ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΟΝ ΙΣΤΟ

ΣΧΕΔΙΟ ΑΣΦΑΛΕΙΑΣ

ΣΥΓΓΡΑΦΕΙΣ: Δούσης Νικόλαος 3160030

Σωτηροπούλου Δικαία 3160172

ΕΡΓΑΣΙΑ ΕΑΡΙΝΟΥ ΕΞΑΜΗΝΟΥ 2019-2020

1. Πρόλογος

Η επιλογή της καλύτερης δυνατής τεχνολογίας front-end είναι κάτι αρκετά σημαντικό για την υλοποίηση μιας web εφαρμογής. Ο σημερινός κόσμος του διαδικτύου έχει εμπλουτιστεί με πάρα πολλές τεχνολογίες που μπορούν να χρησιμοποιήσουν οι προγραμματιστές ώστε να καλύψουν τις ανάγκες τους. Η εξέλιξη αυτών των τεχνολογιών είναι ραγδαία και ο προγραμματιστής τίθεται στο δίλλημα του τι εργαλείο και ποια αρχιτεκτονική πρέπει να χρησιμοποιήσει. Τέτοια εργαλεία είναι το framework της Angular και η βιβλιοθήκη της React τα οποία είναι ευρέως γνωστά σε όλους τους προγραμματιστές web εφαρμογών. Εκτός από τα εργαλεία ο προγραμματιστής πρέπει να επιλέξει και την κατάλληλη αρχιτεκτονική με την οποία θα υλοποιήσει την web εφαρμογή του. Δυο τέτοιες αρχιτεκτονικές είναι της Single Page Application και της Isomorphic web application. Η παρούσα εργασία έχει ως σκοπό να αναλύσει εκτενώς τα εργαλεία και τις αρχιτεκτονικές αυτές και να ξεκαθαρίσει πότε πρέπει να προτιμάτε το καθένα.

2. Μέρος Α' : Συγκριτική μελέτη των εργαλείων Angular και React

Στο πρώτο μέρος της εργασίας γίνεται μια εκτενής παρουσίαση των δύο αυτών εργαλείων, αναλύοντας τα χαρακτηριστικά τους, τους λόγους για τους οποίους οι προγραμματιστές καταλήγουν στην επιλογή αυτών των εργαλείων, αλλά και εκείνους για τους οποίους τα αποφεύγουν. Ακόμη, γίνεται αναφορά στο πότε είναι ενδεικτικότερη η χρήση κάθε εργαλείου, ενώ στο τέλος του μέρους αυτού παρατίθεται ένας συγκριτικός πίνακας με τα χαρακτηριστικά της Angular και της React JS.

2.1. Angular JS

2.1.1. Η ιστορία της Angular JS [1]

Η ιστορία της Angular JS άρχισε το 2009 όταν ο Misko Hevery και ο Adam Abrons δούλευαν σε ένα project την δημιουργία ενός web development εργαλείου το οποίο θα προσέφερε μια υπηρεσία αποθήκευσης δεδομένων σε ένα online JSON storage και μια client side βιβλιοθήκη, η οποία μπορούσε να χρησιμοποιηθεί για την δημιουργία web εφαρμογών οι οποίες θα επικοινωνούσαν με αυτό το storage. Παράλληλα με την ανάπτυξη αυτού του Project ο Hevery που εργαζόταν στην Google ήταν υπεύθυνος για την δημιουργία ενός feedback project μαζί με άλλους 2 προγραμματιστές. Μαζί καταφέραν να γράψουνε περισσότερες από 17,000 σειρές κώδικα μέσα σε 6 μήνες. Ο κώδικάς τους όμως ήταν πολύ μεγάλος για να πραγματοποιηθούν έλεγχοι. Λόγω αυτού, ο Hevery ρώτησε τον manager του εάν θα μπορούσε να γράψει εκ νέου την εφαρμογή, μόνο που αυτήν τη φορά θα χρησιμοποιούσε την βιβλιοθήκη που είχε δημιουργήσει ο ίδιος. Πράγματι, μετά από 3 εβδομάδες κατάφερε να ξαναγράψει όλη την εφαρμογή μειώνοντας σημαντικά τον κώδικα στις 1500 γραμμές. Η Google βλέποντας τις δυνατότητες αυτού του εργαλείου αποφάσισε να το αναπτύξει περεταίρω δίνοντάς του το όνομα Angular JS.

2.1.2. Τι είναι η Angular [2]

Η Angular JS είναι ένα structural framework για δυναμικές web εφαρμογές. Σου επιτρέπει να χρησιμοποιείς HTML ως την βασική σου γλώσσα και να επεκτείνεις την σύνταξή της ώστε να εκφράζεις καθαρά και με ευκρίνεια τα στοιχεία της εφαρμογής. Οι τεχνικές data binding και το dependency injection της Angular JS μειώνουν σημαντικά τον κώδικα που θα έπρεπε να γράψει ο προγραμματιστής. Όλα αυτά συμβαίνουν μέσα στον browser, πράγμα το οποίο την κάνει ιδανική σε συνδυασμό με τεχνολογίες βάσης δεδομένων.

Ενώ με την χρήση απλής HTML θα χρειαζόμασταν την βοήθεια βιβλιοθηκών ή framework, με την χρήση της Angular Js μπορούμε και το αποφεύγουμε. Ειδικότερα, η Angular JS μαθαίνει στον browser μια νέα σύνταξη μέσω μιας κατασκευής που ονομάζουμε directives. Μερικά παραδείγματα είναι: το data binding, δομές ελέγχου του DOM για επανάληψη, εμφάνιση και απόκρυψη των μερών του DOM, υποστήριξη για form και form validation, προσθήκη νέων λειτουργιών σε DOM elements, όπως DOM event handling, ομαδοποίηση της της HTML σε επαναχρησιμοποιήσιμα κομμάτια κ.α.

Επιπλέον, δημιουργήθηκε γύρω από την πεποίθηση ότι ο declarative code είναι καλύτερος από τον imperative όσον αφορά τη δημιουργία user interface και την επικοινωνία μεταξύ software components, ενώ ο imperative code είναι καταλληλότερος για την έκφραση επιχειρησιακής λογικής.

Η Angular JS αποδεσμεύει τον προγραμματιστή από συνήθεις δυσκολίες όπως:

- Τα registering callbacks που γεμίζουν τον κώδικα κάνοντάς τον δυσνόητο. Με την αφαίρεση των callbacks μειώνεται σημαντικά ο κώδικας σε javascript που πρέπει να γραφτεί και κάνει τον κώδικα πιο κατανοητό και εύκολο στην ανάγνωση.
- Η επεξεργασία HTML DOM είναι κάτι που ο προγραμματιστής δεν μπορεί να αποφύγει στις AJAX εφαρμογές, αλλά είναι πολύπλοκη και επιρρεπής σε λάθη. Περιγράφοντας πως το UI πρέπει να αλλάξει σύμφωνα με τις αλλαγές στην εφαρμογή, ο προγραμματιστής δεν χρειάζεται να επεξεργαστεί low-level DOM tasks. Οι περισσότερες εφαρμογές που έχουν γραφτεί με Angular JS δεν χρειάζεται να επεξεργαστούν DOM, ωστόσο δίνεται αυτή η δυνατότητα στον προγραμματιστή.
- Οι λειτουργίες CRUD αναπληρώνουν το μεγαλύτερο μέρος των εργασιών των AJAX εφαρμογών. Η διαδικασία απόκτησης δεδομένων από τον χρήστη μέσω HTML form, η επεξεργασία τους και η αποθήκευση τους στον server, καθώς και η εμφάνιση τυχόν λαθών δημιουργεί μια αρκετά περίπλοκη ροή στον κώδικα. Με την χρήση της Angular JS αυτό περιορίζεται σε σημαντικό βαθμό, δίνοντας έμφαση στην λειτουργία της εφαρμογής και όχι στις λεπτομέρειες υλοποίησης.
- Τυπικά ο προγραμματιστής θα χρειαζόταν να γράψει πολύ κώδικα για να αρχίσει μια απλή εφαρμογή AJAX. Με την Angular JS μπορεί να ξεκινήσει την εφαρμογή του εύκολα χρησιμοποιώντας services τα οποία ενσωματώνονται αυτόματα. Αυτό επιτρέπει στον προγραμματιστή να ξεκινήσει να προγραμματίζει πιο γρήγορα. Επιπλέον, του δίνει πλήρη έλεγχο στην αρχικοποίηση διεργασιών σε αυτόματους ελέγχους.

2.1.3. Πότε χρησιμοποιείται [3]

Ακόμη, απλοποιεί την ανάπτυξη μίας εφαρμογής προσφέροντας μεγαλύτερο βαθμό αφαιρετικότητας στον προγραμματιστή. Αυτό, ωστόσο, έρχεται σε κόστος ευελιξίας. Ειδικότερα, δεν είναι όλες οι εφαρμογές συμβατές με την Angular JS καθώς αρχικά ήταν σχεδιασμένη για CRUD εφαρμογές. Παιχνίδια και GUI editors είναι παραδείγματα εφαρμογών που έχουν έντονη και περίπλοκη διαχείριση του DOM . Αυτού του είδους οι εφαρμογές είναι διαφορετικές από αυτές με CRUD λογική, και ως αποτέλεσμα δεν συνίσταται η χρήση της Angular JS. Αντιθέτως, οι εφαρμογές στις οποίες η Angular συμβάλει περισσότερο είναι εφαρμογές τύπου video streaming, αξιολόγησης αντικειμένων και υπηρεσιών από χρήστες, ταξιδιωτικές εφαρμογές, εφαρμογές πρόβλεψης καιρού, e-shop και εφαρμογές κοινωνικής δικτύωσης.

2.1.4. Κριτήρια που ενισχύουν την επιλογή [4]

Υπάρχουν συγκεκριμένες δυνατότητες της Angular για τις οποίες πολλοί προγραμματιστές την προτιμούν. Μερικές από αυτές τις δυνατότητες είναι:

a. Σωστό MVC

Τα περισσότερα framework εφαρμόζουν MVC με το να ζητούν από τον προγραμματιστή να χωρίσει την εφαρμογή του σε MVC components και μετά να γράψει τον κώδικα για την επικοινωνία μεταξύ τους. Αντιθέτως η Angular JS αφού ζητήσει από τον προγραμματιστή τον διαχωρισμό της εφαρμογής σε MVC components κάνει τις υπόλοιπες λειτουργίες αυτόματα. Με άλλα λόγια, διαχειρίζεται τα components και βοηθά στην επικοινωνία μεταξύ τους.

b. Δηλωτικό UI

Η Angular JS χρησιμοποιεί την HTML για να δηλώσει το UI της εφαρμογής. Η HTML είναι λιγότερο περίπλοκη στην δημιουργία interface από την JavaScript. Ακόμη, η HTML είναι λιγότερο εύθραυστη στην οργάνωση του interface συγκριτικά με την JavaScript όπου είναι πιο πιθανό να εμφανίσει προβλήματα. Επίσης, η Angular JS χρησιμοποιεί την HTML και στο κομμάτι της εκτέλεσης της εφαρμογής, καθώς ειδικά attributes της HTML καθορίζουν ποιος controller θα χρησιμοποιηθεί για κάθε στοιχείο και το περιεχόμενο που θα φορτωθεί χωρίς να επηρεάζεται από το πώς θα φορτωθεί. Αυτή η δηλωτική προσέγγιση απλουστεύει σημαντικά την ανάπτυξη εφαρμογών αφού ο προγραμματιστής δεν χρειάζεται να χάσει χρόνο για να βρει τον τρόπο διάταξης του προγράμματος, απλά δηλώνει τι χρειάζεται και η Angular τα υλοποιεί.

c. Συνηθισμένα μοντέλα δεδομένων

Τα μοντέλα δεδομένων στην Angular JS είναι τα κλασικά JavaScript αντικείμενα και δεν χρειάζονται getters και setters. Ο προγραμματιστής μπορεί να προσθέσει και να αλλάξει ιδιότητες απευθείας και να προσπελάσει πίνακες και αντικείμενα όποτε θέλει κάνοντας τον κώδικα πιο ευανάγνωστο. Όλες οι ιδιότητες που βρίσκονται στο scope του αντικειμένου συνδέονται αυτόματα στο view από την Angular. Αυτό σημαίνει πως η Angular βλέπει στο background για αλλαγές σε αυτές τις ιδιότητες και ενημερώνει αυτόματα το view. Το scope δεν έχει εξαρχής δεδομένα και βασίζεται στον controller να του δώσει δεδομένα, ανάλογα με το τι χρειάζεται.

d. Directives

Τα directives είναι ο τρόπος την Angular JS να προσδίδει περισσότερη λειτουργικότητα στην HTML. Ο προγραμματιστής μπορεί να δημιουργήσει οποιοδήποτε στοιχείο θέλει χωρίς να χρειάζεται επεξεργασία του DOM. Το μόνο που χρειάζεται είναι να θέσει ιδιότητες στο στοιχείο. Τα directives το επιτυγχάνουν αυτό με το να του δίνουν την δυνατότητα να μπορεί να δημιουργήσει δικά του HTML στοιχεία. Με το να βάλει όλο το DOM μέσα σε directives, μπορεί να το διαχωρίσει εκτός της MVC εφαρμογής. Αυτό επιτρέπει στην MVC εφαρμογή να επικεντρώνεται μόνο στην ενημέρωση του view με όλα τα νέα δεδομένα. Το πώς θα συμπεριφερθεί το view εξαρτάται από τα directives.

e. Ευελιξία με φίλτρα

Προτού τα δεδομένα φτάσουν στο view περνούν από φίλτρα τα οποία μπορεί να έχουν λειτουργίες όπως: αλλαγή δεκαδικών ψηφίων σε αριθμούς, αντιστροφή της διάταξης ενός πίνακα, φιλτράρισμα ενός πίνακα μέσω κάποιας παραμέτρου ή υλοποίηση σελιδοποίησης. Τα φίλτρα είναι σχεδιασμένα ως ξεχωριστές συναρτήσεις που είναι ανεξάρτητες από την κύρια εφαρμογή παρόμοια με τα directives, αν και επικεντρώνονται κυρίως στην επεξεργασία δεδομένων.

f. Λιγότερος κώδικας

Με βάση τα παραπάνω είναι εμφανές ότι με την χρήση της Angular JS ο προγραμματιστής θα χρειαστεί να γράψει λιγότερο κώδικα για την εφαρμογή του. Ειδικότερα, δεν είναι αναγκαίο να δημιουργήσει τα δικά του MVC pipeline, το UI είναι κατασκευασμένο με την χρήση HTML, τα μοντέλα δεδομένων είναι απλά αφού δεν χρειάζονται setters και getters, με την χρήση του data-binding δεν χρειάζεται να βάλει δεδομένα στο view χειροκίνητα, τα φίλτρα επιτρέπουν την επεξεργασία δεδομένων στο view χωρίς την αλλαγή controller.

g. Service Providers

Τα controller στην Angular είναι απλές συναρτήσεις που έχουν μόνο μια λειτουργία, την αλλαγή του scope. Σε αντίθεση με άλλα frameworks, τα controller δεν είναι αντικείμενα και δεν κληρονομούν ιδιότητες ή χαρακτηριστικά από πουθενά. Επειδή τα controllers είναι τόσο απλά, η Angular χρησιμοποιεί services για να εκτελέσει πιο σύνθετες λειτουργίες. Τα services δεν ασχολούνται με το MVC της εφαρμογής, αλλά παρέχουν ένα εξωτερικό API για να προβάλουν ότι ο προγραμματιστής θέλει να προβάλει. Τις περισσότερες φορές συγχρονίζεται με ένα server για να διατηρεί δεδομένα όταν η εφαρμογή είναι εκτός λειτουργίας και χρησιμοποιεί μεθόδους για να καταχωρεί ή να παίρνει δεδομένα από τον server. Εναλλακτικά, μπορεί να χρησιμοποιηθεί στην κατανομή των ίδιων πόρων σε πολλαπλά controllers.
Τα services είναι σχεδιασμένα να είναι ξεχωριστά αντικείμενα από την εφαρμογή και επιτρέπουν στον controller να παραμείνει αφοσιωμένος στο view και scope που του έχει ανατεθεί. Η χρήση services δεν είναι απαραίτητη και ο controller μπορεί να χρησιμοποιηθεί για απλές extra λειτουργίες που δεν τον κάνουν περίπλοκο.

h. Επικοινωνία ανάλογα με το πλαίσιο

Ενώ τα περισσότερα PubSub implementations στο διαδίκτυο δεν λαμβάνουν υπόψιν το πλαίσιο στο οποίο έχουν υλοποιηθεί και η επικοινωνία μεταξύ ancestor controller και child controller δεν περιορίζεται μόνο μεταξύ τους, το Angular το επιτυγχάνει. Πιο συγκεκριμένα, το PubSub της Angular επιτρέπει την αποστολή μηνυμάτων μεταξύ ancestor controller και child controller χωρίς να διαβάζονται από στοιχεία του MVC που δεν σχετίζονται με αυτά.
Παρόλα αυτά, το PubSub δεν είναι ο μόνος τρόπος επικοινωνίας μεταξύ controllers. Εάν ο προγραμματιστής έχει πρόθεση να ανανεώνει τα views των controllers όταν αλλάζει μια ιδιότητα, τότε πρέπει να χρησιμοποιήσει data-binding. Τα scope κληρονομούν ιδιότητες από τους parent scope, αυτό σημαίνει ότι εάν μια ιδιότητα υπάρχει στον parent scope και ένα child scope την αλλάζει, τότε όλα τα scopes που κληρονομούν από τον ίδιο πατέρα θα δούνε την ίδια αλλαγή και τα views τους θα ενημερωθούν αυτόματα από την Angular. Αυτός ο τρόπος είναι πολύ πιο αξιόπιστος από το PubSub.

i. Unit testing ready

Η Angular σαν σύνολο είναι συνδεδεμένη μεταξύ της μέσω Dependency Injection και με την χρήση του μπορεί να διαχειρίζεται τους controllers και τα scopes. Επειδή όλα τα controllers βασίζονται στο Dependency Injection για να πάρουν πληροφορίες, τα unit tests της Angular μπορούν να το χρησιμοποιήσουν για να κάνουν unit testing με το να εκχωρούν ψευδή δεδομένα στο controller και να μετρούν την έξοδο και την συμπεριφορά του. Πράγματι, η Angular έχει έναν εικονικό HTTP provider για να εισάγει ψευδείς απαντήσεις από τον server στους controllers. Αυτή η τεχνική είναι καλύτερη από τον παραδοσιακό τρόπο ελέγχου των διαδικτυακών εφαρμογών, γιατί δημιουργεί ξεχωριστές σελίδες ελέγχων που χρησιμοποιούν ένα component και επικοινωνούν μαζί του για να δουν αν λειτουργεί σωστά.

2.1.5. Μειονεκτήματα

Παρόλο που η Angular JS φαίνεται να έχει πληθώρα πλεονεκτημάτων, στην πραγματικότητα έχει και μερικούς περιορισμούς λόγω της JavaScript. Ορισμένα από τα μειονεκτήματα του εργαλείου της Angular JS που αξίζει να αναφερθούν είναι τα εξής:

a. Λιγότερο ασφαλής

Στην Angular δεν υπάρχει δυνατότητα εξουσιοδότησης και ταυτοποίησης από τον server. Αυτό σημαίνει πως όλοι οι χρήστες έχουν τα ίδια δικαιώματα πρόσβασης δεδομένων στον server και δεν υπάρχει κάποια ταυτοποίηση των στοιχείων του.

b. Εξάρτηση από τη JavaScript

Η Angular εξαρτάται εξ ολοκλήρου από την JavaScript. Εάν τα αρχεία των script δεν χρησιμοποιηθούν, οι ιστοσελίδες φαίνονται σαν να έχει γίνει χρήση απλής HTML.

c. Διαρροή μνήμης

Αφού η Angular είναι ένα framework της JavaScript, τότε και εκείνη δεν μπορεί να αποφύγει τη διαρροή μνήμης. Μέρος της μνήμης το οποίο δεν χρησιμοποιείται πλέον από την εφαρμογή δεν καταφέρνει να επιστρέψει στο σημείο που βρίσκεται η ελεύθερη μνήμη. Αυτό έχει ως αποτέλεσμα την ύπαρξη μεγάλων καθυστερήσεων και τον αναπάντεχο τερματισμό της εφαρμογής.

d. Μεγάλος αριθμός τρόπων χρήσης

Αν και αυτό αρχικά φαίνεται να είναι ένα από τα θετικά στοιχεία της Angular, στην πραγματικότητα αποτελεί δυσκολία στους προγραμματιστές καθώς δεν τους καθοδηγεί σε ένα μοναδικό και βέλτιστο τρόπο εκτέλεσης μιας διεργασίας.

e. Δεν υποστηρίζεται παντού

Αν και οι περισσότεροι browsers υποστηρίζουν την χρήση Angular, υπάρχουν και κάποιοι που αποτελούν εξαίρεση. Παράδειγμα αποτελεί ο Internet Explorer 8.0.

f. Καθυστέρηση του UI

Σε περίπτωση που υπάρχει μεγάλος αριθμός επισκεπτών στην ιστοσελίδα, το UI της ιστοσελίδας που χρησιμοποιεί Angular τείνει να υπολειτουργεί. Αυτό σημαίνει πως η Angular δεν μπορεί να χρησιμοποιηθεί για την δημιουργία πολύπλοκων forms. Για παράδειγμα, χρήση big data grids.

g. Σύγκρουση ονομάτων

Με την χρήση Angular ο προγραμματιστής δεν έχει την δυνατότητα να δημιουργήσει πολλά NG-app στην ίδια ιστοσελίδα, αφού αυτό αυξάνει την πιθανότητα να εμφανιστεί σύγκρουση ονομάτων μεταβλητών.

2.2. React JS

2.2.1. Η ιστορία της React

Ένα άλλο εργαλείο της JavaScript που χρησιμοποιείται από πληθώρα προγραμματιστών είναι η βιβλιοθήκη της ReactJS. Τα τελευταία 20 χρόνια το διαδίκτυο έχει αλλάξει ραγδαία. Όλο και περισσότερες λογικές υλοποίησης web εφαρμογών μεταφέρονται από την πλευρά του server στην πλευρά του client. Αρχικά, frameworks, όπως η Angular που αναφέραμε παραπάνω, φέρανε την επανάσταση στην client side ανάπτυξη εφαρμογών, ύστερα όμως εμφανίστηκε η React η οποία έφερε και πάλι στο φως την ανάπτυξη εφαρμογών από την πλευρά του server που είχε χαθεί τόσα χρόνια.

Η ReactJS είναι από τις πιο καινούργιες τεχνολογίες που είναι διαθέσιμες σήμερα για την ανάπτυξη web εφαρμογών. Η βιβλιοθήκη δημιουργήθηκε το 2011 από τον Jordan Walke, ο οποίος ήταν software engineer στο Facebook. Η δημιουργία της React επηρεάστηκε από την XHP το οποίο είναι ένα απλό HTML component framework για τη PHP. Η πρώτη χρήση της React έγινε στην ιστοσελίδα του Facebook το 2011 και έπειτα, χρησιμοποιήθηκε και από την πλατφόρμα του Instagram. Με την πάροδο του χρόνου, η React αναπτυσσόταν όλο και περισσότερο σε σημείο που το Facebook αποφάσισε να το κάνει open source τον Μάιο του 2013. Δύο χρόνια αργότερα, το 2015, ανακοινώθηκε η React Native η οποία θα μπορούσε να χρησιμοποιηθεί για την ευκολότερη ανάπτυξη Android και iOS εφαρμογών.

2.2.2. Τι είναι η React [5]

Η ReactJS δημιουργήθηκε με σκοπό την επίλυση του προβλήματος του επιπέδου του View της αρχιτεκτονικής MVC. Δεν υπάρχει κάποιος συγκεκριμένος τρόπος με τον οποίο μπορεί να χρησιμοποιηθεί, αλλά ο βασικός της ρόλος είναι να δημιουργεί abstract αναπαραστήσεις των view. Διαχωρίζει μέρη του view σε components, τα οποία κατέχουν τη λογική διαχείρισης της εμφάνισης του view, καθώς και το ίδιο το view. Περιέχει δεδομένα τα οποία χρησιμοποιεί ώστε να αλλάξει κατάλληλα την κατάσταση της εφαρμογής.

Προκειμένου να αποφευχθούν η πολυπλοκότητα στην επικοινωνία και η επακόλουθη διεργασία απεικόνισης που απαιτείται, η ίδια η ReactJS αναλαμβάνει την πλήρη απεικόνιση της εφαρμογής. Με αυτό τον τρόπο διατηρεί μια απλή ροή στην εκτέλεση του κώδικα.

Η ιδέα γύρω από τη ReactJS είναι πως η επεξεργασία του DOM είναι μια πολύ κοστοβόρα διεργασία η οποία πρέπει με κάποιο τρόπο να μειωθεί. Η επεξεργασία του DOM από τον ίδιο τον προγραμματιστή έχει ως αποτέλεσμα τη δημιουργία επιπρόσθετου και επαναλαμβανόμενου κώδικα, το οποίο μπορεί να οδηγήσει σε προγραμματιστικά σφάλματα. Η λύση που δίνει η React σε αυτό το πρόβλημα είναι η δημιουργία ενός εικονικού DOM το οποίο μπορεί να επεξεργαστεί ο προγραμματιστής έναντι του πραγματικού. Συγκρίνοντας αυτά τα δύο DOM η React εντοπίζει τις αλλαγές που έχουν γίνει στο εικονικό και τις ενσωματώνει στο πραγματικό με όσο το δυνατόν λιγότερες αλλαγές μπορεί.

Επίσης η React είναι declarative, πράγμα που σημαίνει ότι μόλις αλλάξουν τα δεδομένα, γίνεται αυτόματα ανανέωση στα συγκεκριμένα μέρη της εφαρμογής.

Αυτή η απλή ροή δεδομένων σε συνδυασμό με την απλή μέθοδο προβολής δεδομένων καθιστά τη ReactJS ένα πολύ άμεσο και απλό στη χρήση framework.

Με την React ο προγραμματιστής έχει το πλεονέκτημα να μπορεί να υλοποιήσει isomorphic web application που του δίνει την δυνατότητα να μπορεί να χρησιμοποιήσει τον ίδιο κώδικα τόσο στην πλευρά του server όσο και στην πλευρά του client. Έτσι, όταν μια ιστοσελίδα φορτώνει, η React μπορεί ταυτόχρονα και να φορτώνει τα components και να απεικονίζει το περιεχόμενο στην οθόνη. Με αυτό τον τρόπο η εφαρμογή φορτώνει πολύ γρήγορα και εξοικονομείται κόστος και χρόνος στην ανάπτυξή της.

2.2.3. Κριτήρια που ενισχύουν την επιλογή [6, 7, 8, 9]

Εκτός από το παραπάνω πλεονέκτημα, υπάρχουν και αρκετοί άλλοι λόγοι για τους οποίους η ReactJS είναι αρκετά διαδεδομένη όσον αφορά το web development. Μερικοί από αυτούς είναι οι εξής:

a. Εύκολη στην κατανόηση

Ένα από τα βασικά χαρακτηριστικά που αναζητούν οι προγραμματιστές στα frameworks που θα χρησιμοποιήσουν είναι η απλότητα και η ευκολία στην κατανόηση τους. Η ReactJS είναι το ιδανικότερο framework για όσους είναι αρκετά εξοικειωμένοι με τη JavaScript. Αυτό όμως δε σημαίνει ότι κάποιος δε μπορεί να τη χρησιμοποιήσει χωρίς να γνωρίζει JavaScript.

b. Επαναχρησιμοποίηση Components

Στις περισσότερες web εφαρμογές, αν όχι σε όλες, χρησιμοποιούνται διάφορα components, όπως buttons, dropdown menus, checkboxes κ.α. Κατά τη χρήση της ReactJS, κάθε component αποφασίζει πως θα παρουσιαστεί και έχει τη δική του λογική, ανεξάρτητα από τα υπόλοιπα. Με αυτό τον τρόπο επιτυγχάνεται η επαναχρησιμοποίηση τους κάτι που κάνει πιο εύκολη τη διατήρηση του κώδικα, καθώς και την ανάπτυξη της εφαρμογής.

c. Μοναδική αφαιρετική τεχνική

Ο προγραμματιστής και ο χρήστης της εφαρμογής δε χρειάζεται να γνωρίζουν τις πολύπλοκες τεχνικές που χρησιμοποιεί η ReactJS. Συγκεκριμένα, ο προγραμματιστής χρειάζεται να γνωρίζει μόνο τις βασικές λειτουργίες των εργαλείων που χρησιμοποιούνται. Ακόμη, η ReactJS δεν καθοδηγεί τον προγραμματιστή να υλοποιήσει το πρόγραμμα του σύμφωνα με κάποια από τις γνωστές αρχιτεκτονικές, όπως τις MVC, MVP και MVVM. Αντιθέτως, του δίνεται η δυνατότητα να αποφασίσει μόνος του τον τρόπο σχεδίασης της αρχιτεκτονικής της εφαρμογής.

d. Πληθώρα προγραμματιστικών εργαλείων

Η ReactJS προσφέρει ένα πλούσιο περιβάλλον εργασίας στον προγραμματιστή, δίνοντας του τη δυνατότητα να χρησιμοποιήσει έτοιμα γραφικά, πίνακες και άλλα components. Χάρη σε αυτά, είναι ικανός να υλοποιήσει την εφαρμογή του με περισσότερη ευκολία και ταχύτητα.

e. React Native

Με τη χρήση της ReactJS, ο προγραμματιστής έχει την ευκαιρία να αξιοποιήσει ένα πολύ χρήσιμο framework που παρέχει, τη React Native. Με τη χρήση του, μπορεί να μεταφέρει μεθοδολογίες και αρχιτεκτονικές από μια web σε μια Android ή iOS εφαρμογή. Η ύπαρξη αυτού του framework αποδεικνύει για άλλη μια φορά ότι η ReactJS υποστηρίζει την επαναχρησιμοποίηση κώδικα.

f. Μονόδρομη ροή δεδομένων

Η κύρια ιδέα της Flux αρχιτεκτονικής είναι η δημιουργία ενεργειών που ελέγχονται από έναν κεντρικό dispatcher για την ενημέρωση των stores. Έπειτα, ενημερώνεται το view λαμβάνοντας υπόψιν τις αλλαγές που έγιναν εκεί. Όλα τα δεδομένα που θα εμφανιστούν από τα components αποθηκεύονται σε αυτά χωρίς να δημιουργούνται διπλότυπα σαν μοντέλα MVC. Αυτό βοηθάει στη διατήρηση του συγχρονισμού των δεδομένων καθ' όλη τη διάρκεια εκτέλεσης της εφαρμογής, καθώς και τη μονόδρομη ροή των δεδομένων της.
Ένα αρνητικό που παρατηρείται στη Flux είναι ότι δε διαθέτει βιβλιοθήκη έτοιμη για χρήση. Ωστόσο, υπάρχει μια βιβλιοθήκη που υιοθετεί την αρχιτεκτονική αυτή, και είναι έτοιμη για χρήση, η Redux. Παρέχει μόνο ένα store στο οποίο αποθηκεύονται όλα τα δεδομένα της εφαρμογής, πράγμα που σημαίνει ότι χρειάζεται μόνο ένα αντικείμενο για τη διαχείριση όλης της εφαρμογής.

g. Performance Enhancement

Πολλές φορές όταν γίνεται ενημέρωση στο πραγματικό DOM προκαλείται μεγάλη καθυστέρηση στην εκτέλεση του προγράμματος. Η ReactJS λύνει αυτό το πρόβλημα χρησιμοποιώντας ένα εικονικό DOM, το οποίο βρίσκεται εξ ολοκλήρου στη μνήμη και είναι μια αναπαράσταση του DOM της ιστοσελίδας. Χάρη σε αυτό, όταν γράφουμε ένα component της React δε χρειάζεται να γράψουμε απευθείας πάνω στο DOM. Αντιθέτως, δημιουργούμε εικονικά components, τα οποία η React θα τα μετατρέψει στο DOM, κάνοντας έτσι ομαλότερη και γρηγορότερη την εκτέλεση του προγράμματος.

h. Scope for testing

Η απλότητα στη χρήση της ReactJS μπορεί να παρατηρηθεί ακόμα και στο debugging της, αφού παρέχει στον προγραμματιστή πληθώρα τοπικών εργαλείων (native tools) που είναι εύκολα στη χρήση.

2.2.4. Μειονεκτήματα [10]

Αν και τα θετικά που συναντάμε στη χρήση της ReactJS, υπάρχουν και μερικά αρνητικά, τα οποία μπορεί να αποθαρρύνουν τους προγραμματιστές από το να τη χρησιμοποιήσουν. Μερικά από αυτά είναι τα εξής:

a. Πολύ γρήγορος ρυθμός ανάπτυξης

Παρόλο που αυτό θα ήταν θετικό στοιχείο για πάρα πολλούς προγραμματιστές, η υπερβολικά γρήγορη ανάπτυξη της βιβλιοθήκης αυτής καθιστά δύσκολη την εκμάθηση της. Αυτό συμβαίνει επειδή αρκετά συχνά προστίθενται νέες αλλαγές και νέοι τρόποι υλοποίησης πραγμάτων που οι προγραμματιστές είναι αναγκασμένοι να τη μάθουν εκ νέου για να τη χρησιμοποιήσουν.

b. Ανεπαρκές Documentation

Σε συνδυασμό με το παραπάνω μειονέκτημά της, η τόσο ραγδαία εξέλιξη της, καθιστά αδύνατη την ύπαρξη ενός αναλυτικού και ολοκληρωμένου documentation. Για να ξεπεραστεί αυτό το πρόβλημα, οι προγραμματιστές από μόνοι τους προσπαθούν να γράφουν οδηγίες για τον τρόπο χρήσης της μέσα στα projects που υλοποιούν.

c. Επικεντρώνεται στο frontend

Η ReactJS καλύπτει αποκλειστικά το UI κομμάτι της εφαρμογής. Λόγω αυτού, ο προγραμματιστής είναι αναγκασμένος να χρησιμοποιήσει και άλλου είδους τεχνολογίες για να ολοκληρώσει την ανάπτυξη του project που το έχει ανατεθεί.

d. JSX

Λόγω της χρήσης JSX μέσα στη React, πολλοί προγραμματιστές (κυρίως εκείνοι που τη μαθαίνουν για πρώτη φορά) βρίσκουν δύσκολη την κατανόηση της και υψηλό το βαθμό δυσκολίας της. Αυτό οφείλεται στο γεγονός ότι η JSX είναι ένας τρόπος σύνταξης που συνδυάζει την HTML και τη JavaScript μαζί.

2.2.5. Πότε χρησιμοποιείται

Όπως αναφέραμε και παραπάνω, η ReactJS βοηθάει σημαντικά στην ανάπτυξη εφαρμογών, και συγκεκριμένα σε εφαρμογές με δυναμικό περιεχόμενο και πολλαπλά views. Ειδικότερα, οι κατηγορίες εφαρμογών στις οποίες οι δυνατότητες της ReactJS χρησιμοποιούνται στο έπακρον είναι αυτές των social media, video streaming, ενημερωτικές πλατφόρμες. Ενδεικτικά, κάποιες από αυτές τις εφαρμογές είναι το Facebook, Instagram, Netflix, η ιστοσελίδα των New York Times.

2.2.6. Συγκριτικός Πίνακας [11, 12]

	Angular JS	React JS
Γλώσσα Προγραμματισμού	JavaScript, HTML	JSX
Τύπος	Open Source MVC Framework	Open Source JavaScript Library
DOM	Πραγματικό DOM	Εικονικό DOM
Testing	Unit και Integration Testing	Unit Testing
Αρχιτεκτονική	MVC	Flux
Dependencies	Αυτόματη διαχείριση	Χρήση επιπρόσθετων εργαλείων
Απόδοση	Αργή εκτέλεση	Γρήγορη εκτέλεση, λόγω εικονικού DOM
Data Binding	Αμφίδρομη ροή δεδομένων	Μονόδρομη ροή δεδομένων
Εφαρμογές	CRUD λογική	Δυναμικού περιεχομένου
View Handling	SPA με ένα view	SPA με πολλαπλά views
Δημοτικότητα	2η σε προτίμηση	9η σε προτίμηση

3. Μέρος Β' : Συγκριτική μελέτη των αρχιτεκτονικών «Single-Page Application» και «Isomorphic Web Application»

Στο δεύτερο και τελευταίο μέρος της παρούσας εργασίας, παρουσιάζονται δύο από τις πιο διαδεδομένες αρχιτεκτονικές που υπάρχουν για την ανάπτυξη μιας web εφαρμογής. Αναλυτικότερα, γίνεται εκτενής αναφορά των πλεονεκτημάτων, καθώς και των μειονεκτημάτων που έχουν οι δυο αυτές αρχιτεκτονικές. Στο τέλος της ενότητας αυτής παρατίθεται ξανά ένας συνοπτικός πίνακας με τις διαφορές που εντοπίστηκαν μεταξύ των δύο αυτών αρχιτεκτονικών.

3.1. Single Page Application (SPA)

3.1.1. Ιστορία της SPA

Από την αρχή της ανάπτυξης web εφαρμογών , η αρχιτεκτονική του Multiple Page Application ήταν η πιο διαδεδομένη ανάμεσα στους προγραμματιστές. Βασική ιδέα της ήταν η δημιουργία ενός static HTML κώδικα σε συνδυασμό με μερικές τεχνολογίες που έκαναν χρήση server όπως η PHP, Java κ.α. Η MPA λειτουργούσε κάνοντας request μεταξύ client και server καθώς κάθε ιστοσελίδα ερχόταν από την πλευρά του server. Οι προγραμματιστές όμως χρειάζονταν μια ιστοσελίδα η οποία θα μπορούσε να ανανεώνει τα δεδομένα της χωρίς να φορτώνει η σελίδα εκ νέου. Αυτό επιτεύχθηκε το 2009 με την δημιουργία του Backbone.js το οποίο προσέφερε στους προγραμματιστές ένα μικρό client side framework που κατέστησε ευκολότερη την υλοποίηση SPA εφαρμογής. Παρόλα αυτά, οι προγραμματιστές έπρεπε και πάλι να γράφουν επαναλαμβανόμενο κώδικα. Έτσι, η πρώτη πραγματική υλοποίηση SPA πραγματοποιήθηκε με την έκδοση της Angular.js η οποία έκανε πράξη όλες αυτές τις ιδέες το 2010.

3.1.2. Γενικά για την SPA

Ένα Single Page Application είναι ένα Web Application το οποίο χρησιμοποιεί μία μοναδική HTML ιστοσελίδα ως τον πυρήνα της για όλες τις ιστοσελίδες της εφαρμογής και κάνει χρήση HTML, JavaScript και CSS για να επιτύχει επικοινωνία με τον χρήστη. Το μεγαλύτερο κομμάτι ανάπτυξης ενός SPA γίνεται στο front end σε αντίθεση με τις παραδοσιακές web εφαρμογές οι οποίες βασίζονται κυρίως στην επικοινωνία με έναν web server ο οποίος φορτώνει συνεχώς νέες σελίδες όταν υπάρχει περίπτωση ανακατεύθυνσης. Οι SPA εφαρμογές θυμίζουν αρκετά τις native εφαρμογές με την διαφορά ότι η διεργασία εκτελείται από τον browser και όχι από την ίδια την εφαρμογή.

Πιο συγκεκριμένα, μια Single Page Application είναι ένα web application το οποίο αλληλοεπιδρά με τον χρήστη γράφοντας δυναμικά πάνω στην παρούσα σελίδα. Αυτό επιτυγχάνεται ανανεώνοντας μόνο το HTML element του οποίου το content έχει αλλάξει και όχι φορτώνοντας εκ νέου την ιστοσελίδα. Σε πολλές ιστοσελίδες υπάρχει επαναλαμβανόμενο περιεχόμενο σε οποιοδήποτε σημείο και αν κάνει ανακατεύθυνση ο χρήστης. Το περιεχόμενο αυτό συνήθως είναι headers, footers, το λογότυπο της ιστοσελίδας, μπάρα πλοήγησης, layouts και templates.

Η χρήση της αρχιτεκτονικής Single Page Application εκμηδενίζει τον χρόνο αναμονής του χρήστη κατά την μετάβασή του από την μία σελίδα στην άλλη κάνοντας έτσι την εφαρμογή να μοιάζει περισσότερο με ένα desktop application.

3.1.3. Πότε χρησιμοποιείται [13, 14]

Η SPA χρησιμοποιείται κυρίως σε εφαρμογές οι οποίες επικεντρώνονται στο user interface και θέλουν να το εμπλουτίσουν με πολλές δυνατότητες. Ακόμη, χρησιμοποιείται όταν ο προγραμματιστής θέλει να κάνει χρήση JavaScript ή TypeScript, όταν η εφαρμογή θέλει να εκθέσει ένα API σε άλλους χρήστες, καθώς και όταν πρέπει να δημιουργηθεί μια ιστοσελίδα που χρειάζεται μια δυναμική πλατφόρμα και επεξεργάζεται λίγα δεδομένα. Επίσης, αν ο προγραμματιστής σκοπεύει να δημιουργήσει μια εκδοχή της εφαρμογής για κινητά, η SPA κρίνεται ως η καταλληλότερη αρχιτεκτονική. Παραδείγματα εφαρμογών που χρησιμοποιούν αυτήν την αρχιτεκτονική είναι το Facebook, το Google Maps, το Gmail, το Twitter, το GitHub κ.α.

3.1.4. Κριτήρια που ενισχύουν την επιλογή της SPA [15]

Υπάρχουν πολλά θετικά σχετικά με την χρήση της αρχιτεκτονικής SPA. Μερικά από αυτά είναι:

a. Γρήγορη κατά την εκτέλεση

Όπως αναφέρθηκε και παραπάνω, η SPA δεν χρειάζεται να φορτώσει εκ νέου τα αρχεία HTML, CSS και JavaScript, παρά μόνο μια φορά κατά την διάρκεια λειτουργείας της εφαρμογής. Αλλαγές γίνονται μόνο σε συγκεκριμένα HTML element, όταν αυτό είναι απαραίτητο, κάτι το οποίο καθιστά την εφαρμογή γρήγορη και αποδοτική.

b. Ο server δε φορτώνει άλλες σελίδες

Επιτυγχάνεται η απλότητα και η βελτιστοποίηση της ανάπτυξης της εφαρμογής, καθώς δεν υπάρχει η ανάγκη εγγραφής κώδικα για να φορτώσουν οι ιστοσελίδες από τον server. Η ανάπτυξη της εφαρμογής μπορεί να αρχίσει απευθείας από τον προγραμματιστή με την χρήση απλών φακέλων, χωρίς την ύπαρξη server.

c. Κατάλληλη για mobile εφαρμογές

Η ανάπτυξη mobile εφαρμογών με την χρήση SPA είναι πολύ απλή, αφού ο προγραμματιστής μπορεί να χρησιμοποιήσει τον ίδιο backend κώδικα από μία υπάρχουσα web εφαρμογή που έχει ήδη υλοποιήσει.

d. Αξιοποίηση μνήμης cache

Η SPA αξιοποιεί την cache οποιουδήποτε local storage αποδοτικά. Η εφαρμογή εφόσον αποθηκεύσει, μέσω ενός request , τα δεδομένα της στο local storage, είναι ικανή να τα ανακτήσει ακόμη και όταν βρίσκεται εκτός σύνδεσης.

e. Γρήγορο και αποδοτικό front-end

Μαζί με την γρήγορη εκτέλεση του που αναφέραμε παραπάνω, η SPA δίνει την δυνατότητα στον προγραμματιστή να δημιουργήσει και ένα γρηγορότερο front-end. Αυτό επιτυγχάνεται με την σωστή διαχώριση του front-end και του back-end. Πολλές από τις βασικές λειτουργίες του back-end δεν αλλάζουν καθόλου. Ο προγραμματιστής μπορεί να κρατήσει τα δεδομένα του back-end σταθερά και να τα μετατρέψει σε services τα οποία μπορεί αργότερα να χρησιμοποιήσει με διάφορους τρόπους στο front-end. Ακόμη, η αποθήκευση των δεδομένων του χρήστη γίνεται με την αξιοποίηση αυτών των services και η επεξεργασία τους γίνεται με την χρήση ξεχωριστών API. Έτσι, ο προγραμματιστής έχει την δυνατότητα να πειραματιστεί και να τροποποιήσει το front-end χωρίς φόβο για αλλοίωση των δεδομένων του back-end.

f. Ενισχυμένη εμπειρία προγραμματιστή

Τα framework της SPA είναι ιδανικά ώστε ο προγραμματιστής να μπορεί να πειραματιστεί με διάφορα modular services ώστε να δημιουργήσει μία δυναμική, ελκυστική και μοναδική εμπειρία στον χρήστη. Ακόμη, κάθε προγραμματιστής έχει διαφορετικές προτιμήσεις και οικειότητα σε γλώσσες προγραμματισμού. Χάρη στην ύπαρξη διαφορετικών API, η SPA που θα δημιουργήσει μπορεί να λειτουργεί με οποιαδήποτε γλώσσα προγραμματισμού θέλει στο back-end κάνοντας την εμπειρία του προγραμματισμού της ακόμα πιο ευχάριστη.

3.1.5. Μειονεκτήματα [16]

Ανεξάρτητα από τα θετικά γνωρίσματα της χρήσης της αρχιτεκτονικής Single Page Application, υπάρχουν αρκετοί λόγοι οι οποίοι δεν την καθιστούν την ιδανικότερη επιλογή των προγραμματιστών. Αναλυτικότερα:

a. Δε μπορεί να αξιοποιηθεί με τον καλύτερο τρόπο

Είναι πολύ δύσκολο και πολύπλοκο να αξιοποιήσει κάποιος στο έπακρο μια Single Page εφαρμογή, καθώς το περιεχόμενό της φορτώνεται μέσω AJAX. Αυτή η μέθοδος ανταλλαγής δεδομένων και ενημέρωσης της εφαρμογής γίνεται χωρίς να ανανεώνεται η σελίδα.

b. Αργή στη φόρτωση

Μια SPA είναι πολύ αργή στην φόρτωση αφού βαριά frameworks χρειάζεται να φορτώσουν από τον client. Εάν οι χρήστες έχουν αργές συσκευές, τότε θα έχουν μια κακή εμπειρία κατά την χρήση της εφαρμογής.

c. Εξάρτηση από τη JavaScript

Για να λειτουργήσει μια SPA χρειάζεται να υπάρχει στον browser ενεργοποιημένη η επιλογή χρήσης της JavaScript. Εάν είναι απενεργοποιημένη δεν είναι δυνατή η ομαλή εκτέλεση της εφαρμογής και ο χρήστης θα πρέπει να βρει έναν άλλο τρόπο για να αποκτήσει πρόσβαση στην εφαρμογή ή την ιστοσελίδα.

d. Λιγότερο ασφαλή

Σε σύγκριση με άλλες παραδοσιακές εφαρμογές, η SPA είναι λιγότερο ασφαλής. Αυτό οφείλεται στο Cross-Site Scripting, το οποίο επιτρέπει στους επιτιθέμενους να μπορούν να εισάγουν δικά τους scripts μέσα στην web εφαρμογή.

e. Ευαίσθητα δεδομένα

Άλλο ένα πρόβλημα που αντιμετωπίζει η SPA είναι τα ευαίσθητα δεδομένα. Η αρχική φόρτωση της σελίδα δεν πρέπει να περιέχει πληροφορίες που θα έπρεπε να αποκρύπτονται από τους χρήστες. Αφού η SPA φορτώνεται ολόκληρη και απευθείας στην συσκευή του χρήστη, υπάρχει περίπτωση να του εμφανιστούν δεδομένα τα οποία δε θα έπρεπε να του είναι διαθέσιμα.

f. Διαρροή μνήμης

Αφού η SPA κάνει χρήση της JavaScript, έτσι έχει υιοθετήσει και πολλά από τα αρνητικά της. Ένα από αυτά είναι η διαρροή μνήμης. Από τη στιγμή που η εφαρμογή τρέχει για αρκετή ώρα, ο προγραμματιστής πρέπει να ελέγχει πως η SPA δεν χρησιμοποιεί παραπάνω μνήμη από ότι χρειάζεται, αλλιώς οι σελίδες θα σταματήσουν να φορτώνουν γρήγορα.

3.2. Isomorphic Web Application

3.2.1. Γενικά για την αρχιτεκτονική [17]

Με την διάδοση της SPA στους προγραμματιστές, πολύ ήρθαν σε επαφή με μερικά προβλήματα της αρχιτεκτονικής αυτής, τα οποία δεν μπορούσαν να αποφύγουν. Για την επίλυση αυτών των προβλημάτων οι προγραμματιστές οδηγήθηκαν στην ανάπτυξη μιας νέας αρχιτεκτονικής, της Isomorphic web application.

Isomorphic JavaScript applications είναι εφαρμογές που χρησιμοποιούν το ίδιο JavaScript κώδικα μεταξύ browser client και web application server. Τέτοιες εφαρμογές είναι isomorphic με την έννοια ότι παίρνουν ίδια μορφή και σχήμα ανάλογα με το περιβάλλον στο οποίο εκτελούνται, είτε αυτό είναι client είτε server. Η Isomorphic JavaScript είναι το επόμενο βήμα στην ανάπτυξη της JavaScript σαν σύνολο. Ειδικότερα, μια isomorphic εφαρμογή προσπαθεί να συνδυάσει ένα server rendered web app με ένα single page application. Από την μία, θέλουμε τα εκμεταλλευτούμε την γρήγορη απόδοση και την SEO (Search Engine Optimization) ευνοϊκή φόρτωση από τον server. Από την άλλη, θέλουμε να διαχειριζόμαστε πολύπλοκες ενέργειες των χρηστών στον browser. Ακόμη, θέλουμε να αξιοποιήσουμε το browser push history και την ικανότητα να κάνουμε χρήση του XMLHttpRequest ώστε να ζητάμε πληροφορίες από τον server λιγότερο συχνά.

3.2.2. Πώς λειτουργεί [18, 19]

Για την καλύτερη κατανόηση του πως επιτυγχάνεται η σύνδεση μεταξύ μιας εφαρμογής που φορτώνεται μέσω ενός server και μιας SPA, γίνεται διαχωρισμός στη ροή των βημάτων που ακολουθούνται, αρχικά από τη μεριά του server και έπειτα από τη μεριά του browser.

Αρχικά, κάθε κλήση μιας web εφαρμογής ξεκινάει όταν ο χρήστης πληκτρολογήσει το URL της ιστοσελίδας ή κάνει κλικ σε κάποιο link της ιστοσελίδας που ήδη βρίσκεται. Ο browser στέλνει ένα αίτημα στο server ο οποίος και το λαμβάνει. Έπειτα ο server για να αποφασίσει ποια ιστοσελίδα πρέπει να φορτώσει, συλλέγει τα δεδομένα που απαιτούνται για το μέρος της εφαρμογής που έχει αιτηθεί και παράγει το HTML αρχείο με βάση τα δεδομένα αυτά. Τέλος, η απάντηση που στέλνει ο server στον browser είναι το πλήρες δομημένο HTML αρχείο.

Όσον αφορά το κομμάτι του browser, εκείνος αναλαμβάνει να απεικονίσει τον HTML κώδικα που έλαβε από τον server και το DOM πλέον μπορεί να υποστεί επεξεργασία. Σε αυτό το σημείο, εκτελείται η ροή μιας SPA, ώστε ο χρήστης να μπορεί να αλληλοεπιδράσει με την εφαρμογή. Όταν ο χρήστης κάνει κάποια ενέργεια, για παράδειγμα αν πατήσει κάποιο κουμπί ή προβεί σε καταχώρηση προσωπικών του στοιχείων, ο browser επικοινωνεί με τον server για να ανταλλάξουν δεδομένα. Αυτή η ροή εκτέλεσης επαναλαμβάνεται έως ότου ο χρήστης τερματίσει τη σύνδεση του με την εφαρμογή.

3.2.3. Πότε χρησιμοποιείται [20]

Οι isomorphic web εφαρμογές μπορούν να χρησιμοποιήσουν πολλά είδη frameworks και βιβλιοθηκών. Ειδικότερα, τα δύο framework που αναλύσαμε παραπάνω της Angular και της React είναι κατάλληλα για την χρήση αυτής της αρχιτεκτονικής. Στην React τα components της μπορούν να χρησιμοποιηθούν τόσο στο front end όσο και στο back end. Ομοίως στην Angular, το rendering της είναι διαθέσιμο σε όλα τα μέρη της εφαρμογής δίνοντας της καλύτερη απόδοση και βελτιστοποίηση στις μηχανές αναζήτησης. Άλλα framework που χρησιμοποιούνται για την επίτευξη isomorphism είναι το Meteor που είναι ένα από τα πιο δημοφιλή framework ανάμεσα στους προγραμματιστές, το Rendr το οποίο δημιουργήθηκε από την ίδια την εταιρία της Airbnb, το Derby.js το οποίο είναι ένα full stack framework για την υλοποίηση σύγχρονων web εφαρμογών, το Fluxible που είναι ιδανικό για isomorphic Flux εφαρμογές, κ.α.

Ενδεικτικά παραδείγματα εφαρμογών που κάνουν χρήση της αρχιτεκτονικής αυτής είναι η Asana, που είναι ένα εργαλείο διαχείρισης εργασιών, το Airbnb, το Telescope το οποίο χρησιμοποιεί Meteor και React για να πετύχει isomorphism και το Coursera που είναι isomorphic λόγο της χρήσης React και Flux.

3.2.4. Κριτήρια που ενισχύουν την επιλογή [21, 22]

Πολλοί προγραμματιστές βρίσκουν την αρχιτεκτονική αυτή δύσκολη και περίπλοκη και αποφεύγουν να τη χρησιμοποιήσουν. Ωστόσο, υπάρχουν αρκετοί λόγοι για να την επιλέξει κανείς. Μερικοί από αυτούς είναι:

a. Καλή χρήση SEO

Οι web εφαρμογές χρειάζονται καλό SEO ώστε να μεγιστοποιήσουν τον αριθμό ατόμων που επισκέπτονται την ιστοσελίδα μέσω κάποια μηχανής αναζήτησης. Οι SPA είναι δύσκολο να βρεθούν μέσω μηχανών αναζήτησης, καθώς δε φορτώνουν δεδομένα της εφαρμογής, μόνο αφού γίνει κλήση της JavaScript στον browser. Οι isomorphic εφαρμογές επίσης ξεκινούν να λειτουργούν αφού φορτώσει η JavaScript, αλλά επειδή το περιεχόμενο τους φορτώνεται από τον server, οι χρήστες δε χρειάζεται να περιμένουν για να ξεκινήσει η εφαρμογή και να εμφανιστεί το περιεχόμενο.

b. Γρήγορη απόδοση

Πολλές φορές οι SPA εφαρμογές καταλήγουν να έχουν αργές αποδόσεις ακόμη και αν ο χρήστης διαθέτει γρήγορη σύνδεση στο διαδίκτυο είτε έχει στην κατοχή του μια συσκευή τελευταίας γενιάς. Αυτό οφείλεται στο γεγονός ότι υπάρχει μεγάλο χρονικό κενό μεταξύ εκτέλεσης του προγράμματος και ανάκτησης των δεδομένων. Όπως είναι λογικό, αυτό δεν προσφέρει την καλύτερη εμπειρία στους χρήστες. Αντιθέτως, η χρήση isomorphic αρχιτεκτονικής καθιστά τις εφαρμογές πιο γρήγορες, αφού οι ιστοσελίδες που φορτώνονται από την πλευρά του server στέλνουν το περιεχόμενο τους στον browser και αυτός το απεικονίζει μόλις λάβει το HTML αρχείο που έχει δημιουργηθεί. Αυτό έχει σαν αποτέλεσμα το περιεχόμενο να προβάλλεται στον χρήστη αρκετά δευτερόλεπτα ταχύτερα συγκριτικά με τη χρήση SPA αρχιτεκτονικής. Παρόλο που ιστοσελίδα ακόμα χρειάζεται την εκτέλεση JavaScript για

να ξεκινήσει η αλληλεπίδραση με το χρήστη, αυτός έχει τη δυνατότητα να επεξεργαστεί το περιεχόμενο που προβάλλεται χωρίς ακόμα να μπορεί να προβεί σε κάποια ενέργεια.

c. Ανεξάρτητη από τη JavaScript

Άλλο ένα θετικό που προσφέρεται στους χρήστες των isomorphic εφαρμογών είναι ότι μπορούν να δουν τα κομμάτια της ιστοσελίδας που δεν απαιτούν τη χρήση JavaScript. Οι χρήστες που δε θέλουν ή δεν έχουν τη δυνατότητα να τρέξουν JavaScript κώδικα μπορούν και πάλι να ωφεληθούν από τη χρήση της ιστοσελίδας, αφού ο server δίνει μια ολοκληρωμένη HTML σελίδα στον browser. Αυτό έχει ως αποτέλεσμα την εξυπηρέτηση χρηστών που χρησιμοποιούν παλαιότερα browsers και συσκευές που δεν υποστηρίζουν τη JavaScript.

d. Λιγότερος κώδικας

Κατά την ανάπτυξη μιας isomorphic εφαρμογής, το περισσότερο μέρος του κώδικα μπορεί να χρησιμοποιηθεί τόσο για το server, όσο και για τον browser. Αν ο προγραμματιστής χρειάζεται ένα κομμάτι κώδικα να τρέξει και στα δύο μέρη, δεν είναι υποχρεωμένος να γράψει τον κώδικα αυτόν σε όλες τις γλώσσες προγραμματισμού που χρησιμοποιεί. Αυτό είναι ένα πλεονέκτημα για τις εφαρμογές που χρειάζεται να επικοινωνήσουν με κάποιο server και για τον προγραμματιστή που δε χρειάζεται να γράψει το ίδιο κομμάτι κώδικα σε διαφορετικές γλώσσες. Ως αποτέλεσμα αυτού, ο κώδικας της εφαρμογής είναι αρκετά λιγότερος και πιο εύκολος στη συντήρηση του.

3.2.5. Μειονεκτήματα [23]

Επιλέγοντας να δημιουργήσουμε μια εφαρμογή με βάση την Isomorphic web αρχιτεκτονική αντιμετωπίζουμε και τα αρνητικά της. Εκτός από το γεγονός ότι είναι ένας νέος τρόπος σκέψης για τον προγραμματιστή και χρειάζεται χρόνο για να προσαρμοστεί σε αυτόν, υπάρχουν και κάποιες άλλες δυσκολίες που θα κληθεί να αντιμετωπίσει

a. Πολυπλοκότητα σε debugging και ελέγχους

Όλος ο κώδικας χρειάζεται να ελεγχθεί δύο φορές, μία όταν φορτώνεται κατευθείαν από τον server και μία όταν φορτώνεται σαν κομμάτι της single page ροής. Το debugging απαιτεί από τον προγραμματιστή την άριστη γνώση τόσο των εργαλείων debugging του browser όσο και αυτών του server, καθώς και την κρίση του εάν το σφάλμα που έχει προκύψει βρίσκεται στον browser, στον server, ή και στα δύο περιβάλλοντα. Επιπλέον, χρειάζεται μία διεξοδική στρατηγική για το unit testing της εφαρμογής, όπου οι έλεγχοι γράφονται και εκτελούνται στο κατάλληλο περιβάλλον. Κώδικας που χρησιμοποιείτε μόνο από τον server θα πρέπει να ελεγχθεί μόνο στο περιβάλλον που τον χρησιμοποιεί, ενώ κοινός κώδικας πρέπει να ελεγχθεί και σε όλα τα περιβάλλοντα όπου θα τρέξει.

b. Μεγάλος όγκος φόρτου

Πολύ μεγάλο κομμάτι της υλοποίησης της εφαρμογής πρέπει να εκτελεσθεί από τον server. Η διαχείριση Http requests, το routing, η προβολή των σελίδων στον browser καθώς και ο σχεδιασμός τους μπορεί να γίνουν αρκετά πολύπλοκες διαδικασίες και να χρειαστούν εξωτερικές βιβλιοθήκες για την υλοποίησή τους.

c. Προβλήματα χρήσης του ίδιου κώδικας

Παρόλο που στα θετικά αναφέρθηκε πως το ίδιο κομμάτι κώδικα μπορεί να χρησιμοποιηθεί σε όλα τα περιβάλλοντα, αυτό δεν είναι πάντα εφικτό. Η πλευρά του server και η πλευρά του client έχουν διαφορετικές δυναμικές και ως αποτέλεσμα η χρήση του ίδιου κώδικα ενδέχεται να δημιουργήσει προβλήματα σε κάποιες περιπτώσεις.

Η ιδέα γύρω από την αρχιτεκτονική των Isomorphic Web Applications είναι κάτι το οποίο είναι σίγουρα χρήσιμο και προσφέρει πολλά στους προγραμματιστές. Παρόλα αυτά, πολλοί την χρησιμοποιούν χωρίς εμφανή λόγο. Η φόρτωση σελίδων από την πλευρά του server πρέπει να περιορίζεται μόνο όταν ο προγραμματιστής θέλει θα πετύχει την υποστήριξη παλαιότερων browser ή την γρηγορότερη εύρεση της ιστοσελίδας του από μηχανές αναζήτησης. Αν δεν κρίνονται αυτά σημαντικά τότε καλό θα ήταν να αποφευχθεί η υλοποίηση μιας τόσο βαριάς, από την πλευρά του server, αρχιτεκτονικής.

3.2.6. Συγκριτικός Πίνακας

	Single Page Application	**Isomorphic Web Application**
SEO	Κακή χρήση	Βέλτιστη χρήση του
Περιβάλλον υλοποίησης	Client side	Client side & Server side
JavaScript	Άμεση εξάρτηση	Τρέχει και χωρίς αυτήν
Page Loading	Αργή στην φόρτωση	Γρήγορη στην φόρτωση
Debugging	Εύκολο	Δύσκολο λόγο πολλών περιβαλλόντων
Rendering	Φορτώνει τον HTML και JavaScript κώδικα και τα στέλνει όλα ταυτόχρονα	Φορτώνει HTML, το στέλνει και έπειτα στέλνει το JavaScript
Γλώσσα προγραμματισμού	Χρήση διαφορετικών γλωσσών για client και server	Μέσο της ίδιας γλώσσας εκτελούμε εντολές και σε client και σε server
Κατανόηση	Εύκολη στην εκμάθηση	Δύσκολη λόγο πολλαπλών περιβαλλόντων

4. Βιβλιογραφία

[1] https://www.ryadel.com/en/angular-angularjs-history-through-years-2009-2019/
[2] AngularJS Documentation https://docs.angularjs.org/guide/introduction
[3] Vikrant Bhalodia,: AngularJS -What-Why, Advantages and Disadvantages https://www.weblineindia.com/blog/angularjs-development-advantages-disadvantages/
[4] Vikrant Bhalodia,: AngularJS -What-Why, Advantages and Disadvantages https://www.weblineindia.com/blog/angularjs-development-advantages-disadvantages/
[5] Vipul A M, Prathamesh Sonpatki, ReactJS by Example- Building Modern Web Applications with React (p.2) https://books.google.gr/books?hl=el&lr=&id=Ht3JDAAAQBAJ&oi=fnd&pg=PP1&dq=react+js&ots=EwGazRB7lM&sig=0wLfpa6kLAzV2uuLXqbUz05bppY&redir_esc=y#v=onepage&q=react%20js&f=false
[6] https://www.c-sharpcorner.com/article/what-and-why-reactjs/
[7] https://www.javatpoint.com/pros-and-cons-of-react
[8] Kutlu Sahin, Links: 7 Reasons why you should use React https://stories.jotform.com/7-reasons-why-you-should-use-react-ad420c634247
[9] Prayaag Kasundra, Why and Where Should you Use React for Web Development https://www.simform.com/why-use-react/
[10] https://www.javatpoint.com/pros-and-cons-of-react
[11] Mosh Hamedani, React vs. Angular: The Complete Comparison https://stories.jotform.com/7-reasons-why-you-should-use-react-ad420c634247
[12] JavaTPoint, Difference between AngularJS and ReactJS https://www.javatpoint.com/reactjs-vs-angularjs
[13] Microsoft Docs, Choose Between Traditional Web Apps And Single Page Apps https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps
[14] Zee Gimon, What is a Single Page Application https://huspi.com/blog-open/definitive-guide-to-spa-why-do-we-need-single-page-applications
[15] https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58
[16] https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58
[17] Jason Strimpel, Maxime Najim: Building Isomorphic JavaScript Apps: From Concept to Implemantation to Real-World Solutions (p.4) https://books.google.gr/books?hl=el&lr=&id=0nMNDQAAQBAJ&oi=fnd&pg=PR2&dq=isomorphic+app&ots=O5Bv-zsAD5&sig=1dId57C2jq2jcKUDHVe6swQBqA4&redir_esc=y#v=onepage&q=isomorphic%20app&f=false
[18] Elyse Kolker Gordon: An Introduction to Isomorphic Web Application Architecture https://medium.com/@ElyseKoGo/an-introduction-to-isomorphic-web-application-architecture-a8c81c42f59
[19] https://livebook.manning.com/book/isomorphic-web-applications/chapter-1/23
[20] Mehmetcan Gayberi: Isomorphic (Universal JavaScript https://medium.com/commencis/isomorphic-universal-javascript-496dc8c4341a
[21] https://livebook.manning.com/book/isomorphic-web-applications/chapter-1/23
[22] Spike Brehm: Isomorphic JavaScript: The Future Of Web Apps https://medium.com/airbnb-engineering/isomorphic-javascript-the-future-of-web-apps-10882b7a2ebc
[23] https://livebook.manning.com/book/isomorphic-web-applications/chapter-1/23