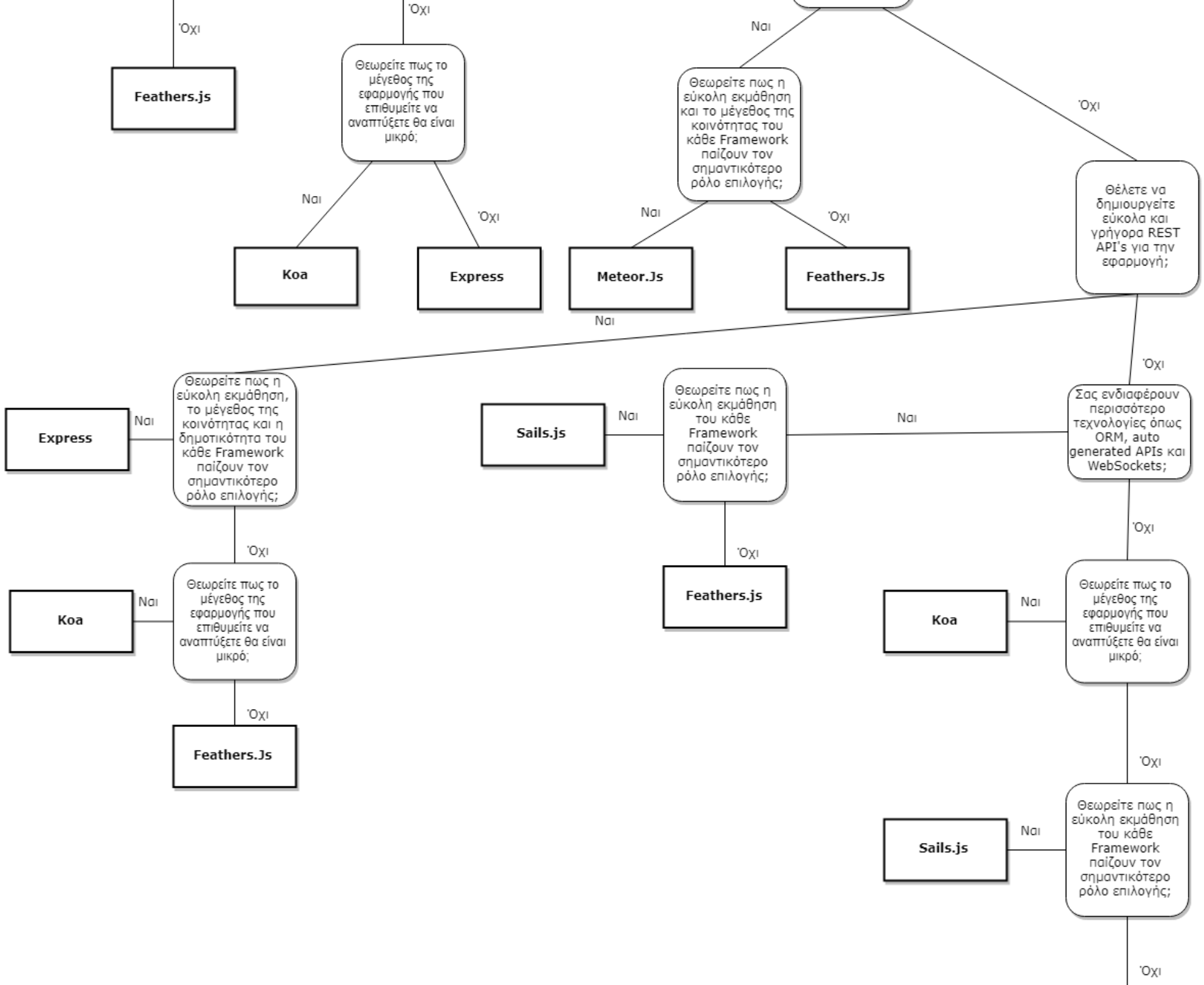

Όχι
Aurelia
Όχι
Έχετε αναπτύξει μια εφαρμογή σε React αλλά θέλετε να δοκιμάστε μια εναλλακτική με βελτιωμένες επιδόσεις και μέγεθος;
Ναι
Preact
Όχι
Vue.js

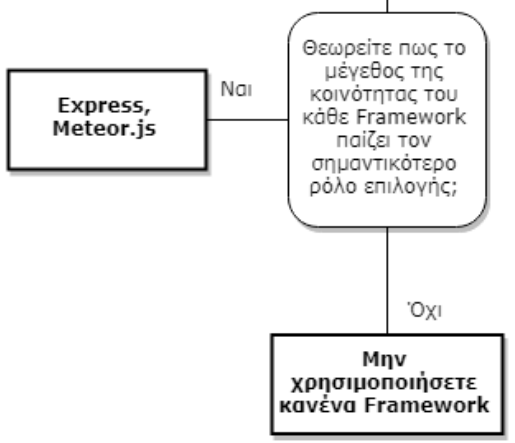
3.4.2 Data-Layer Frameworks



3.4.3 Back-End Frameworks




Όχι
Feathers.js
Όχι
Θεωρείτε πως το μέγεθος της εφαρμογής που επιθυμείτε να αναπτύξετε θα είναι μικρό;
Ναι
Koa
Όχι
Express
Ναι
Θεωρείτε πως η εύκολη εκμάθηση και το μέγεθος της κοινότητας του κάθε Framework παίζουν τον σημαντικότερο ρόλο επιλογής;
Ναι
Meteor.Js
Όχι
Feathers.Js
Όχι
Θέλετε να δημιουργείτε εύκολα και γρήγορα REST API's για την εφαρμογή;
Ναι
Express
Ναι
Θεωρείτε πως η εύκολη εκμάθηση, το μέγεθος της κοινότητας και η δημοτικότητα του κάθε Framework παίζουν τον σημαντικότερο ρόλο επιλογής;
Όχι
Koa
Ναι
Θεωρείτε πως το μέγεθος της εφαρμογής που επιθυμείτε να αναπτύξετε θα είναι μικρό;
Όχι
Feathers.Js
Sails.js
Ναι
Θεωρείτε πως η εύκολη εκμάθηση του κάθε Framework παίζουν τον σημαντικότερο ρόλο επιλογής;
Όχι
Feathers.js
Ναι
Όχι
Σας ενδιαφέρουν περισσότερο τεχνολογίες όπως ORM, auto generated APIs και WebSockets;
Όχι
Koa
Ναι
Θεωρείτε πως το μέγεθος της εφαρμογής που επιθυμείτε να αναπτύξετε θα είναι μικρό;
Όχι
Sails.js
Ναι
Θεωρείτε πως η εύκολη εκμάθηση του κάθε Framework παίζουν τον σημαντικότερο ρόλο επιλογής;
Όχι


Express, Meteor.js
Ναι
Θεωρείτε πως το μέγεθος της κοινότητας του κάθε Framework παίζει τον σημαντικότερο ρόλο επιλογής;
Όχι
Μην χρησιμοποιήσετε κανένα Framework

4 Υλοποίηση της Web Εφαρμογής

4.1 Σκοπός και βασικές λειτουργίες της εφαρμογής

Η συγκεκριμένη web εφαρμογή που υλοποιήθηκε, έχει ως στόχο την ενημέρωση των προγραμματιστών οι οποίοι επιθυμούν να χρησιμοποιήσουν κάποιο Framework για την ανάπτυξη μιας δικής τους web εφαρμογής. Με την πληθώρα και τις τόσες διαφορετικές επιλογές που έχει ο χρήστης, τις περισσότερες φορές δυσκολεύεται στην επιλογή ή δεν γνωρίζει το οφέλη της κάθε μιας. Με το συγκεκριμένο εργαλείο, οι χρήστες μετά από μια σειρά ερωτήσεων, μπορούν να ενημερωθούν και να μάθουν για το Framework που ταιριάζει κατά πάσα πιθανότητα καλύτερα στην περίπτωση τους με αποτελέσματα εξαγώμενα ανάλογα με τις απαιτήσεις που έχει ο καθένας. Συνοπτικά, ο ρόλος της εφαρμογής θα είναι συμβουλευτικός προς τους χρήστες.

Πιο συγκεκριμένα, μετά την έρευνα που παρουσιάστηκε στις προηγούμενες ενότητες και κεφάλαια της εργασίας, προέκυψαν συγκεκριμένα συμπεράσματα και ερωτήσεις προς τους χρήστες. Όλες οι πληροφορίες που συγκεντρώθηκαν για το κάθε Framework ξεχωριστά αλλά και συγκριτικά, προέρχονται από απόψεις προγραμματιστών διαδικτύου, σχόλια τους, άρθρα τους, αλλά και από τους επίσημους ιστοτόπους του κάθε Framework ξεχωριστά. Στην συνέχεια με μελέτη και σύγκριση αποτελεσμάτων δημιουργήθηκαν οι συσχετίσεις των Framework με συγκεκριμένα ερωτήματα και αποτελέσματα. Ωστόσο, σε καμία περίπτωση τα αποτελέσματα που προκύπτουν δεν είναι απόλυτα και δεσμευτικά. Το κάθε Framework μπορεί να χρησιμοποιηθεί πραγματικά όπως ο χρήστης επιθυμεί και να πραγματοποιηθεί με την χρήση του κάθε απαιτούμενο έργο λογισμικού. Η συγκεκριμένη έρευνα απλά δείχνει πως κάποια μπορεί να είναι καταλληλότερα από τα άλλα σε συγκεκριμένες συνθήκες και απαιτήσεις.

Αναλυτικότερα και όσων αφορά την εφαρμογή, με την εκτέλεση της, εμφανίζεται η κεντρική οθόνη-menu στον χρήστη. Το μενού που εμφανίζεται, περιέχει τρείς καρτέλες, κάθε μια για διαφορετική κατηγορία Frameworks (Front-End, Data-Layer, Back-End). Πατώντας στο αντίστοιχο κουμπί, ανοίγει ένα νέο παράθυρο στο οποίο ξεκινάει η σχεδίαση του γραφήματος του δέντρου αποφάσεων με τις ερωτήσεις προς τον χρήστη. Απαντώντας σε αυτές με 'ΝΑΙ' ή 'ΟΧΙ', ο χρήστης τελικά λαμβάνει σαν αποτέλεσμα ένα ή περισσότερα Frameworks. Τέλος μπορεί να αποθηκεύσει το γράφημα σε διάφορες μορφές αρχείων όπως PDF, SVG, CSV κ.α.

4.2 Εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν

Ένα από τα βασικά συμπεράσματα το οποίο προέκυψε μετά από την συγκεκριμένη έρευνα είναι πως σε πολλές περιπτώσεις ο χρήστης δεν χρειάζεται να μπει στην διαδικασία επιλογής Framework για να αναπτύξει ένα έργο. Πολλές φορές η απλή HTML5 με τις τεχνολογίες της, HTML, CSS, JS είναι αρκετή για να παρέχει καθαρό και εύκολα συντηρούμενο κώδικα. Αυτό κυρίως επιτυγχάνεται καλύτερα σε περιπτώσεις μικρών σε μέγεθος Project, όπως το συγκεκριμένο που αναπτύχθηκε στα πλαίσια της έρευνας. Η μόνη JavaScript βιβλιοθήκη-Framework που χρησιμοποιήθηκε, ήταν η 'jQuery', μια τεχνολογία ισχυρή, απλή και διαχρονική.

Πιο συγκεκριμένα, τα στοιχεία που περιέχει το Project είναι:

- HTML File (Index.html): το βασικό αρχείο που υπάρχει η σχεδίαση του Website με όλα τα Elements που το απαρτίζουν.
- CSS Files: αρχεία που αφορούν το styling των παραπάνω Elements και την μορφοποίηση της τελικής εικόνας της ιστοσελίδας.
- JS Files: αρχεία JavaScript τα οποία ευθύνονται για την λειτουργικότητα της ιστοσελίδας και της εφαρμογής.
- JSON Files: αρχεία τα οποία εμπεριέχουν σε JSON μορφή τις απαραίτητες πληροφορίες για την σχεδίαση των κόμβων και του συνολικού δένδρου αποφάσεων που παρέχεται στον χρήστη.
- Αρχεία εικόνων: τα οποία περιέχουν τις εικόνες που εμφανίζονται στο βασικό Menu της εφαρμογής.

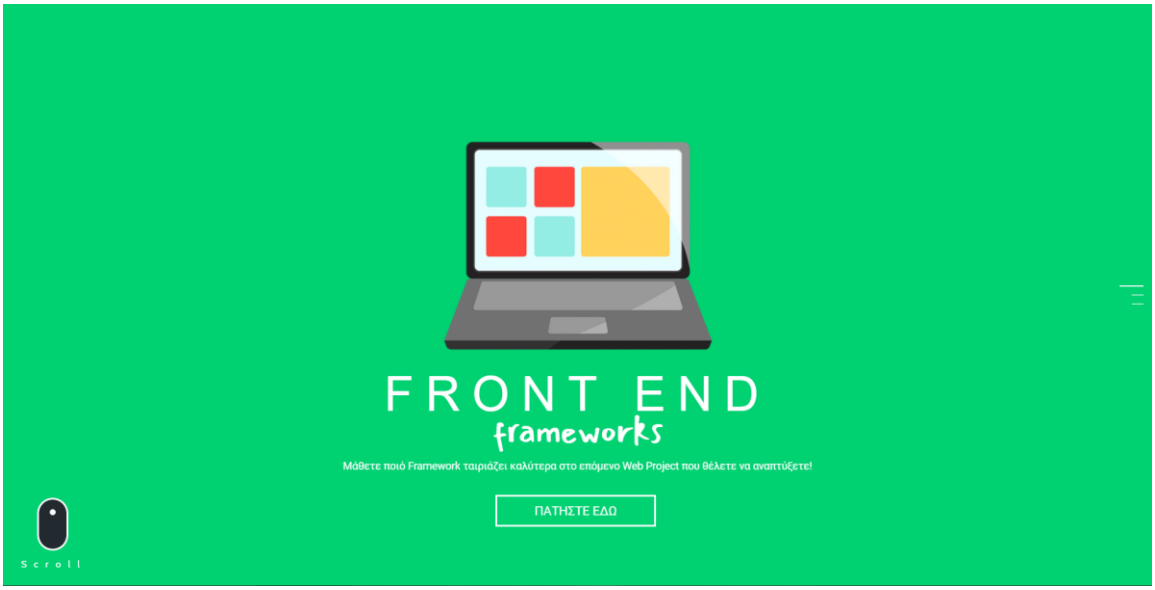
Σχετικά με τις έξτρα βιβλιοθήκες JavaScript, χρησιμοποιήθηκαν οι 'Slick.js' και η 'Mousewheel.js' για την επίτευξη των animations και της λειτουργίας της εναλλαγής καρτελών με την χρήση του Mousewheel..

Όσων αφορά τα γραφήματα συγκεκριμένα, χρησιμοποιήθηκε μια τρίτη βιβλιοθήκη η οποία εξειδικεύεται με συγκεκριμένες συναρτήσεις και μεθόδους στην σχεδίαση και διαχείριση τέτοιων γραφημάτων όπως αυτό του δένδρου αποφάσεων. Η συγκεκριμένη βιβλιοθήκη ονομάζεται 'OrgChart.js' και παρέχεται από την BALKANGraph (BALKANGraph, 2019). Η συγκεκριμένη βιβλιοθήκη δημιουργεί SVG σχήματα με δεδομένα που ο χρήστης εισάγει και τα εμφανίζει σε σημεία της ιστοσελίδας που ο χρήστης της ορίζει.

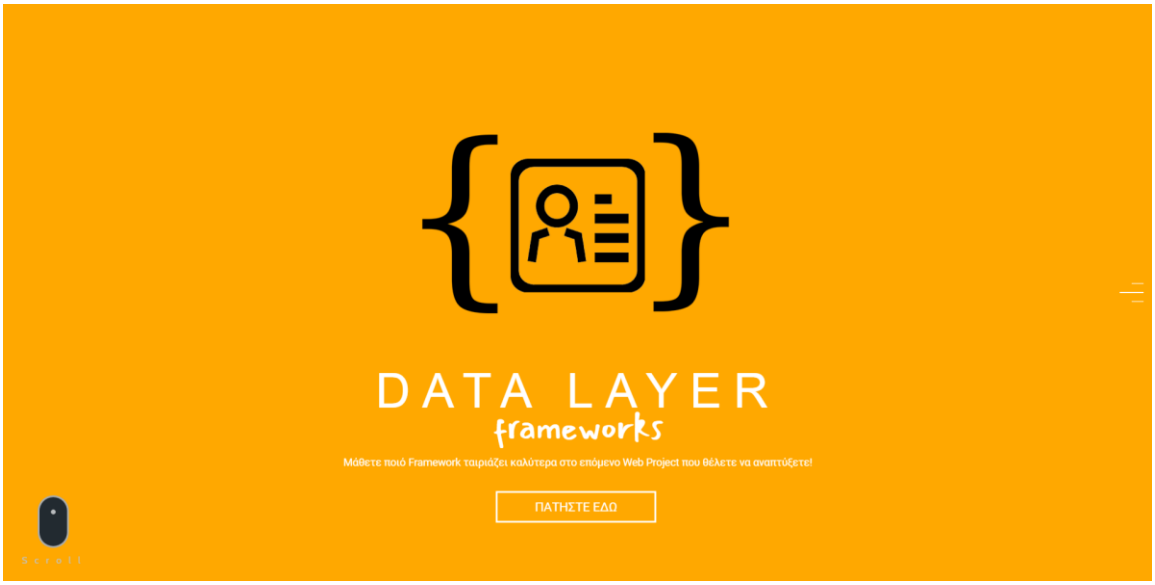
Δημιουργήθηκαν κόμβοι με συγκεκριμένη μορφοποίηση, σχήμα, χρώμα, μέγεθος κτλ, και δόθηκαν δεδομένα σε μορφή JSON αρχείων για την απεικόνιση του κάθε δένδρου αποφάσεων ξεχωριστά. Συνολικά δημιουργήθηκαν τρία δένδρα, ένα για κάθε κατηγορία Framework. Η λειτουργικότητα και ο τρόπος δημιουργίας του κάθε δένδρου είναι κοινοί. Το μόνο που αλλάζει είναι τα δεδομένα εισαγωγής στο κάθε ένα, με αποτέλεσμα να δημιουργήθηκαν και τρία διαφορετικά JSON αρχεία που περιέχουν τα δεδομένα αυτά.

4.3 Περιγραφή σχεδίασης και τρόπου λειτουργίας

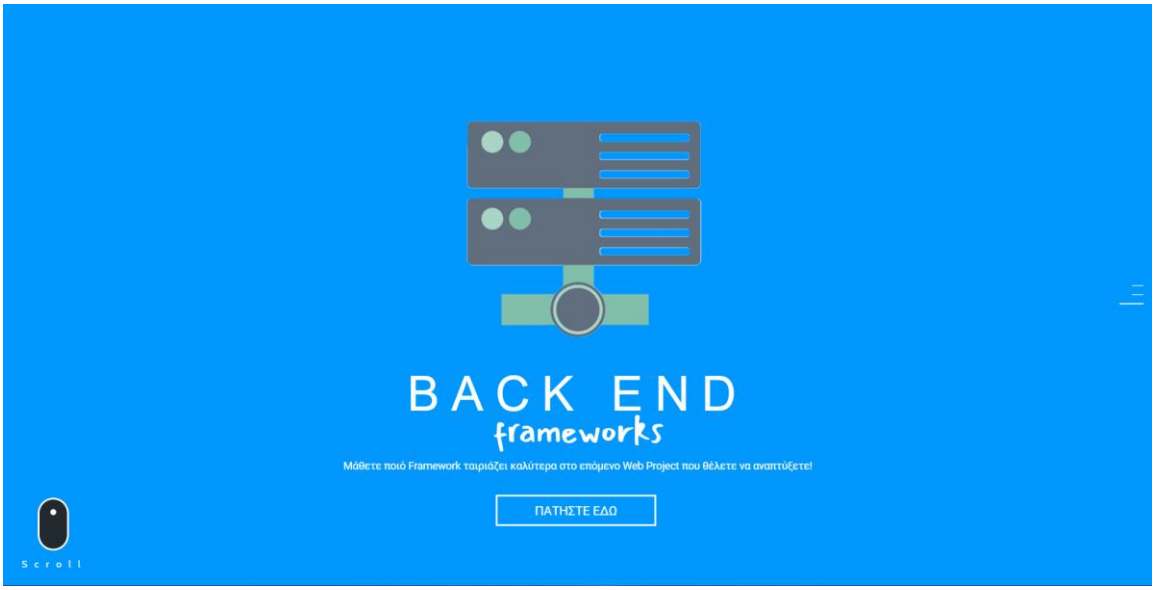
Το menu της εφαρμογής είναι πολύ απλό στην σχεδίαση του. Αποτελείται από μια εικόνα με μονόχρωμο φόντο, μια λεζάντα κειμένου και ένα Button το οποίο κατευθύνει τον χρήστη στην εφαρμογή που του προτείνει το εκάστοτε JavaScript Framework. Συμπληρωματικά στην οθόνη υπάρχουν ένα μικρό animation ενός Mouse το οποίο προτρέπει τον χρήστη να χρησιμοποιήσει το Scroll Button για να περιηγηθεί και στο υπόλοιπο Menu, καθώς και κάποια μικρά Buttons στην δεξιά πλευρά της σελίδας το οποία διευκρινίζουν το ποιά καρτέλα του Menu που είναι αυτή την στιγμή ανοικτή. Παρακάτω, στις Εικόνες 4-1 - 4-3, παρουσιάζονται στιγμιότυπα από τις τρείς διαφορετικές καρτέλες του Menu.



Εικόνα 4-1: Menu των Front-End frameworks



Εικόνα 4-2: Menu των Data-Layer frameworks

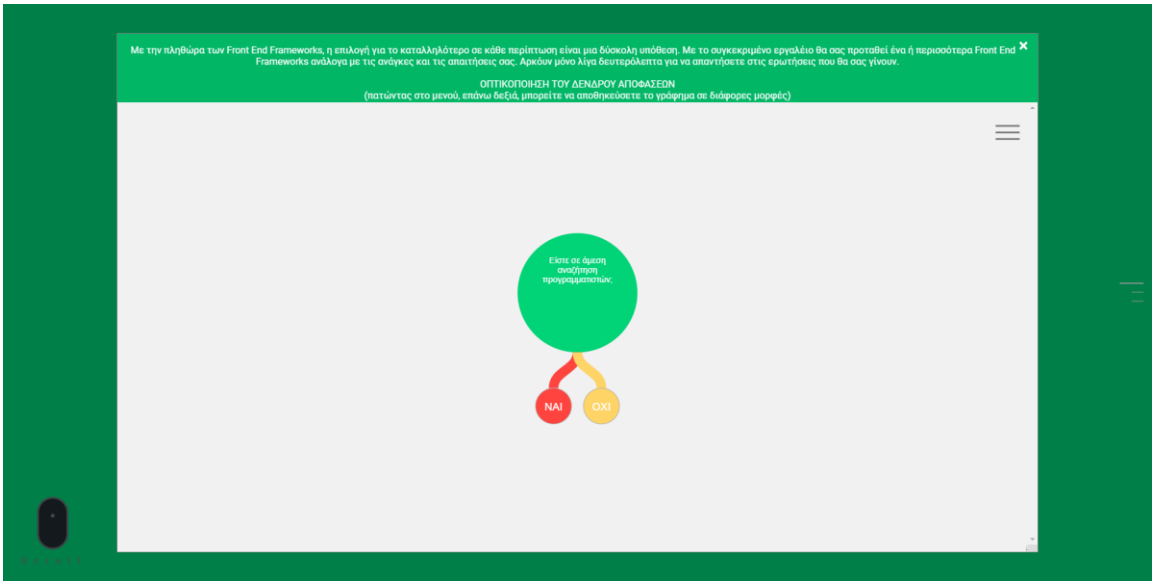


Εικόνα 4-3: Menu των Back-End frameworks

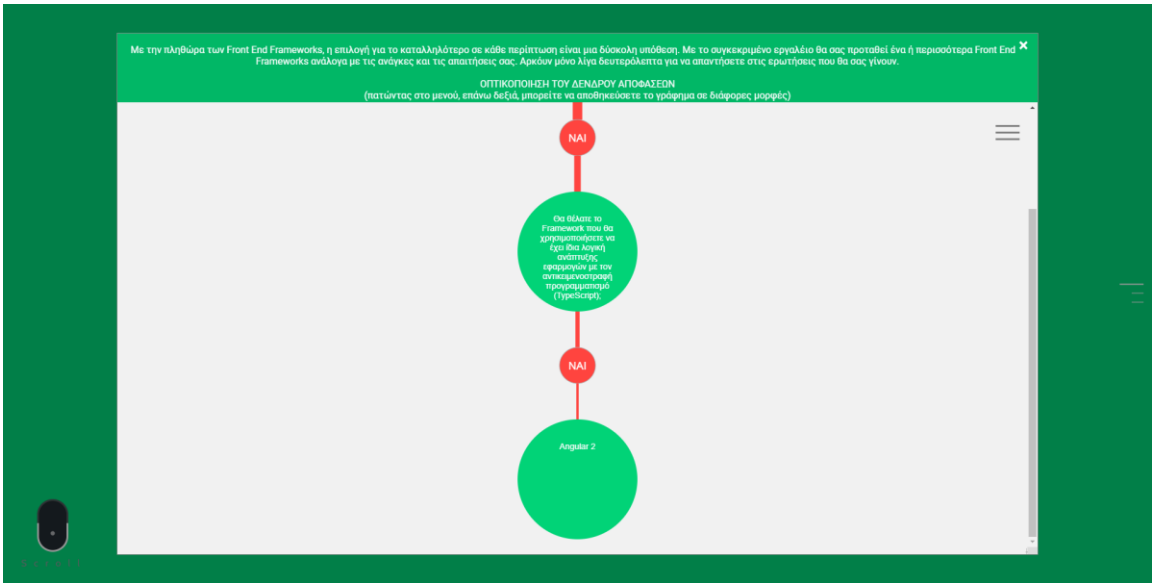
Στην συνέχεια ο χρήστης πατώντας το κουμπί, και ανάλογα με την οθόνη στην οποία βρίσκεται, εμφανίζεται ένα Modal παράθυρο, το οποίο περιέχει πληροφορίες και ξεκινάει η σχεδίαση του γραφήματος του δένδρου με τις ερωτήσεις προς αυτόν. Χρησιμοποιείται ως δείγμα παρουσίασης το δένδρο απόφασης των Front-End Frameworks. Όπως φαίνεται στην Εικόνα 4-4 το Modal εμφανίζεται στην οθόνη και περιέχει:

- Header: μέσα στο οποίο υπάρχουν πληροφορίες για το σύνολο της κατηγορίας των Framework που αναλύονται όπως και βασικές πληροφορίες για το γράφημα που εμφανίζεται παρακάτω.
- Body: στο μεγαλύτερο μέρος του Modal, υπάρχει η περιοχή που σχεδιάζεται το γράφημα, καθώς και ένα κουμπί για το μενού το οποίο μπορεί να το εξάγει σε διάφορες μορφές αρχείων.

Όσων αφορά το γράφημα, αυτό αποτελείται από SVG στοιχεία (Circles, Texts, Lines κ.α) τα οποία συνδυαζόμενα δημιουργούν το δένδρο αποφάσεων. Όπως παρουσιάζεται και στην εικόνα αρχικά εμφανίζεται στον χρήστη μόνο ο πρώτος κόμβος του δένδρου, μαζί με δύο έξτρα κόμβους της επιλογής ('ΝΑΙ' και 'ΟΧΙ'). Ο χρήστης πρέπει να πατήσει με το αριστερό κλικ του Mouse στον κόμβο του 'ΝΑΙ' ή του 'ΟΧΙ' ανάλογα με την απάντησή του στην ερώτηση που εμφανίζεται. Επιλέγοντας μια απάντηση το δένδρο αρχίζει να αναπτύσσεται και εμφανίζεται ο επόμενος κόμβος μαζί με την διαδρομή που τον συνδέει με τους προηγούμενους, όπως φαίνεται στην Εικόνα 4-5. Τελικά όταν η ανάλογη διαδρομή φτάσει στο τέλος της, εμφανίζεται η επιλογή του προτεινόμενου Framework στον χρήστη. Εάν ο χρήστης θέλει να αλλάξει μια επιλογή του μπορεί να πατήσει στον επιλεγμένο κόμβο 'ΝΑΙ' ή 'ΟΧΙ' και το δένδρο θα συνεχίσει να σχεδιάζεται ξανά από το συγκεκριμένο σημείο. Στο τέλος και εφόσον ο χρήστης θέλει να κρατήσει αποθηκευμένη αυτήν την διαδρομή στο δένδρο με τις ερωτήσεις, μπορεί πατήσει στο κουμπί του μενού στην επάνω δεξιά πλευρά του Body, και να αποθηκεύσει σε 4 διαφορετικές μορφές το γράφημα. Οι προσφερόμενες μορφές αποθήκευσης είναι: PDF, PNG, SVG, CSV. Στην συνέχεια, μπορεί να κλείσει το Modal, πιέζοντας με αριστερό κλικ στο Mouse, οποιαδήποτε περιοχή γύρω από το Modal ή στο κουμπί με ένδειξη 'X', που βρίσκεται επάνω δεξιά στο Header. Σε περίπτωση που ο χρήστης προσπαθήσει να ξανανοίξει το Modal πατώντας στο ίδιο κουμπί, το γράφημα θα είναι συνεχίσει από εκεί που το άφησε και δεν διαγράφεται. Εάν επιθυμεί την επανεκκίνηση της διαδικασίας με τον σχεδιασμό του δένδρου από την αρχή, τότε αρκεί να με το Scroll Button του Mouse να ανέβει στον πρώτο κόμβο του δένδρου και να πατήσει τον κόμβο με το πρώτο 'ΝΑΙ'. Εναλλακτικά μπορεί να επανεκκινίσει την σελίδα.



Εικόνα 4-4: Modal - 1ο άνοιγμα



Εικόνα 4-5: Modal - Δένδρο απόφασης

Από τεχνικής πλευράς η διαδικασία δημιουργίας του δένδρου με την χρήση της JavaScript πραγματοποιήθηκε ως εξής. Αρχικά και ακολουθώντας τις οδηγίες που παρέχονται από το documentation του OrgChart.js, το οποίο αναφέρθηκε παραπάνω πως χρησιμοποιήθηκε, ορίζονται με συγκεκριμένο τρόπο τα Components μαζί με τις ιδιότητες και τα χαρακτηριστικά του τα οποία θα απαρτίζουν το δένδρο. Έπειτα με συγκεκριμένη συνάρτηση ξεκινάει η δημιουργία του γραφήματος. Η βασική λογική είναι

πως όταν ο χρήστης επιλέγει έναν νέο κόμβο 'ΝΑΙ' ή 'ΟΧΙ', το πραγματοποιείται μια διαδικασία κατά την οποία εμφανίζεται στο γράφημα και στην οθόνη ο αντίστοιχος νέος κόμβος προς προβολή. Από την άλλη όταν ο χρήστης αποεπιλέξει έναν κόμβο με 'ΝΑΙ' ή 'ΟΧΙ', τότε εκτελείται η ανάλογη διαδικασία για να κρυφτούν οι επόμενοι κόμβοι ναι να σχεδιαστούν εκ νέου με τις νέες επιλογές του.

Σημαντικό ρόλο σε όλη αυτή την διαδικασία παίζουν τα δεδομένα τα οποία έχουν εισαχθεί και ουσιαστικά τα ίδια παρέχουν οδηγίες στις συναρτήσεις για την σειρά εμφάνισης των κόμβων. Πιο συγκεκριμένα τα δεδομένα, όπως προαναφέρθηκε, εισάγονται στην εφαρμογή από JSON αρχεία στα οποία βρίσκονται αποθηκευμένα σε συγκεκριμένη μορφή, όπως φαίνεται στην Εικόνα 2.31. Πιο αναλυτικά, ο κάθε κόμβος πρέπει να έχει:

- ένα ξεχωριστό **id,** το οποίο ουσιαστικά είναι ένας αύξων αριθμός και αντιπροσωπεύει τους κόμβους.
- ένα ξεχωριστό **pid** (εκτός από τον 1ο κόμβο), το οποίο είναι ένας αριθμός που καθορίζει την σύνδεση των κόμβων μεταξύ τους.
- εάν πρόκειται για έναν 'ΝΑΙ' ή 'ΟΧΙ' κόμβο τότε πρέπει να δοθεί και η αντίστοιχη τιμή στο πεδίο **tags.**
- τέλος το πεδίο **text** το οποίο είναι ένα String που θα περιέχει το κείμενο του κάθε κόμβου.

Όσων αφορά τον τρόπο με τον οποίο φορτώνονται και καλούνται τα JSON αρχεία, χρησιμοποιείται η συνηθισμένη τεχνική των AJAX Requests. Τα αρχεία όντας αποθηκευμένα στον Server, ζητούνται από τον Client και ο πρώτος τους τα παρέχει προς ανάγνωση. Τα δεδομένα για τις τρείς διαφορετικές κατηγορίες Framework βρίσκονται αποθηκευμένα σε τρία διαφορετικά αρχεία που το καθένα έχει την μορφή της Εικόνας 4-6. Έπειτα μέσω της JavaScript και ανάλογα με το κουμπί που ο χρήστης πιέζει στο αρχικό Menu, καλείται το αρχείο με το αντίστοιχο όνομα. Τέλος η δυνατότητα για απευθείας αποθήκευση του δένδρου σε διάφορες μορφές αρχείων παρέχεται αυτόματα μέσω την βιβλιοθήκης OrgChart.js.

frontend.json

```
[
   {
      "id": 1,
      "text": "Είστε σε άμεση αναζήτηση προγραμματιστών;"
   },
   {
      "id": 2,
      "pid": 1,
      "tags": [
         "yes"
      ],
      "text": "NAI"
   },
   {
      "id": 3,
      "pid": 1,
      "tags": [
         "no"
      ],
      "text": "OXI"
   },
   {
      "id": 4,
      "pid": 2,
      "text": "Θα θέλατε το Framework που θα χρησιμοποιήσετε να έχει ίδια λογική ανάπτυξης εφαρμογών με τον αντικειμενοστραφή προγραμματισμό (TypeScript);"
   },
   {
      "id": 5,
      "pid": 4,
      "tags": [
         "yes"
      ],
      "text": "NAI"
   },
   {
      "id": 6,
      "pid": 4,
      "tags": [
         "no"
      ],
      "text": "OXI"
   },
   {
      "id": 7,
      "pid": 5,
      "text": "Angular 2"
   },
```

Εικόνα 4-6: Δείγμα της JSON αναπαράστασης των δεδομένων της εφαρμογής

5 Επίλογος

5.1 Σύνοψη και συμπεράσματα

Στα παραπάνω κεφάλαια της έρευνας παρουσιάστηκαν αναλυτικά όλα τα στάδια της από την αρχή μέχρι και την υλοποίηση της Web εφαρμογής της. Αρχικά πραγματοποιήθηκε μια εισαγωγή για την σημαντικότητα της συγκεκριμένης έρευνας και των προβλημάτων τα οποία θεραπεύει. Αναφέρθηκαν επίσης οι στόχοι της και ορίστηκαν ορολογίες που βοηθούν το χρήστη να αντιληφθεί καλύτερα κάποιες έννοιες με τις οποίες ενδεχομένως να έρχεται σε επαφή για πρώτη φορά. Έπειτα επιλέχθηκαν με συγκεκριμένα κριτήρια τα Frameworks προς ανάλυση και κύρια πηγή την έρευνα του StateofJs (2018). Στην συνέχεια συλλέχθηκαν πληροφορίες για την κάθε κατηγορία και το κάθε Frameworks ξεχωριστά μέσα από πηγές στο διαδίκτυο. Κύριες πηγές ήταν άρθρα σε γνωστά προγραμματιστικά forums αλλά και απόψεις προγραμματιστών ανά τον κόσμο για τις εμπειρίες του με το εκάστοτε Framework. Ακόμη σημαντικό ρόλο στην έρευνα έπαιξαν και οι πληροφορίες που η επίσημες ιστοσελίδες των Frameworks παρείχαν με το Documentation τους να δίνει χρήσιμα στοιχεία γι αυτά. Στην συνέχεια πραγματοποιήθηκαν συγκρίσεις βασικών χαρακτηριστικών των Frameworks με συγκεκριμένα κριτήρια. Στο επόμενο στάδιο της έρευνας ολοκληρώθηκε ένας πίνακας μεγάλος πίνακας με τις βέλτιστες περιπτώσεις χρήσης του κάθε Frameworks βασισμένες στις εμπειρίες των προγραμματιστών και στα στοιχεία που συλλέχθηκαν παραπάνω. Έπειτα δημιουργήθηκαν τα ερωτήματα προς τον χρήστη, από τα οποία θα εξαχθούν οι προτάσεις για την τελική επιλογή του κατάλληλου Framework ανά περίπτωση. Ακολούθησε ο σχεδιασμός των δένδρων αποφάσεων για κάθε κατηγορία Framework και τέλος η υλοποίηση της Web εφαρμογής η οποία μέσω ερωτήσεων προτείνει στον χρήστη το κατάλληλο Framework προς χρήση ανάλογα με τις απαιτήσεις του.

Με το τέλος της έρευνας, τα συμπεράσματα που προκύπτουν επιβεβαιώνουν σε μεγάλο βαθμό τις αρχικές υποθέσεις. Με μεγάλη ασφάλεια πλέον μπορεί να επιβεβαιωθεί πως δεν υπάρχει η έννοια του 'καλύτερου' Framework σαν όρος σε μια τυφλή σύγκριση μεταξύ τους. Όλα εξαρτώνται από τις απαιτήσεις του Project και του χρήστη. Ένα από τα βασικά αποτελέσματα όμως, είναι πως πραγματικά υπάρχουν τα κατάλληλα Framework για κάθε περίπτωση χρήσης, κάτι που είναι πολύ γενικό σαν συμπέρασμα. Πιο συγκεκριμένα, μπορεί να ειπωθεί πως ανάλογα με τις απαιτήσεις και

τις προϋποθέσεις του κάθε Project κάποια Frameworks έχουν συγκεκριμένα θετικά και άλλα αρνητικά χαρακτηριστικά. Με την ολοκλήρωση και της διαδικτυακής εφαρμογής που συνοδεύει την συγκεκριμένη έρευνα, οι ενδιαφερόμενοι χρήστες θα μπορούν να την συμβουλευτούν για να τους προτείνει συγκεκριμένες τεχνολογίες οι οποίες μέσα από την έρευνα, προκύπτει πως ταιριάζουν στις απαιτήσεις τους.

Σημαντικό αποτέλεσμα της έρευνας, είναι πως μέσα από το σύνολο των πηγών που αναλύθηκαν στο σύνολο του διαδικτύου προκύπτει μια 'εμμονή' των χρηστών στην χρήση JavaScript Frameworks ακόμη και όταν αυτά δεν είναι απαραίτητα. Η 'μόδα' πιθανώς που κυκλοφορεί και οι καθημερινές αναφορές που γίνονται γι' αυτά σε Forum στο διαδίκτυο, έχουν επηρεάσει τους χρήστες. Προκύπτει λοιπόν, πως τα περισσότερα Project των χρηστών είναι πολλές φορές ευκολότερο να υλοποιηθούν χωρίς την χρήση κάποιου Framework, με χρήση απλού JavaScript κώδικα.

Από την άλλη πλευρά, δεν εξήχθησαν απόλυτα συμπεράσματα για την εξ' ολοκλήρου ταύτιση κάποιας τεχνολογίας με κάποιο συγκεκριμένο στυλ Project που θέλει ο χρήστης να αναπτύξει.

Επίσης, δεν μπορεί να πραγματοποιηθεί κάποια ασφαλής κατηγοριοποίηση των Frameworks. Ωστόσο ανάλογα με τα χαρακτηριστικά του καθενός και την χρήση τους από τους προγραμματιστές, μπορούμε να τα ομαδοποιήσουμε, με στόχο την κατάλληλη χρήση τους σε κάθε περίπτωση.

5.2 Όρια και περιορισμοί της έρευνας

Τα συμπεράσματα που προαναφέρθηκαν δεν μπορούν να είναι απόλυτα. Παρόλο που έγιναν πολλές προσπάθειες για την όσο πιο πιστή, έγκυρη και πολύπλευρη αποτύπωση δεδομένων, υπάρχουν φυσικά πολλοί παράγοντες που θέτουν όρια και συγκεκριμένους περιορισμούς στην έρευνα που πραγματοποιήθηκε. Πολύ βασικό μέρος αυτών είναι αρχικά ο αριθμός των Frameworks προς σύγκριση. Έχοντας επιλέξει ένα δείγμα με 11 Front-End, 3 Data-Layer, και 6 Back-End Frameworks, γνωρίζοντας την ύπαρξη των χιλιάδων υπόλοιπων Frameworks καθώς και αυτών που κάθε μέρα δημιουργούνται και εισάγονται στην αγορά, γίνεται αντιληπτό πως μόνο ένα μικρό μέρος αυτών εκπροσωπείται στην συγκεκριμένη έρευνα. Επίσης τα κριτήρια σύγκρισης που επιλέχθηκαν, είναι από την μία αρκετά πλήρη, αλλά από την άλλη ο κάθε χρήστης μπορεί να έχει πολύ συγκεκριμένες απαιτήσεις από ένα Framework οι οποίες είναι

δύσκολο πολλές φορές να συμπεριληφθούν εξ' ολοκλήρου σε μια τέτοια έρευνα. Τέλος το γεγονός πως οι απόψεις και οι εμπειρίες προγραμματιστών δημιούργησαν τα δεδομένα της έρευνας, κάνει την τελική επιλογή και πρόταση του κάθε Framework ακόμη πιο υποκειμενική.

Συνολικά η συγκεκριμένη έρευνα δίνει χρήσιμες λύσεις στο βασικό πρόβλημα επιλογής των κατάλληλων Framework ανά περίπτωση. Ωστόσο σίγουρα υπάρχουν και άλλες λύσεις που είτε δεν αναφέρονται, είτε δεν προτείνονται με βάση τα κριτήρια σύγκρισης που έχουν επιλεχτεί.

5.3 Μελλοντικές Επεκτάσεις

Στην συγκεκριμένη ενότητα θα αναφερθούν μελλοντικές κατευθύνσεις για έρευνα όσων αφορά συνολικά τα JavaScript Frameworks. Με την συγκεκριμένη μελέτη θεραπεύτηκε ένα κομμάτι αυτής της θεματικής ενότητας που είχε να κάνει με την αποτελεσματική επιλογή ενός JavaScript Framework προς χρήση. Ωστόσο η θεματική ενότητα περιέχει και άλλα προβλήματα τα οποία μελλοντικά από κάποια νέα έρευνα μπορούν να λυθούν ή να προταθούν λύσεις γι αυτά.

Μεγάλο ζήτημα είναι το εάν ο χρήστης θα πρέπει να χρησιμοποιήσει κάποιο JavaScript Framework ή όχι σε κάποιο Project που θέλει να δημιουργήσει. Το συγκεκριμένο ερώτημα απαντάται εν μέρη με την συγκεκριμένη έρευνα ωστόσο για πιο σαφή αποτελέσματα θα μπορούσε να γίνει μια ξεχωριστή έρευνα πάνω σε αυτό, η οποία θα διαλεύκανε, βασισμένη και αυτή σε συγκεκριμένα κριτήρια και περιπτώσεις, την απόφαση αυτή.

Ακόμη, μια έρευνα στο επίπεδο μεσαίων/μεγάλων επιχειρήσεων στην Ελλάδα και στο εξωτερικό, για τις JavaScript τεχνολογίες που αυτές χρησιμοποιούν σε πραγματικό χρόνο με ορατά αποτελέσματα, θα έδινε μια ολοκληρωμένη εικόνα για την κατάσταση και τις περιπτώσεις χρήσης της κάθε τεχνολογίας.

Εξίσου μεγάλο ενδιαφέρον θα έχει ο ετήσιος έλεγχος και παρατήρηση των αποτελεσμάτων που το stateofjs.com εξάγει ετησίως. Τα συγκεκριμένα αποτελέσματα στα οποία βασίστηκε η έρευνα αφορούσαν το 2018. Έτσι, τα επόμενα αποτελέσματα για τα έτη 2019 και 2020 θα έχουν ιδιαίτερο ενδιαφέρον για να αποφασίσουμε και να μελετήσουμε την πορεία της κάθε τεχνολογίας μέσα στην προγραμματιστική κοινότητα.

Επίσης, σημαντική να αναφερθεί είναι, μια μελλοντική επέκταση, βελτίωση και ενημέρωση της Web εφαρμογής που υλοποιήθηκε στα πλαίσια της συγκεκριμένης έρευνας. Υπάρχουν μέρη που μπορούν να εμπλουτιστούν με συμπληρωματικό περιεχόμενο για την ολοκληρωμένη πρόταση τεχνολογιών από την εφαρμογή προς τον χρήστη. Για παράδειγμα, μαζί με την προτεινόμενη τεχνολογία που προκύπτει από το δένδρο αποφάσεων, μπορούν να προτείνονται συγκεκριμένα resources, σύνδεσμοι, παραδείγματα, ακόμη και τα στοιχεία και πληροφορίες που εμπεριέχονται μέσα στην έρευνα που πραγματοποιήθηκε. Επίσης μπορούν να εξάγονται στατιστικά, συνολικά για τις τεχνολογίες που έχουν προταθεί μέσα από την εφαρμογή. Για παράδειγμα, ένα ποσοστό των προγραμματιστών που απάντησαν στις ερωτήσεις τους προτάθηκε μια συγκεκριμένη τεχνολογία κτλ. Ακόμα όσο νέες τεχνολογίες εμφανίζονται και εδραιώνονται στο προσκήνιο η εφαρμογή να μπορεί να είναι πάντα ενημερωμένη και να μπορεί να συμβουλέψει και να ενημερώσει τον χρήστη γι αυτές. Θα μπορούσε πιο συγκεκριμένα να υπάρχει ένα περιβάλλον διαχείρισης των Web στοιχείων για ενημέρωσή τους από τον χρήστη, προσθήκη νέων τεχνολογιών, τροποποίηση παλιών κτλ, με αποτέλεσμα η εύκολη τροποποίηση του περιεχομένου. Τέλος, σημαντικό βήμα στην βελτίωση της εφαρμογής, θα ήταν η δημιουργία ενός μηχανισμού ο οποίος θα ενημέρωνε τον χρήστη σε πραγματικό χρόνο κατά την στιγμή που αυτός θα απαντάει στις ερωτήσεις που του γίνονται. Για παράδειγμα, έστω η απάντησή του χρήστη σε μια ερώτηση καθόρισε μονοσήμαντα σχεδόν την επιλογή συγκεκριμένης τεχνολογίας ή απέκλισε ένα μεγάλο τμήμα άλλων τεχνολογιών. Σε αυτές τις περιπτώσεις θα μπορούσαν να εμφανίζονται οι λόγοι που αυτό συνέβη, με στόχο την μεγαλύτερη τεκμηρίωση των προτάσεων της εφαρμογής.

Βιβλιογραφία

altexsoft.com, 2017. Node.js Frameworks: Express.js, Meteor.js, Sails.js and more | AltexSoft [WWW Document]. URL https://www.altexsoft.com/blog/engineering/node-js-frameworks-comparison-for-your-back-end-solution-express-js-meteor-js-sails-js-and-more/ (accessed 6.15.19).

Angular, 2019a. Angular - Getting Started with Angular: Your First App [WWW Document]. URL https://angular.io/start (accessed 4.6.19).

Angular, 2019b. Angular - Architecture overview [WWW Document]. URL https://angular.io/guide/architecture (accessed 4.6.19).

AngularJS, 2016. Angular, version 2: proprioception-reinforcement. URL http://blog.angularjs.org/2016/09/angular2-final.html (accessed 4.5.19).

Apollo, 2019. Welcome [WWW Document]. Apollo Docs. URL https://www.apollographql.com/docs/ (accessed 5.4.19).

ApolloGraphQL, 2019. Developer tools [WWW Document]. Apollo Docs. URL https://www.apollographql.com/docs/react/features/developer-tooling/ (accessed 6.11.19).

Appsee, 2019. Getting Started With Cross-Platform App Development In 2019 [WWW Document]. Hacker Noon. URL https://hackernoon.com/getting-started-with-cross-platform-app-development-in-2019-dd2bf7f6161b (accessed 5.28.19).

Aurelia, 2019. What is Aurelia? | Aurelia [WWW Document]. URL https://aurelia.io/docs/overview/what-is-aurelia#what-is-aurelia (accessed 4.20.19).

BackboneJS, 2019. Backbone.js [WWW Document]. URL https://backbonejs.org/ (accessed 4.26.19).

BALKANGraph, 2019. OrgChart JS | BALKANGraph [WWW Document]. URL https://balkangraph.com/ (accessed 10.10.19).

Bouchefra, A., 2018. An Introduction to Sails.js. SitePoint. URL https://www.sitepoint.com/an-introduction-to-sails-js/ (accessed 5.18.19).

Bucaran, J., 2019. JavaScript framework for building web user interfaces: jorgebucaran/hyperapp.

Budhani, B., 2017. Getting started with PreactJS — A Step By Step Guide [WWW Document]. Kiprosh Engineering. URL https://blog.kiprosh.com/getting-started-with-preactjs-a-step-by-step-guide-f3197f871753/ (accessed 4.9.19).

Burkholder, B., 2019. JavaScript SEO: Server Side Rendering vs Client Side Rendering [WWW Document]. Medium. URL https://medium.com/@benjburkholder/javascript-seo-server-side-rendering-vs-client-side-rendering-bc06b8ca2383 (accessed 6.5.19).

Chandran, S., 2017. MobX vs Redux with React: A noob's comparison and questions [WWW Document]. codeburst. URL https://codeburst.io/mobx-vs-redux-with-react-a-noobs-comparison-and-questions-382ba340be09 (accessed 6.25.19).

Chinnathambi, K., n.d. Creating Animations in JavaScript [WWW Document]. kirupa.com. URL https://www.kirupa.com/html5/animating_in_code_using_javascript.htm (accessed 6.4.19).

conceptanext, 2019. The Importance of Scalability In Software Design [WWW Document]. Concepta. URL https://conceptainc.com/blog/importance-of-scalability-in-software-design/ (accessed 6.13.19).
Conery, R., 2014. Ember.js Fundamentals [WWW Document]. URL https://www.pluralsight.com/courses/emberjs-fundamentals (accessed 4.12.19).
Deutsch, D., 2019. Starting with Authentication (A tutorial with Node.js and MongoDB) [WWW Document]. Medium. URL https://medium.com/createdd-notes/starting-with-authentication-a-tutorial-with-node-js-and-mongodb-25d524ca0359 (accessed 6.20.19).
Educba, 2018. Mobx vs Redux - Top 8 Useful Differences You Should Know. EDUCBA. URL https://www.educba.com/mobx-vs-redux/ (accessed 4.30.19).
EggHead, 2019. Short, instructional screencast video tutorials for web developers on @eggheadio [WWW Document]. URL https://egghead.io/ (accessed 6.8.19).
Elliott, E., 2019. Master the JavaScript Interview: What is a Pure Function? [WWW Document]. Medium. URL https://medium.com/javascript-scene/master-the-javascript-interview-what-is-a-pure-function-d1c076bec976 (accessed 6.15.19).
Elliott, E., 2017. Must See JavaScript Dev Tools That Put Other Dev Tools to Shame [WWW Document]. Medium. URL https://medium.com/javascript-scene/must-see-javascript-dev-tools-that-put-other-dev-tools-to-shame-aca6d3e3d925 (accessed 6.11.19).
Eluwande, Y., 2017. Introduction to Preact — a smaller, faster React alternative [WWW Document]. LogRocket Blog. URL https://blog.logrocket.com/introduction-to-preact-a-smaller-faster-react-alternative-ad5532eb6d79/ (accessed 4.9.19).
Ember, 2019. How To Use The Guides - Getting Started - Ember Guides [WWW Document]. URL https://guides.emberjs.com/release/getting-started/intro/ (accessed 4.11.19).
Esteban, 2017. Routing in Javascript - Esteban [WWW Document]. Medium. URL https://medium.com/@fro_g/routing-in-javascript-d552ff4d2921 (accessed 6.3.19).
Express, 2019. Express - Node.js web application framework [WWW Document]. URL https://expressjs.com/ (accessed 5.6.19).
FeathersJs, 2019. Introduction · FeathersJS [WWW Document]. URL https://docs.feathersjs.com/ (accessed 5.20.19).
GitHub, 2019. Build software better, together [WWW Document]. GitHub. URL https://github.com (accessed 6.10.19).
GraphQL, 2019. GraphQL: A query language for APIs. [WWW Document]. URL http://graphql.org/ (accessed 5.3.19).
Greif, S., 2018. Who Took the State of JavaScript 2018 Survey? [WWW Document]. Developer News. URL https://www.freecodecamp.org/news/who-took-the-state-of-javascript-2018-survey-8b51bca63a0/ (accessed 4.5.19).
Hamedani, M., 2019. Redux Vs. Mobx - What Should I Pick For My Web App? Programming with Mosh. URL https://programmingwithmosh.com/react/redux-vs-mobx-what-should-i-pick-for-my-web-app/ (accessed 6.13.19).
Hannah, J., 2018a. The Ultimate Guide to JavaScript Frameworks [WWW Document]. URL https://jsreport.io/the-ultimate-guide-to-javascript-frameworks/ (accessed 4.2.19).

Hannah, J., 2018b. The Ultimate Guide to JavaScript Frameworks [WWW Document]. JavaScript Report. URL https://jsreport.io/the-ultimate-guide-to-javascript-frameworks/ (accessed 4.2.19).

Hayani, S., 2018a. Nextjs for everyone — with some basic knowledge of React [WWW Document]. freeCodeCamp.org News. URL https://www.freecodecamp.org/news/an-introduction-to-next-js-for-everyone-507d2d90ab54/ (accessed 5.8.19).

Hayani, S., 2018b. Here are the most popular ways to make an HTTP request in JavaScript [WWW Document]. freeCodeCamp.org News. URL https://www.freecodecamp.org/news/here-is-the-most-popular-ways-to-make-an-http-request-in-javascript-954ce8c95aaa/ (accessed 6.3.19).

HowToGraphQL, 2019. Learn GraphQL Fundamentals with Fullstack Tutorial [WWW Document]. URL https://www.howtographql.com/basics/0-introduction (accessed 5.3.19).

Ighodaro, N., 2018. Why use Redux? Reasons with clear examples [WWW Document]. LogRocket Blog. URL https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835/ (accessed 4.30.19).

KoaJS, 2019. Koa - next generation web framework for node.js [WWW Document]. URL https://koajs.com/ (accessed 5.11.19).

Krause, S., 2018. Interactive Results [WWW Document]. URL https://www.stefankrause.net/js-frameworks-benchmark8/table.html (accessed 4.8.19).

LinkedIn, 2019. LinkedIn: Log In or Sign Up [WWW Document]. LinkedIn: Log In or Sign Up. URL https://www.linkedin.com/ (accessed 6.11.19).

Manjunath, M., 2019. Server-Side vs. Client-Side Rendering - An In-Depth Overview [WWW Document]. URL https://storylens.com/@manjunath/server-side-vs-client-side-rendering---an-in-depth-overview (accessed 6.5.19).

MDN web docs, 2019. Web Components [WWW Document]. MDN Web Docs. URL https://developer.mozilla.org/en-US/docs/Web/Web_Components (accessed 6.25.19).

MeteorJS, 2019. Meteor API Docs | Meteor API Docs [WWW Document]. URL http://docs.meteor.com (accessed 5.15.19).

Meyghani, A.J., 2019. JavaScript Frameworks, Performance Comparison - AJ Meyghani [WWW Document]. Medium. URL https://medium.com/@ajmeyghani/javascript-frameworks-performance-comparison-c566d19ab65b (accessed 4.6.19).

MithrilJS, 2019. Introduction - Mithril.js [WWW Document]. URL https://mithril.js.org/ (accessed 4.28.19).

MobX, 2019. MobX: Ten minute introduction to MobX and React [WWW Document]. URL https://mobx.js.org/getting-started.html (accessed 5.6.19).

NativeScript, 2019. Native mobile apps with Angular, Vue.js, TypeScript, JavaScript - NativeScript [WWW Document]. NativeScript.org. URL https://www.nativescript.org/ (accessed 5.30.19).

NextJS, 2019. Learn | Next.js [WWW Document]. URL https://nextjs.org/learn (accessed 5.8.19).

NodeJS, 2019. About [WWW Document]. Node.js. URL https://nodejs.org/en/about/ (accessed 5.3.19).

NPMTrends, 2019. npm trends: Compare NPM package downloads [WWW Document]. URL https://www.npmtrends.com/ (accessed 6.7.19).

Nwamba, C., 2019. Build Light-Weight REST and Realtime Apps with FeathersJS [WWW Document]. Scotch. URL http://scotch.io/tutorials/build-light-weight-rest-and-realtime-apps-with-feathersjs (accessed 5.20.19).

Oki, C., 2019. State management patterns in JavaScript: Sharing data across components [WWW Document]. LogRocket Blog. URL https://blog.logrocket.com/state-management-pattern-in-javascript-sharing-data-across-components-f4420581f535/ (accessed 6.3.19).

Oliff, L., 2018. Build Your First App with Polymer and Web Components [WWW Document]. Auth0 - Blog. URL https://auth0.com/blog/build-your-first-app-with-polymer-and-web-components/ (accessed 4.13.19).

Parody, L., 2019. Choosing the right Node.js Framework: Express, Koa, or Hapi? [WWW Document]. The NodeSource Blog - Node.js Tutorials, Guides, and Updates. URL http://nodesource.com/blog/Express-Koa-Hapi/ (accessed 5.6.19).

Polymer, 2019. Polymer library - Polymer Project [WWW Document]. URL https://polymer-library.polymer-project.org/3.0/docs/devguide/feature-overview (accessed 4.13.19).

PreactJS, 2019. Differences to React | Preact: Fast 3kb React alternative with the same ES6 API. Components & Virtual DOM. [WWW Document]. URL https://preactjs.com/guide/differences-to-react (accessed 4.9.19).

Ravichandran, A., 2019. A definitive guide to Redux vs. MobX [WWW Document]. LogRocket Blog. URL https://blog.logrocket.com/redux-vs-mobx/ (accessed 6.28.19).

React, 2019. React – A JavaScript library for building user interfaces [WWW Document]. URL https://reactjs.org/ (accessed 4.2.19).

RealWordApp, 2019. “The mother of all demo apps” — Exemplary fullstack Medium.com clone powered by React, Angular, Node, Django, and many more : gothinkster/realworld. Thinkster.

Redux, 2019. Getting Started with Redux · Redux [WWW Document]. URL https://redux.js.org/ (accessed 4.30.19).

Restuta, 2019. Sizes of JS frameworks, just minified + minified and gzipped, (React, Angular 2, Vue, Ember) [WWW Document]. Gist. URL https://gist.github.com/Restuta/cda69e50a853aa64912d (accessed 4.12.19).

Rusendić, S.K., 2018. Do you really need WebSockets? [WWW Document]. Not again. URL https://blog.stanko.io/do-you-really-need-websockets-343aed40aa9b (accessed 6.22.19).

SailsJS, 2019. Sails.js | Realtime MVC Framework for Node.js [WWW Document]. URL https://sailsjs.com/ (accessed 5.18.19).

Schae, J., 2019. A RealWorld Comparison of Front-End Frameworks with Benchmarks (2019 update) [WWW Document]. freeCodeCamp.org News. URL https://www.freecodecamp.org/news/a-realworld-comparison-of-front-end-frameworks-with-benchmarks-2019-update-4be0d3c78075/ (accessed 4.17.19).

Siddharth, 2011. Getting Started with Backbone.js [WWW Document]. Code Envato Tuts+. URL https://code.tutsplus.com/tutorials/getting-started-with-backbonejs--net-19751 (accessed 4.25.19).

StackOverflow, 2019. Stack Overflow - Where Developers Learn, Share, & Build Careers [WWW Document]. Stack Overflow. URL https://stackoverflow.com/ (accessed 6.10.19).
StateofJs, 2018a. The State of JavaScript 2018: Home [WWW Document]. URL https://2018.stateofjs.com/ (accessed 4.12.19).
StateofJs, 2018b. The State of JavaScript 2018: Front-end Frameworks - React [WWW Document]. URL https://2018.stateofjs.com/front-end-frameworks/react/ (accessed 4.5.19).
StateofJs, 2018c. The State of JavaScript 2018: Front-end Frameworks - Angular [WWW Document]. URL https://2018.stateofjs.com/front-end-frameworks/angular/ (accessed 4.6.19).
StateofJs, 2018d. The State of JavaScript 2018: Front-end Frameworks - Vue.js [WWW Document]. URL https://2018.stateofjs.com/front-end-frameworks/vuejs/ (accessed 4.7.19).
StateofJs, 2018e. The State of JavaScript 2018: Front-end Frameworks - Preact [WWW Document]. URL https://2018.stateofjs.com/front-end-frameworks/preact/ (accessed 4.9.19).
StateofJs, 2018f. The State of JavaScript 2018: Front-end Frameworks - Ember [WWW Document]. URL https://2018.stateofjs.com/front-end-frameworks/ember/ (accessed 4.12.19).
StateofJs, 2018g. The State of JavaScript 2018: Front-end Frameworks - Polymer [WWW Document]. URL https://2018.stateofjs.com/front-end-frameworks/polymer/ (accessed 4.13.19).
StateofJs, 2018h. The State of JavaScript 2018: Data Layer - Overview [WWW Document]. URL https://2018.stateofjs.com/data-layer/overview/ (accessed 4.28.19).
StateofJs, 2018i. The State of JavaScript 2018: Data Layer - Redux [WWW Document]. URL https://2018.stateofjs.com/data-layer/redux/ (accessed 4.30.19).
StateofJs, 2018j. The State of JavaScript 2018: Data Layer - GraphQL [WWW Document]. URL https://2018.stateofjs.com/data-layer/graphql/ (accessed 5.3.19).
StateofJs, 2018k. The State of JavaScript 2018: Data Layer - Apollo [WWW Document]. URL https://2018.stateofjs.com/data-layer/apollo/ (accessed 5.4.19).
StateofJs, 2018l. The State of JavaScript 2018: Data Layer - MobX [WWW Document]. URL https://2018.stateofjs.com/data-layer/mobx/ (accessed 5.5.19).
StateofJs, 2018m. The State of JavaScript 2018: Back-end Frameworks - Express [WWW Document]. URL https://2018.stateofjs.com/back-end-frameworks/express/ (accessed 5.6.19).
StateofJs, 2018n. The State of JavaScript 2018: Back-end Frameworks - Next.js [WWW Document]. URL https://2018.stateofjs.com/back-end-frameworks/nextjs/ (accessed 5.9.19).
StateofJs, 2018o. The State of JavaScript 2018: Back-end Frameworks - Koa [WWW Document]. URL https://2018.stateofjs.com/back-end-frameworks/koa/ (accessed 5.11.19).
StateofJs, 2018p. The State of JavaScript 2018: Back-end Frameworks - Meteor [WWW Document]. URL https://2018.stateofjs.com/back-end-frameworks/meteor/ (accessed 5.15.19).

StateofJs, 2018q. The State of JavaScript 2018: Back-end Frameworks - Sails [WWW Document]. URL https://2018.stateofjs.com/back-end-frameworks/sails/ (accessed 6.25.19).
StateofJs, 2018r. The State of JavaScript 2018: Back-end Frameworks - FeathersJS [WWW Document]. URL https://2018.stateofjs.com/back-end-frameworks/feathers/ (accessed 5.20.19).
Svelte, 2019. Svelte 3: Rethinking reactivity [WWW Document]. URL https://svelte.dev/blog/svelte-3-rethinking-reactivity (accessed 4.17.19).
Tate, D., 2015. JavaScript Templating What is Templating? - Darlene Tate [WWW Document]. Medium. URL https://medium.com/@BuildMySite1/javascript-templating-what-is-templating-7ff49d97db6b (accessed 6.3.19).
Tinning, M., 2016. A Functional Front-End with React [WWW Document]. Scott Logic. URL https://blog.scottlogic.com/2016/04/04/a-functional-front-end-with-react.html (accessed 4.4.19).
TodoMVC, 2019. TodoMVC [WWW Document]. URL http://todomvc.com (accessed 6.7.19).
tutorialspoint.com, 2019a. Node.js Express Framework [WWW Document]. www.tutorialspoint.com. URL https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm (accessed 5.5.19).
tutorialspoint.com, 2019b. TypeScript Overview [WWW Document]. www.tutorialspoint.com. URL https://www.tutorialspoint.com/typescript/typescript_overview.htm (accessed 6.2.19).
tutorialspoint.com, 2019c. Node.js RESTful API [WWW Document]. www.tutorialspoint.com. URL https://www.tutorialspoint.com/nodejs/nodejs_restful_api (accessed 6.20.19).
Udemy, 2019. Online Courses - Learn Anything, On Your Schedule [WWW Document]. Udemy. URL https://www.udemy.com/ (accessed 6.8.19).
Vandivier, J., 2018. Which JavaScript ORM should you be using in 2018? [WWW Document]. Developer News. URL https://www.freecodecamp.org/news/a-comparison-of-the-top-orms-for-2018-19c4feeaa5f/ (accessed 6.22.19).
Vega, J., 2017. Client-side vs. server-side rendering: why it's not all black and white [WWW Document]. freeCodeCamp.org News. URL https://www.freecodecamp.org/news/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d/ (accessed 6.4.19).
Vepsäläinen, J., 2016. Svelte - The magical disappearing UI framework - Interview with Rich Harris [WWW Document]. SurviveJS. URL https://survivejs.com/blog/svelte-interview/ (accessed 4.17.19).
Vue.js, 2019. Comparison with Other Frameworks — Vue.js [WWW Document]. URL https://vuejs.org/v2/guide/comparison.html (accessed 4.7.19).
W3Schools, 2019a. ECMAScript 6 [WWW Document]. URL https://www.w3schools.com/js/js_es6.asp (accessed 6.1.19).
W3Schools, 2019b. JavaScript Debugging [WWW Document]. URL https://www.w3schools.com/js/js_debugging.asp (accessed 6.11.19).
webcomponents.org, 2019. webcomponents.org [WWW Document]. URL https://www.webcomponents.org/ (accessed 6.3.19).

Wieruch, R., 2017. Redux or MobX: An attempt to dissolve the Confusion - RWieruch [WWW Document]. URL https://www.robinwieruch.de/redux-mobx-confusion/ (accessed 6.25.19).

WikiPedia, 2019a. ECMAScript. Wikipedia.

WikiPedia, 2019b. Microsoft TypeScript. Wikipedia.

Wodehouse, C., 2016. Is MeteorJS the Right Node.js Framework For You? [WWW Document]. Hiring | Upwork. URL https://www.upwork.com/hiring/development/meteorjs-node-js-framework/ (accessed 5.15.19).

You, E., 2014. First Week of Launching Vue.js [WWW Document]. Evan You. URL http://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/index.html (accessed 4.7.19).

Zaveri, M., 2018. What is boilerplate and why do we use it? Necessity of coding style guide [WWW Document]. Developer News. URL https://www.freecodecamp.org/news/whats-boilerplate-and-why-do-we-use-it-lets-check-out-the-coding-style-guide-ac2b6c814ee7/ (accessed 6.13.19).