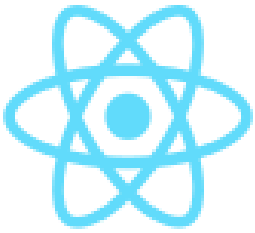
Πτυχιακή εργασία - Ημερομηνία Παράδοσης: *8/2/2018*
Φοιτητής: *Ντερμάρης Ευάγγελος - Α.Μ.:103573*
Επιβλέπων Καθηγητής: *Ράπτης Πασχάλης*



ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΩΝ ΕΦΑΡΜΟΓΩΝ ΜΕ ΤΗ ΒΙΒΛΙΟΘΗΚΗ

REACT**JS**

1. Εισαγωγή στις διαδικτυακές εφαρμογές 2

2. Αρχιτεκτονική διαδικτυακών εφαρμογών 3
2.1 Αρχιτεκτονική τριών στρωμάτων, three-tier architecture 4
2.2 Αρχιτεκτονική της εφαρμογής με React 6

3. Χαρακτηριστικά MVC 7

4. Front-end στις διαδικτυακές εφαρμογές 10

5. Εφαρμογή της ReactJs στις διαδικτυακές εφαρμογές 12
5.1 3D Γραφικά με reactJs 13
5.2 CMS με reactJs 13
5.3 Γραφικές αναπαραστάσεις με ReactJs 14

6. Χαρακτηριστικά της Reactjs 14
6.1 Αρχές σχεδίασης 16

7. Hello world σε react 17

8. Η εντολή create-react-app 19

9. Components 20
9.1 Component life-cycle 21

10. Routing 22

11. Redux 23

12. React Native 25

13. Isomorphic react 26

14. Styling components 27

15. Σύγκριση της ReactJs με άλλα framework 30

16. Debugging 31

17. Ανάλυση project-template-functions 32
17.1 Περιγραφή Εφαρμογής 32
17.2 Οι κλάσσεις του template 34
17.3 Ανάλυση μεθόδων που χρησιμοποιούνται 40

18. Σύνδεση με API και βάση δεδομένων 52

19. Εφαρμογή του template 55

20. Μελλοντικές επεκτάσεις 57
20.1 Τεχνικές αναβαθμίσεις 59

21. Συμπεράσματα - επίλογος 60

22. Βιβλιογραφία 61

1. Εισαγωγή στις διαδικτυακές εφαρμογές

Το διαδίκτυο αποτελεί εδώ και καιρό κομμάτι της καθημερινότητας πολλών ανθρώπων . Κι αυτό είναι αποτελέσμα της ελκυστικότητας της διεπαφής που έχει να προσφέρει η εκάστοτε εφαρμογή στο κοινό της.

Με τις ανάγκες των χρηστών του διαδικτύου ολοένα να αυξάνονται, οδηγηθήκαμε σε μία εποχή που οι διαδικτυακές εφαρμογές θα πρέπει να είναι και εύκολα προσβάσιμες από τον μέσο χρήστη, άλλα επιπλέον ικανές να υλοποιούν πολύπλοκα συστήματα, χωρίς αυτό να επηρεάζει την χρηστικότητά τους. Ταυτόχρονα, τα συστήματα που έχουν δημιουργηθεί για να διευκολύνουν ή εξελίξουν την ανάπτυξη αυτών των εφαρμογών ποικίλουν αρκετά, ενώ βελτιώνονται και δοκιμάζονται μέρα με τη μέρα.

Έτσι ο καθορισμός των απαιτήσεων της κάθε εφαρμογής και η εύρεση της βέλτισης λύσης είναι κρίσιμο κομμάτι και αποτελεί τα θεμέλια της εφαρμογής. Σαφώς κάποιος θα πρέπει να εστιάσει σε συγκεκριμένες τεχνολογίες και για τους ανάλογους σκοπούς, διότι οι επιλογές στις διαθέσιμες τεχνολογίες σήμερα μοιάζουν ατελείωτες, ειδικά αν ερευνηθεί η καθεμία απ' αυτές πιο αναλυτικά.

Ωστόσο η αρχιτεκτονική των διαδικτυακών εφαρμογών δεν μπορεί να ξεφύγει απο ορισμένα πλαίσια και τίνει να ακολουθεί τα ίδια μοτίβα προσέγγισης και ανάπτυξης εδώ και καιρό. Και όντως η γενικότερη προσέγγιση στις διαδικτυακές τεχνολογίες αφορά το που υπάρχουν και μεταβάλλονται τα δεδομένα , δηλαδή στην πλευρά του διακομιστή ή του πελάτη.

Επιπλέον γίνονται διαφοροποιήσεις σε αυτές τις τεχνολογίες ανάλογα με την χρήση τους: στην πλευρά του διακομιστή, να εξυπηρετούν πελάτες ή να αποθηκεύουν δομημένα δεδομένα ,ενώ στην πλευρά του χρήστη να δέχονται,μοντελοποιούν,εμφανίζουν ή τροποποιούν τα εκάστοτε δεδομένα. Όμως αυτές οι τεχνολογίες πρέπει να συνδυάζονται και να επικοινωνούν μεταξύ τους, οπότε ο συνδυασμός τους δύναται να διευκολύνει την ανάπτυξη του έργου.

Έτσι έχουν δημιουργηθεί στοίβες τεχνολογιών που παρέχουν ολοκληρωμένο περιβάλλον για διαδικτυακές εφαρμογές , ορισμένες από τις οποίες είναι οι LAMP

(Linux -Apache -MySql -Php) και η MEAN (Mongodb -Express -Angular -NodeJs). Αυτές είναι έτοιμες και αναλαμβάνουν όλη τη διαμεσολάβηση ανάμεσα στον χρήστη και στα δεδομένα.

Ωστόσο και αυτές επεκτείνονται περαιτέρω με τη χρήση τεχνολογιών που αναλαμβάνει ο χρήστης , τις front-end τεχνολογίες δηλαδή, οι οποίες περιλαμβάνουν τεχνολογίες όπως css, html5, και διάφορα framework σε javascript όπως το jquery.

Επιπλέον υπάρχουν πλατφόρμες που εγκαθιστούνται στα εκάστοτε συστήματα και δημιουργούν μία ενιαία διεπαφή για τον χρήστη, στο περιβάλλον LAMP για παράδειγμα το wordpress, το drupal, ή το joomla. Αυτές υλοποιούν έτοιμες δομές , αλλά λόγω τις διεπαφής που παρέχουν , είναι επίσης και εύκολα επεκτάσιμες με θέματα και πρόσθετα.

2. Αρχιτεκτονική διαδικτυακών εφαρμογών

Η αρχιτεκτονική των διαδικτυακών εφαρμογών δεν διαφέρει πολύ από εφαρμογή σε εφαρμογή, στην περίπτωση που οι εφαρμογές επικοινωνούν δεδομένα με τους χρήστες. Θα πρέπει να υπάρχει δηλαδή ο server ο οποίος θα αποκρίνεται στο εκάστοτε ερώτημα, στο χρήστη, ή να επικοινωνεί με τη βάση. Δεύτερον χρειάζεται να υπάρχει μια βάση για τα δεδομένα , ώστε να αποτραπεί η αποθήκευση των δεδομένων σε αρχεία , και να υπάρχει δυναμική πρόσβαση σε αυτά. Και τρίτον χρειάζεται η κατεύθυνση της διεπαφής του χρήστη στο πως να παρουσιαστεί, το οποίο συμβαίνει σε επίπεδο client.

Για τεχνολογίες server αρχικά μπορούμε να διαλέξουμε βάση γλώσσας προτίμησης: apache για php, nodejs για javascript κ.α. Σε αυτό μπορεί να τρέχει ένα framework το οποίο θα δημιουργεί δυναμικά τις αποκρίσεις, όπως το express για javascript. Επιπλέον μπορούν να εγκατασταθούν επιπλέον βιβλιοθήκες για να επεκταθεί η λειτουργικότητα.

Βάση αυτής της τεχνολογίας επιλέγεται και το υποστηριζόμενο λειτουργικό σύστημα: linux για apache, nodejs για javascript , iis για asp.net.

Επόμενη έρχεται η βάση δεδομένων, όπου επιλέγεται αναλόγως του τι μπορεί να εγκατασταθεί στον server, αλλά και των διεπαφών που παρέχονται για την εκάστοτε πλατφόρμα, αλλά και τις φιλοσοφίας που ακολουθείται, με την sql έχουμε δεδομένα σε πίνακες, με τη mongoDB σε αντικείμενα και με τη graphQl σε γράφους.

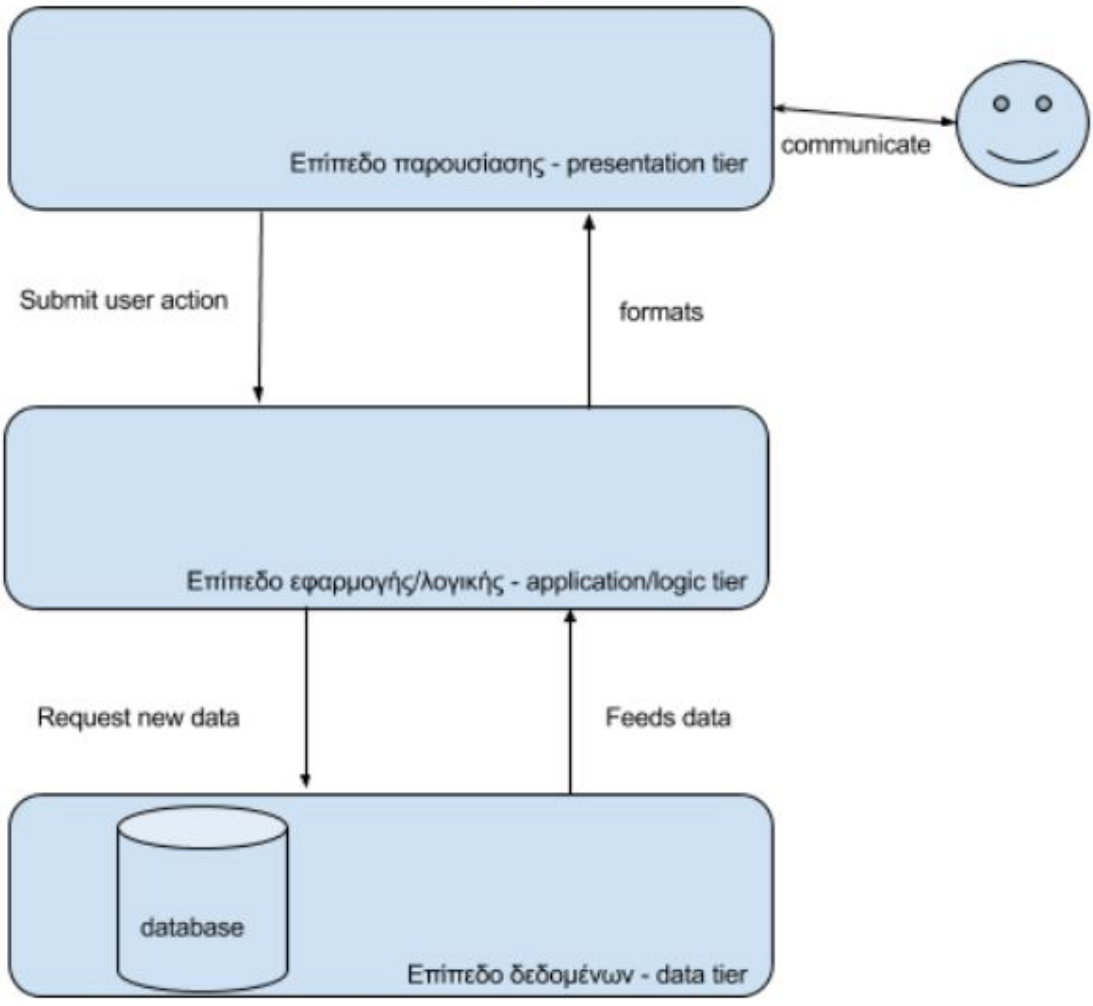
Με αυτή τη στοίβα έτοιμη ενδεχομένως να είναι επιθυμητό ένα υψηλότερο επίπεδο αρχιτεκτονικής για την εφαρμογή, το οποίο είναι εφικτό με ένα framework όπως το meteor για τη javascript,ή το laravel για τη php. Η διεπαφή που παρέχεται διευκολύνει την παραγωγή κώδικα βάση των σύγχρονων προγραμματιστικών προτύπων, και αναλαμβάνει τη σύνδεση και αλληλεπίδραση των επιμέρους τεχνολογιών.

Στη τεχνολογία του χρήστη ωστόσο μπορεί να γίνει επιλογή, σχεδόν ανεξάρτητα από τα υπόλοιπα, βάση των εκάστοτε αναγκών. Συνήθως αυτό το αποτελέσματα το δημιουργεί δυναμικά ο server ή το framework, οπότε είναι ουσιώδεις να συνδυάζονται αρμονικά η τεχνολογία που χρησιμοποιείται στο αποτέλεσμα, με τη διαδικασία παραγωγής της. Η php και τα frameworks της για παράδειγμα δημιουργούν html/html5 σελίδες, οι οποίες μπορεί να περιέχουν javascript και css, ενώ στη περίπτωση τις javascript, όπως angular, αυτές οι σελίδες προκύπτουν από τα template, και στη περίπτωση της react από σύνθετα components, ενώ υπάρχουν σαν αντικείμενα και στο χρήστη διότι είναι σε γλώσσα που καταλαβαίνει, σε αντίθεση με τη php η οποία τρέχει στο server και στέλνει το αποτέλεσμα στο client. Σε ποιο προχωρημένα μοντέλα αρχιτεκτονικής μπορούν να περιέχονται react component μέσα σε template της angular.

2.1 Αρχιτεκτονική τριών στρωμάτων, three-tier architecture

Το front-end και το back-end των web εφαρμογών αντιπροσωπεύουν μέρη της αρχιτεκτονικής των τριών στρωμάτων. Η αρχιτεκτονική των τριών στρωμάτων προκύπτει από το επιχειρησιακό μοντέλο των εφαρμογών που διαχωρίζει τις εφαρμογές σε τρία στρώματα:

- Το επίπεδο της παρουσίασης που αφορά τη διεπαφή αλληλεπίδραση με το χρήστη, και περιλαμβάνει τη γραφική αναπαράσταση και επικοινωνία των δεδομένων προς το χρήστη
- το επίπεδο της εφαρμογής ή λογικής ,που περικλείει τις μεθόδους που ορίζουν τη συμπεριφορά της εφαρμογής στα δεδομένα
- και το τρίτο επίπεδο των δεδομένων περιλαμβάνει τη βάση δεδομένων και το λογισμικό για τη διαχείρισή της.



Το επίπεδο παρουσίασης αποτελεί το front-end στις διαδικτυακές εφαρμογές, το επίπεδο βάσης δεδομένων αποτελείται από το back-end και στο ενδιάμεσο επίπεδο έχουμε λογισμικό του server που ενώνει τα δύο παραπάνω όπως τα Symfony, Spring, ASP.NET, Django, Rails .

2.2 Αρχιτεκτονική της εφαρμογής με React

Στην εφαρμογή που μελετήθηκε ακολουθήθηκε μια πιο αφηρημένη μορφή αυτής της αρχιτεκτονικής, η οποία όμως λόγω του υψηλού επιπέδου της, είναι εύκολα τροποποιήσιμη και επεκτάσιμη. Υπάρχει ο server, ο οποίος τρέχει nodejs, στον οποίο έχουμε τη δυνατότητα να τρέξουμε javascript και να παράγουμε την απόκριση για το χρήστη επίσης σε javascript.

Αυτή η ομοιομορφία στις τεχνολογίες κάνει την ανάπτυξη πιο εύκολη. Τον συνδυασμό αυτό τον αναλαμβάνει μια βιβλιοθήκη που ονομάζεται webpack και υποστηρίζει javascript συντακτικό στα αρχεία όπως es6. Στις ρυθμίσεις του server μας ορίζουμε το default αρχείο που θα αποσταλεί ως απάντηση, τις βιβλιοθήκες που χρειαζόμαστε, απ' αυτές που έχουμε εγκαταστήσει, και άλλες εκάστοτε παραμέτρους.

Έτσι προκύπτει το αρχείο του χρήστη, το οποίο υλοποιεί την εφαρμογή σε react, διότι η react,λόγω του ότι βασίζεται σε javascript, είναι εκτελέσιμη από τον χρήστη. Οπότε τα components που δημιουργούνται στο server σαν αρχεία και κλάσσεις, τα βλέπουμε να αναπαριστώνται και στον client.

Η τεχνολογία βάσης δεδομένων που χρησιμοποιήθηκε ωστόσο έχεις κάποιες διαφοροποιήσεις από τις προαναφερόμενες και σύνηθες τεχνολογίες, και αυτό έγινε διότι αποτελεί μια πιο αφηρημένη προσέγγιση στην επικοινωνία με τη βάση δεδομένων. Η λογική ωστόσο είναι η ίδια : με εντολή του χρήστη συλλέγονται και εμφανίζονται ή τροποιούνται δεδομένα. Και αυτό είναι τόσο απλό διότι και τα δεδομένα από τη βάση, τελικά αναπαριστούνται σε αντικείμενα, το οποίο είναι και αρχιτεκτονικά σωστό, και όχι απλώς σε εμφανίσιμα δεδομένα.

Πρόκειται για τη διεπαφή firebase, η οποία μας επιτρέπει επικοινωνούμε τα δεδομένα μας κατευθείαν με τη διεπαφή, χωρίς να χρειάζεται η εγκατάσταση και εκτέλεση βάσης δεδομένων,αλλά παραμετροποίηση του αρχείου σύνδεσης και εισαγωγή της αντίστοιχης βιβλιοθήκης.

Επιπλέον για τα γραφικά της διεπαφής χρησιμοποιήθηκε η πλατφόρμα material-ui, η οποία υλοποιεί τα βασικά components, όπως textfield,dialog και switch, αλλά τα υλοποιεί στο framework της react και άρα είναι έτοιμα να αλληλεπιδράσουν,

να επεκταθούν και να συνδυαστούν σε πιο σύνθετα components. Το material-ui ανήκει στην κατηγορία framework που αναλαμβάνουν τη διεπαφή των εμφανίσιμων μερών στον χρήστη, σαν το bootstrap, και δεν είναι απαραίτητα, αλλά σίγουρα διευκολύνουν το προγραμματισμό ενώ οδηγούν και στην εφαρμογή σωστών πρακτικών.

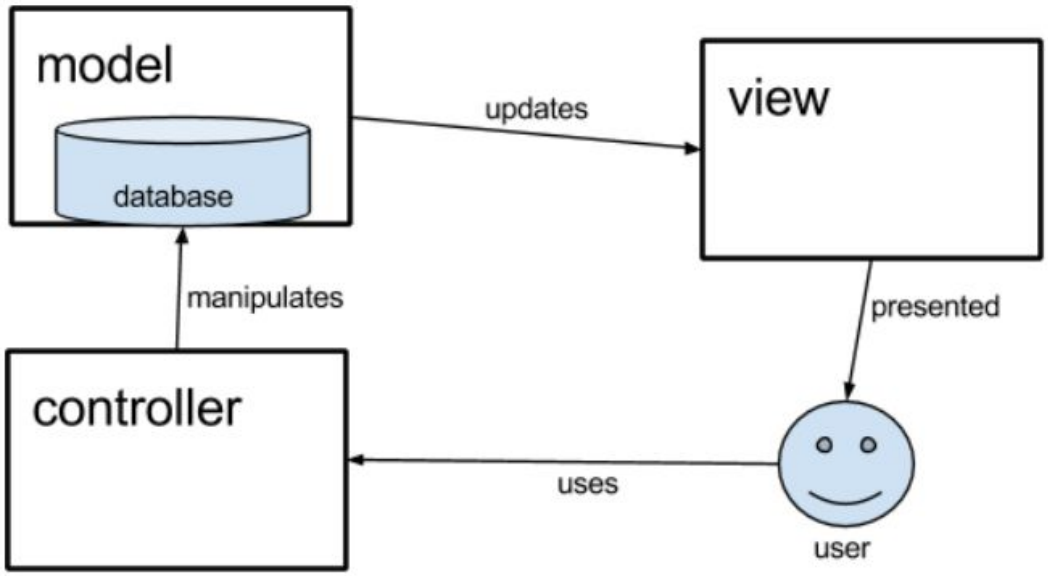
3. Χαρακτηριστικά MVC

Σε όλες τις εφαρμογές, ανεξάρτητα αν επικοινωνούν με το διαδίκτυο ή όχι, υπάρχει η ανάγκη διαχώρισης των επιμέρους στοιχείων ανάλογα με το πως επηρεάζουν τα δεδομένα. Τη λύση σε αυτό έρχεται να δώσει το μοντέλο MVC(model-view- controller), το οποίο ορίζεται ως το αρχιτεκτονικό πρότυπο που ακολουθούν οι εφαρμογές. Βάση αυτού, μπορούμε να οργανώσουμε τον κώδικα ή το σύστημά μας σε τρεις ομάδες συστατικών:

- Το μοντέλο (model), το οποίο αφορά την ρεαλιστική αναπαράσταση των δεδομένων, και τη διατήρηση της λογικής και των κανόνων που απαιτούνται μεταξύ τους. Ορίζει την συμπεριφορά του προγράμματος χωρίς την επιρροή του χρήστη.
- Την όψη (view), η οποία αναλαμβάνει την αναράσταση του αποτελέσματος στο χρήστη. Η ανάγκη για την όψη είναι ότι στις σύγχρονες εφαρμογές, δύναται να εμφανίζουμε τα ίδια δεδομένα με πολλούς τρόπους, και άρα πρέπει απλά να δημιουργήσουμε διαφορετικές όψεις, αντί να επέμβουμε στο μοντέλο και να το τροποποιήσουμε βάση των αναγκών μας.
- Τον ελεγκτή (controller), ο οποίος είναι υπεύθυνος για την επικοινωνία μοντέλου - όψης. Ο ελεγκτής είναι αυτός που βάση των παραμέτρων που του δίνονται, μεταβάλλει τα δεδομένα σε αντίστοιχες πληροφορίες έτοιμες για το μοντέλο αναλόγως με τις εντολές του χρήστη.

Εκτός από αυτά τα κύρια συστατικά, το μοντέλο MVC κάνει διαφοροποίηση στις αλληλεπιδράσεις μεταξύ των συστατικών του:

- Το μοντέλο είναι υπεύθυνο για τη διατήρηση των σωστών δεδομένων, αλλά επίσης πρέπει να ακούει ή απαντάει στις ανάλογες αιτήσεις του ελεγκτή.
- Η όψη είναι αυτή που αλληλεπιδρά άμεσα με τον χρήστη, και του επικοινωνεί τα δεδομένα βάση της απόκρισης του ελεγκτή.
- Ο ελεγκτής είναι υπεύθυνος να ακούει την είσοδο του χρήστη, να ελέγχει και προετοιμάζει τα εισερχόμενα δεδομένα για μοντέλο αρχικά, όμως σε άλλες περιπτώσεις να συλλέγει και μορφοποιεί απ' αυτό τα δεδομένα που απαιτούνται στην εκάστοτε όψη.



Επειδή το mvc είναι αρχιτεκτονικό πρότυπο, οι περισσότερες εφαρμογές μπορούν να σχεδιαστούν αφηρημένα και να οργανωθούν βάση αυτού. Έτσι η έννοιά του, έρχεται να μας βοηθήσει και στο πλαίσιο της ανάπτυξης διαδικτυακών εφαρμογών. Οι διαδικτυακές εφαρμογές, λόγω των τεχνολογιών που πρέπει να συνδυαστούν μεταξύ τους, χρησιμοποιούν το μοντέλο MVC ώστε να προσθέσουν ένα υψηλότερο αρχιτεκτονικά επίπεδο. Μερικά παραδείγματα υλοποίησης του MVC αποτελούν το Django,η ruby on Rails και το Asp.net-mvc . Ωστόσο αυτά βασίζονται

στην κλασσική νοοτροπία να αναθέσουμε το μεγαλύτερο μέρος της εκτέλεσης στον server, και έτσι να μην επιβαρύνουμε τον client.

Παρ' όλ' αυτά όμως, επειδή το mvc αποτελεί αφηρημένη αρχιτεκτονική προσέγγιση, και οι ανάγκες των σύγχρονων εφαρμογών ποικίλουν, μπορούμε να το εφαρμόσουμε και σε εφαρμογές που αναθέτουν μεγαλύτερο φόρτο στο χρήστη, και το αποτέλεσμά που θα δει δεν παράγεται αποκλειστικά στον server, με μερικές γνωστές απ'αυτές τις τεχνολογίες να είναι η AngularJs, VueJs, ReactJs, EmberJs, ajax, ενώ όπως παρατηρούμε έχουν όλες κοινή γλώσσα την javascript, διότι τρέχει στον client.

Αξίζει όμως να αναφερθεί ότι ορισμένα javascript framework αναφέρονται στην όψη και μπορεί να τρέχουν αποκλειστικά στον client, αλλά επεκτείνονται ώστε να υπάρχουν στον server και να αναλαμβάνουν σε σημαντικό βαθμό αρμοδιότητες μοντέλου ή ελεγκτή. Ως εκ τούτου στην υλοποίηση της πλατφόρμας δεν είναι απαραίτητο να ξεχωρίζουν αυτά τα συστατικά, με τον ίδιο τρόπο που ξεχωρίζουν στην αρχιτεκτονική της.

Μπορεί λοιπόν να γίνει διάκριση σε μερικά από τα οφέλη χρήσης mvc, με βασικότερο απ'αυτά να αποτελεί η ταυτόχρονη υλοποίηση της πλατφόρμας από διάφορους προγραμματιστές, αφού έχουν καθοριστεί τα συστατικά και οι αλληλεπιδράσεις τους κατά τον σχεδιασμό.

Τα συστατικά αυτά αν σχεδιαστούν σωστά, μας παρέχουν επίσης αυξημένη συνοχή, δηλαδή αλληλεπιδρούν αρμονικά μεταξύ τους, και μειωμένη σύζευξη, δηλαδή δεν είναι εξαρτημένα το ένα από το άλλο; δύο πρότυπα κλειδιά για την άρτια αρχιτεκτονική μιας εφαρμογής.

Αυτά επίσης οδηγούν στην ευκολία επέκτασης και τροποποίησης της πλατφόρμας , ενώ δεν πρέπει να ξεχαστεί, πως για ένα καλοσχεδιασμένο μοντέλο, μπορούν να παραχθούν διάφορες όψεις ανάλογα με τις απαιτήσεις.

Αν και η χρήση του mvc οδηγεί σε οργανωμένη αρχιτεκτονική, χρειάζεται η προσοχή στην επιλογή του, διότι μπορεί αυτό να κάνει ένα απλό έργο, πολύπλοκο. Γενικότερα το mvc κάνει την πλατφόρμα πολυπλοκότερη διότι εισάγει ένα νέο επίπεδο αφαίρεσης. Επιπλέον ο κώδικας διασπάται σε δυνατόν μικρότερα μέρη, με περιορισμένες αλλά σαφής δυνατότητες, τα οποία χρειάζονται περαιτέρω προσοχή ώστε να αλληλεπιδρούν αρμονικά μεταξύ τους.

Τελικά δεν πρέπει να παραλείπεται πως το mvc έρχεται να συμπληρώσει τη στοίβα που χρησιμοποιούμε, δεν λειτουργεί αυτόνομα, και άρα χρειάζεται η γνώση και η εμπειρία από τους προγραμματιστές με τις εκάστοτε τεχνολογίες σε συνδυασμό με το μοντέλο.

4. Front-end στις διαδικτυακές εφαρμογές

Ένα πολύ σημαντικό χαρακτηριστικό των διαδικτυακών εφαρμογών αποτελεί η διεπαφή με τον χρήστη. Τα κομμάτια που υλοποιούν την διεπαφή αυτή, τα αποκαλούμε front-end και διαφέρουν με τα υπόλοιπα κομμάτια τα οποία επηρεάζουν τη λειτουργία του server και τα κατηγοριοποιούμε ως back-end. Το front-end αφορά κυρίως την όψη ενώ επιπροσθέτως μπορεί να υλοποιεί και λειτουργίες του controller.

Όταν λοιπόν αναφερόμαστε σε front-end, αναφερόμαστε στα συστατικά που επηρεάζουν πρωτίστως την επικοινωνία των δεδομένων στον χρήστη, δηλαδή την οργάνωση των δεδομένων και μορφοποίηση της εμφάνισής τους, ενώ επιπλέον μπορεί να προετοιμάζει αυτά τα δεδομένα για το μοντέλο, υλοποιώντας έτσι και λειτουργίες ελεγκτή.

Η διαφοροποίηση του front-end με το back-end, λόγω της διαφοράς φιλοσοφίας που τα ξεχωρίζουν, διακρίνουν τους προγραμματιστές διαδικτυακών εφαρμογών σε δύο αντίστοιχες κατηγορίες.

Οι πρώτοι, που εξειδικεύονται στο front-end, δίνουν βαρύτητα στις μορφοποιήσεις που χρειάζεται η πλατφόρμα, ώστε να είναι εύχρηστη και ελκυστική, στη παραγωγή απολεσμάτων βάση των εντολών του χρήστη, στη χρήση ajax για ασύγχρονη ανανέωση μερών της σελίδας.

Οι δεύτεροι, που εξειδικεύονται στο back-end, δίνουν βαρύτητα στις λειτουργίες που πρέπει να υλοποιεί ο server προκειμένου να διασφαλιστεί η ακεραιότητα του μοντέλου και των δεδομένων του, στη σωστή κατανάλωση των αιτήσεων του χρήστη και την παραγωγή δεδομένων για κατανάλωση από τις όψεις, ενώ επίσης την διαχείρηση φόρτου, ασφαλείας και όλων των ποικίλων διαδικασιών

χρειάζεται να τρέχουν στον server. Εύκολα διακρίνει κάποιος, πως η έννοια του ελεγκτή μπορεί να υπάρχει ταυτόχρονα και στο back-end, και στο front-end.

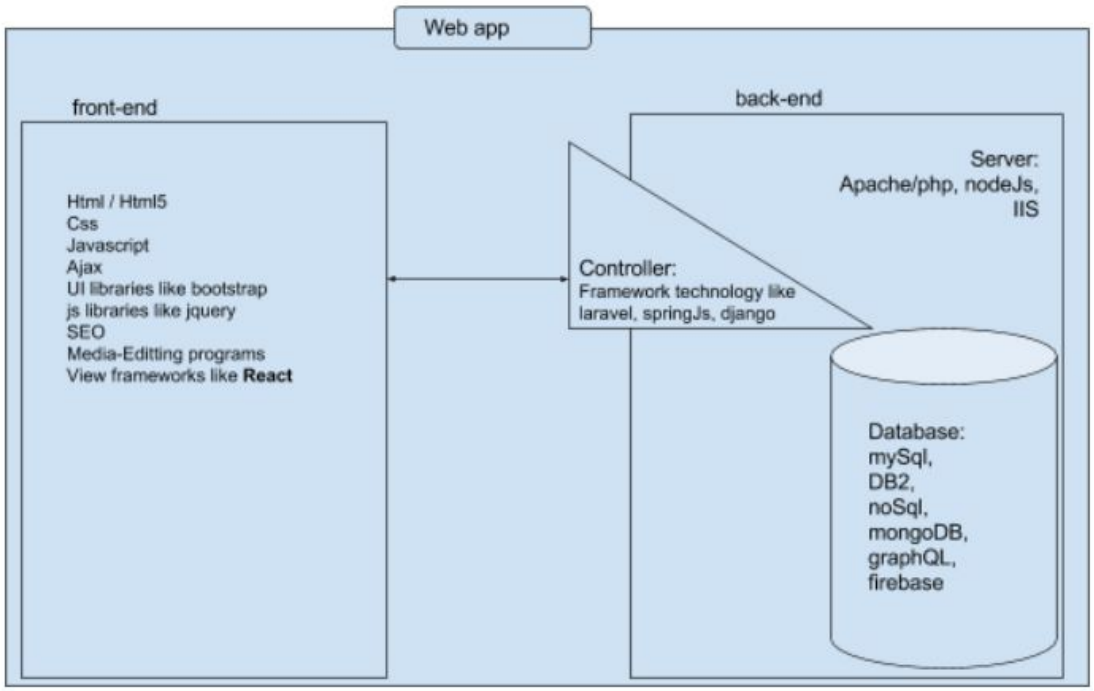
Η έννοια του front-end δεν περιορίζεται στις διαδικτυακές εφαρμογές ωστόσο. Σε οποιαδήποτε εφαρμογή που υπάρχει αλληλεπίδραση με τον χρήστη, υπάρχει και η έννοια του front-end. Επιπροσθέτως ο σωστός σχεδιασμός του front-end, είναι αυτό που θα μας οδηγήσει στην αποτελεσματική αξιοποίηση του εκάστοτε μοντέλου.

Έκτός όμως από το σχεδιασμό του front-end, δεν πρέπει να παραλείψουμε την αποδοτικότητα αυτού και το χρόνο απόκρισης, το οποίο μπορεί να αποτελέσει τη λεπτομέρεια σε δύο παρόμοια σχεδιασμένα front-end και τελικά την προτίμηση του χρήστη σε κάποιο. Για να καταλάβει κάποιος τη διαφορά αρκεί να σκεφτεί μία εφαρμογή με μία φόρμα αλληλεπίδρασης, η όποια στη μία περίπτωση χρειάζεται να ανανεώσει όλη τη σελίδα, ενώ στην άλλη μόνο τα απαραίτητα συστατικά της σελίδας που χρειάζεται να αλλάξουν.

Η αποδοτικότητα αυτή μπορεί εύκολα να βελτιωθεί με τη χρήση των προγραμματιστικών προτύπων και βέλτιστων πρακτικών που οδηγούν σε υψηλής ποιότητας και συντηρίσιμο κώδικα.

Επιπλέον για να είναι αναπαραστατικό το front-end , είναι αναγκαίο να παρέχονται σε αυτό τα δεδομένα από το μοντέλο με την σωστή μορφή. Κάτι το οποίο οδηγεί και στην εύκολη εύρεση αυτών. Σε αυτό το γεγονός αναφέρεται το SEO, search engine optimization. Την οργάνωση και ταξινόμηση δηλαδή των δεδομένων ώστε να είναι αναπαραστατικά , πλοηγίσιμα και εύκολα αναζητίσιμα. Παρόλο που το μοντέλο είναι ανεξάρτητο από το front-end, η οργάνωση αυτή βοηθάει στην ρεαλιστικότερη απεικόνισή του.

Τελικά οι προγραμματιστές του front-end, εκτός από τις τεχνολογίες javascript που επιτρέπουν τη σύνθετη προγραμματιστική λογική και ασύγχρονη επικοινωνία των δεδομένων, έρχονται συχνά σε επαφή με διάφορες τεχνολογίες πολυμέσων, όπως οι εικόνες. Οι μορφοποιήσεις που προσφέρονται για αυτά τα αρχεία είναι πολλές και δεν είναι σπάνιο να χρειάζεται η μετατροπή τους σε ένα πρόγραμμα τρίτου παρόχου, όπως το photoshop, πριν χρησιμοποιηθούν στην εφαρμογή.



5. Εφαρμογή της ReactJs στις διαδικτυακές εφαρμογές

Όπως φαίνεται η τεχνολογία ReactJs αφορά την δημιουργία πολύπλοκων διεπαφών, ικανές να εκτελούνται στην μεριά του χρήστη. Παράλληλα λόγω του ότι βασίζεται σε javascript, μπορεί να συνδυαστεί με όλες τις τεχνολογίες που τρέχουν στον server, ενώ μπορεί να τρέχει και η ίδια στον server. Εκτός απ'αυτό η reactJs έχει δημιουργηθεί ώστε να εκτελείται σε browser διαφορετικών λειτουργικών συστημάτων, ενώ επιπροσθέτως επεκτείνεται με τη react native, η οποία μας επιτρέπει να προσαρμόσουμε την εφαρμογή μας ώστε να τρέχει σε περιβάλλον javascript χωρίς φυλλομετρητή, όπως είναι οι εφαρμογές android για παράδειγμα.

Αξίζει επιπλέον να αναφερθεί πως η ReactJs δημιουργήθηκε από το facebook, ως η διεπαφή που θα αναλάμβανε την εμφάνιση των δεδομένων στο χρήστη. Δημιουργήθηκε δηλαδή με σκοπό την υλοποίηση πολύπλοκων διεπαφών

αλληλεπίδρασης, όπως αυτή του facebook, και βλέπουμε να υιοθετείται και από άλλες πλατφόρμες με παρόμοιες ανάγκες όπως το instagram, το netflix, το dropbox, οι εκπαιδευτικές πλατφόρμες khan academy και codecademy, το yahoo-mail, ενώ και το whatsapp για περιβάλλον android.

Οι εφαρμογές αυτές κατάφεραν με τη βοηθεία της reactJs να δημιουργήσουν ένα έυχρηστο και ανταποκρίσιμο περιβάλλον για το χρήστη, το οποίο χρειάζεται να υλοποιεί διάφορες πολύπλοκες όψεις για το εκάστοτε μοντέλο, και αυτό διότι η ReactJs βοηθάει την ανάπτυξη του front-end, βάση του component-based αρχιτεκτονικού μοντέλου της εφαρμογής. Επιπλέον η όψη της διεπαφής αυτής μπορεί να συνδεθεί με συστήματα που επεκτείνουν την αρχιτεκτονική της εφαρμογής.

5.1 3D Γραφικά με reactJs

Επειδή η React παρέχει τη διεπαφή για τη διαχειρίση της όψης, μπορεί να χρησιμοποιηθεί και για τη διαχειρίση τρισδιάστατων γραφικών. Αυτά είναι γραφικά από 3d βιβλιοθήκες σε javascript , όπως η three.js. Εμπεριεχόνται στη react ως κλάσσεις με τη χρήση αντίστοιχων συνδετικών βιβλιοθηκών. Έτσι μπορούμε να έχουμε components με κλάσσεις αντικειμένων three.js, το οποίο σημαίνει ότι είναι εφικτή η ανάλογη διαχείριση για τα στιγμιότυπα της εφαρμογής που εμφανίζονται για το κάθε component , η χρήση των κλάσσεων που παρέχονται για τα components της react, αλλά και η διαφοροποίηση της λογικής του μοντέλου από την όψη.

Επίσης είναι διαθέσιμη και η πλατφόρμα Aframe , που βασίζεται στη βιβλιοθήκη three.js, και αφορά την ανάπτυξη διαδικτυακών εφαρμογών εικονικού περιβάλλοντος, virtual reality. Και για αυτήν υπάρχουν έτοιμες βιβλιοθήκες που συνδέουν τη React με τα αντικείμενα της βιβλιοθήκης Aframe.

5.2 CMS με reactJs

Στην reactJs πολλές φορές τίθεται το θέμα σύγκρισής της με συστήματα διαχείρισης περιεχομένου, cms . Η react παρόλο που δεν αποτελεί μία τέτοια πλατφόρμα , μπορεί να αποτελέσει το περιβάλλον για την ανάπτυξη ενός τέτοιου

συστήματος. Παραδείγματα εφαρμογής της react με αυτό το σκοπό είναι διαθέσιμα σε μεγάλο βαθμό στο διαδίκτυο, όπως το cosmicJS και το butterCMS , ενώ ήδη πολλές γνωστές πλατφόρμες επεκτείνονται και στη react όπως το wordpress.

5.3 Γραφικές αναπαραστάσεις με ReactJs

Στη javascript υπάρχει η βιβλιοθήκη D3 η οποία επιτρέπει να αναπαραστήσουμε τα εκάστοτε δεδομένα που τροφοδοτούμε σε γραφήματα, ενώ παρέχει και διάφορες μεθόδους για την μορφοποίηση και διαχείριση αυτών. Αυτή τη βιβλιοθήκη μπορεί κάποιος να την συμπεριλάβει στο σύστημά του και έπειτα να τη συνδέσει με τη react , μέσω της βιβλιοθήκης reactD3 , με αποτέλεσμα τελικά τα γραφήματα της D3 να αποτελούν component σε react. Μπορούν να αντιστοιχηθούν οι ανάλογες συναρτήσεις επίσης στη react και να μορφοποιηθούν , ενώ παράλληλα είναι δυνατή η επεξεργασία και τροφοδοσία των δεδομένων που γίνετε σε αυτά τα γραφήματα, τα οποία πηγάζουν από τη βάση δεδομένων, και μεταφέρονται στα components ως prop και έπειτα state.

6. Χαρακτηριστικά της Reactjs

Εφόσον κατανοήσουμε την ανάγκη χρήσης ενός framework όπως η react, μπορούμε να εξετάσουμε τα χαρακτηριστικά της, τα οποία την διαφοροποιούν από άλλες τεχνολογίες ή της δίνουν αυτή τη σύγχρονη αρχιτεκτονική .

Πρώτο και κύριο χαρακτηριστικό είναι η μονόδρομη επικοινωνία των δεδομένων, one-way data-flow. Αυτός ο μηχανισμός ορίζει ότι τα δεδομένα κινούνται μόνο προς τα εμφωλευμένα αντικείμενα και όχι προς τα πίσω, τα οποία δεδομένα που αποστέλονται ονομάζουμε prop, και διαφοροποιούνται με τα δεδομένα του αντικειμένου που προσδιορίζονται ως state. Το εμφωλευμένα αντικείμενο δεν μπορεί να μεταβάλλει τα δεδομένα του γονέα άμεσα, για να γίνει αυτό ο γονέας πρέπει να

περάσει μια συνάρτηση στο παιδί που θα του επιτρέπει να ενημερώνει τον γονέα για την μεταβολή, ο οποίος θα μεταβάλλει τα δεδομένα που τον αφορούν. Ωστόσο επειδή αυτό μπορεί να οδηγήσει σε πολυπλοκότητα κατά την επέκταση μίας εφαρμογής, συνιθίζεται να χρησιμοποιούμε ένα κατάστημα, store, το οποίο ενημερώνουμε όταν θέλουμε να μεταβάλλουμε δεδομένα κάπου στην εφαρμογή μας και αυτό αναλαμβάνει την υλοποίηση των περαιτέρω απαραίτητων διαδικασιών. Αυτό ονομάζεται αρχιτεκτονική flux, και υπάρχουν επεκτάσεις για την react που την υλοποιούνε, με γνωστότερη από αυτές το Redux.

Επόμενο και κυριότερο χαρακτηριστικό της React είναι το εικονικό περιβάλλον DOM, virtual document-object-model, που δημιουργεί πρόσθετα στη μνήμη, και αυτό έπειτα αντιστοιχείζεται με το κανονικό dom που δημιουργείται στον browser. Ουσιαστικά οι αλλαγές συμβαίνουν πρώτα στο virtual dom και έπειτα από σύγκριση με αλγόριθμο, diffing, γίνεται απεικόνιση στο κανονικό dom ότι είναι απαραίτητο, εξοικονομώντας έτσι πόρους. Επίσης η άμεσες αναπαραστάσεις στοιχείων του DOM στη react ονομάζονται element. Η react ορίζει πως το αποτέλεσμα που θα εμφανιστεί, από την συνάρτηση render, θα είναι τύπου element.

Τρίτο χαρακτηριστικό της react είναι η δυνατότητα χρήσης JSX. Η jsx επεκτείνει τον τρόπο σύνταξης και δημιουργίας δεδομένων στη react. Μας επιτρέπει να χρησιμοποιούμε html ετικέτες μέσα σε στην εφαρμογή μας και να μετατρέπονται αυτές απευθείας σε παιδιά της κλάσης γονέα, κρατώντας έτσι το κώδικά μας απλό και αναπαραστατικό . Επιπροσθέτως με την jsx, μπορούμε να αναφερόμαστε σε ιδιότητες του εκάστοτε component μέσα στα html tags. Και αυτό το καταφέρνει διότι δημιουργεί αναπαραστάσεις στη javascript ,για τα html-tags, σε αντικείμενα. Επιπλέον επιτρέπει τη χρήση κλασσικής javascript.

Έτσι μπορούν να υπάρχουν σύνθετα εμφωλευμένα αντικείμενα μέσα στα tags, μπορούν να περάσουν ξεχωριστές ιδιότητες στα αντικείμενα αυτά, επιτρέπεται η χρήση εκφράσεων javascript μέσα σε αγκύλες {} . Επιπλέον επιτρέπει τη χρήση εκφράσεων συνθήκης if - else με χρήση λογικών συντελεστών , ωστόσο συνηθίζεται σε αυτές τις περιπτώσεις να χρησιμοποιούνται μεταβλητές javascript που έχουν πρόσβαση σε μεθόδους javascript και είναι πιο αναπαραστατικές . Επίσης επιτρέπει τη σύνδεση με μεθόδους της κλάσης γονέα μέσω της αναφοράς τους στο παιδί ως prop.

Τελευταίο χαρακτηριστικό της είναι η επέκταση της αρχιτεκτονικής της, ανεξαρτήτως από την HTML, το οποίο οδηγεί στην ευκολία χρήσης της react στο server, το οποίο ονομάζουμε isomorphic react, αλλά και τη δυνατότητα ανάπτυξης εφαρμογών για περιβάλλον javascript και όχι browser,με τη βιβλιοθήκη react native.

6.1 Αρχές σχεδίασης

Οι σχεδιαστές της react έχουν ορίσει συγκεκριμένους άξονες ως βάση για την επέκταση και αναβάθμιση της πλατφόρμας. Αρχικά ο σχεδιασμός της react έχει οριστεί να βασίζεται στο state γενικά, το οποίο περιέχει τις πληροφορίες του component της εφαρμογής, την εκάστοτε στιγμή με μορφή props και state. Αυτό διευκολύνει την διαδικασία της ανάλυσης της ροής των δεδομένων και άρα πρόσθετα και τη διαδικασία της αποσφαλμάτωσης. Τον σκοπό αυτό εξυπηρετεί η εργαλειοθήκη react-developer-tools που παρέχεται για τον εκάστοτε browser και παρέχει τις πληροφορίες των component της εφαρμογής μας κατά την εκτέλεση.

Η σχεδίασή της επίσης βασίζεται στην αρχή της σύνθεσης, ότι δηλαδή η εφαρμογή είναι ένα component που αποτελείται από άλλα component και αυτά με τη σειρά τους από άλλα. Έτσι προκύπτουν επαναχρησιμοποιήσιμα component. Τη λογική αυτή ενισχύει το γεγονός ότι τα component βασίζονται στις ιδιότητες και στις μεθόδους του κύκλου ζωής τους, το οποίο αποτελεί ουσιώδες χαρακτηριστικό των αντικειμένων σε react.

Με τη χρήση αυτών των μεθόδων κάποιος μπορεί να παρέμβει στο εκάστοτε component και να το ανανεώσει ή να το μορφοποιήσει ανά πάσα στιγμή. Επίσης μπορεί να ελέγξει τις τιμές τους και να μορφοποιήσει κάποιο συσχετιζόμενο αντικείμενο. Επιπρόσθετα όλες οι μέθοδοι και τα χαρακτηριστικά των κλάσεων γίνονται διαθέσιμα για πρόσβαση από το API με αναπαραστατικά ονόματα , όπως οι μέθοδοι κύκλου ζωής.

Έτσι οι σχεδιαστές της βιβλιοθήκης αυτής δίνουν έμφαση στη ευχρηστία της γλώσσας, ενώ ταυτόχρονα προσπαθούν να αναβαθμίζουν την διεπαφή με νέες απλοποιήσεις και καινούργιες λειτουργίες.

7. Hello world σε react

Η δημιουργία μιας εφαρμογής με τη χρήση της reactJs μπορεί να είναι εύκολη και να μην χρησιμοποιούνται τεχνολογίες εκτός από τις άμεσα απαραίτητες. Οι απαραίτητες τεχνολογίες είναι οι εξής:

- η βιβλιοθήκη reactJs, η οποία μας παρέχει τη διεπαφή για τις κλάσσεις της reactJs που αποτελούν τα δομικά συστατικά της εφαρμογής μας.
- η βιβλιοθήκη reactDOM η οποία αναλαμβάνει τη δημιουργία αναπαράστασης του εκάστοτε στιγμιοτύπου στο χρήστη.
- η βιβλιοθήκη babel, ή κάποια ανάλογη όπως η webpack, η οποίες παρέχουν λειτουργίες που επεκτείνουν την απλή javascript , όπως η jsx, επιτρέπουν την χρήση αρχείων για παραγωγή κλάσσεων, και σύνθετα συντακτικά για τη javascript ,όπως ecma-script6. Αυτή η τεχνολογία συνήθως εγκαθίσταται στο Server, ο οποίος τη χρησιμοποιεί για να δημιουργήσει το επιστρεφόμενο αποτέλεσμα.

Όμως μπορεί να δημιουργηθεί ένα αρχείο html, το οποίο θα συμπεριλαμβάνει αυτές της απαραίτητες τεχνολογίες και θα τρέχει αποκλειστικά στο browser.

Παράδειγμα αρχείου με ανάλυση, index.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Simplest Example React</title>
    <script
src="https://unpkg.com/react@16/umd/react.development.js"></script
>
    <script
src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
```

```
</script>
    <script
src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></scr
ipt>
  </head>
  <body>
    <div id="app"><h1>app crashed</h1></div>


  </body>
<script type="text/babel">
      ReactDOM.render(<h1>Hello,
world!</h1>,document.getElementById('app'));
    </script>
</html>
```

Το αρχείο ξεκινάει όπως ένα οποιοδήποτε αρχείο html,

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Simplest Example React</title>
```

έπειτα εμπερειέχονται οι βιβλιοθήκες react, reactDom, και babel (έκδοση που τρέχει στο browser του χρήστη).

```
<script
src="https://unpkg.com/react@16/umd/react.development.js"></script
>
    <script
src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>
    <script
src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></scr
ipt>
```

Μετά υπάρχει το μοναδικό html element που θα φιλοξενεί την εφαρμογή και παίρνει αναγνωριστικό ώστε να μπορεί να γίνει αναφορά.

```
<div id="app"><h1>app crashed</h1></div>
```

Έπειτα έρχεται το script, το οποίο δηλώνεται ως "text/babel" και αυτό για να χρησιμοποιηθεί η βιβλιοθήκη babel αντί για απλή javascript, η οποία δεν επιτρέπει τη χρήση react. Μέσα στο script αυτό καλείται η συνάρτηση render από το από τη κλάσση reactDOM με δύο παραμέτρους, πρώτα ένα element,και μετά τη θέση στην οποία θέλουμε να το τοποθετήσουμε στο αρχείο μας.

```
<script type="text/babel">
      ReactDOM.render(<h1>Hello,
world!</h1>,document.getElementById('app'));
    </script>
```

Στην περίπτωση που τρέχει σωστά θα εμφανίσει, hello-world, ενώ αν αποτύχει να συμπεριληφθεί κάποια βιβλιοθήκη, λόγω ορθογραφικού λάθους για παράδειγμα, δεν θα τρέξει το script και θα εμφανιστεί το αρχικό κείμενο του element , app crashed.

Στην περίπτωση που η εφαρμογή προέκυπτε από server και δεν υπήρχε στο αρχείο θα ήταν απαραίτητη και η χρήση της βιβλιοθήκης webpack, που επιτρέπει να συμπεριληφθούν οι απαραίτητες τεχνολογίες από τον server στο αρχείο που θα αποσταλεί στον client, την εισαγωγή και εξαγωγή κλάσεων στην εφαρμογή από αρχεία, ενώ επίσης την παραμετροποίηση των βιβλιοθηκών αυτών και ρυθμίσεις του εξαγόμενου αρχείου, στο αρχείο ρυθμίσεων json package.json .

8. Η εντολή create-react-app

Πολλές φορές αφού εγκατασταθεί το λογισμικό server , έπεται συνήθως η επέκτασή του με τις βιβλιοθήκες που θέλουμε να χρησιμοποιήσουμε .

Στην περίπτωση που χρησιμοποιούμε react, πρέπει να συμπεριλάβουμε αρχικά τη react. Έπειτα δεν πρέπει να παραλείψουμε τεχνολογίες χρειάζονται για να ενεργοποιηθούν οι βασικές της δυνατότητες. Τέτοιες τεχνολογίες είναι η babel, που επεκτείνει το συντακτικό της javascript, και η webpack η οποία μπορεί να ενώνει

κλάσσεις από αρχεία. Επιπλέον χρειάζεται η παραμετροποίηση του server ώστε να μπορεί να ξεκινάει και να αποκρίνεται ανάλογο, για το οποίο επίσης πρέπει να συμπεριληφούν βιβλιοθήκες.

Τη λύση σε αυτό το πρόβλημα δίνει η βιβλιοθήκη create-react-app. Αυτή μπορεί να εγκατασταθεί σε ένα server nodeJs, και έπειτα θα συμπεριληφθούν όλα τα απαραίτητα εργαλεία που μπορεί να χρειάζεται ένας server που παράγει ιστοσελίδες ή εφαρμογές react. Επίσης δημιουργούνται default αρχεία και ρυθμίσεις για να επιβεβαιωθεί η λειτουργία τους, τα οποία στην πορεία αντικαθίστανται με αυτά που χρειάζεται η εφαρμογή.

```
npm i create-react-app
npx create-react-app reactProject
```

9. Components

Τα component αποτελούν τα βασικά δομικά συστατικά της εφαρμογής. Έχουν δικές τους παραμέτρους και μπορούν να δέχονται παραμέτρους από τους γονείς τους. Μπορούν να περιλαμβάνουν άλλα component και να τα συνδυάζουν, έτσι ο σωστός σχεδιασμός αυτών οδηγεί στην επαναχρησιμοποίησή τους. Οι παράμετροι ή ιδιότητες του component χωρίζονται σε props και σε state. Τα props περιλαμβάνουν τις παραμέτρους που έρχονται από τον γονέα και δεν μπορεί να τις αλλάξει άμεσα το παιδί, ενώ το state περιλαμβάνει πληροφορίες αποκλειστικά δικές του, τις οποίες μπορεί να μεταβάλλει και να τις περάσει σε ένα παιδί component με την σειρά του ως props.

Η αναφορά στις εκάστοτε λίστες παραμέτρων γίνετε με τις λεξεις κλειδιά this.state και this.props , ακολουθούμενες από το όνομα της παραμέτρου που θέλουμε να πάρουμε. Ενώ για να μεταβληθεί μία παράμετρος στο state του component η εντολή που παρέχεται είναι η this.setState({}); ενώ μέσα στις αγκίλες περιλαμβάνονται τα ονόματα των παραμέτρων ακολουθούμενα απο τις αντίστοιχες τιμές, και χωρισμένα μεταξύ τους με κόμμα.

Τα component επίσης εμπεριέχουν μεθόδους που αναπαριστούν σε ποιο κομμάτι της εκτέλεσης βρίσκεται το component σε σχέση με την εφαρμογή, για παράδειγμα όταν έχει μόλις φορτώσει , όταν είναι έτοιμο να ανανεωθεί , ή αφού έχει ανανεωθεί. Επιτρέπεται η επέκταση αυτών των συναρτήσεων διότι παρέχονται από το api της react για τα component, ενώ αποκαλείται κύκλος ζωής του component, component life-cycle.

Το api για τα components επιπλέον παρέχει την έννοια του δομητή, με την λέξη κλειδί constructor. Πρόσθετα, παρέχει την ιδιότητα render() που επιστρέφει τα εμφανίσιμα δεδομένα του component στην εφαρμογή , ενώ ταυτόχρονα πρέπει να αποφεύγει να τροποποιεί μέσα εδώ, άμεσα τα δεδομένα του component, διότι οδηγεί σε πολυπλοκότητα και ενδεχόμενα λάθη.

Διαφοροποιούνται ως κλάσσεις από τα απλά element, τα οποία είναι πιο ελαφριά, καθώς δεν περιλαμβάνουν όλες αυτές ιδιότητες. Επιπλέον δεν έχουν κύκλο ζωής όπως τα component. Το αποτέλεσμα του render περιγράφεται επίσης ως element, που περιλαμβάνει το στιγμιότυπου του component για εμφάνιση.

9.1 Component life-cycle

Ο κύκλος ζωής του component χωρίζεται σε δύο κατηγορίες, η μία αφορά την αρχική φόρτωση του component, mount, και η άλλη την ανανέωση αυτού, update. Πιο συγκεκριμένα παρέχονται οι παρακάτω μέθοδοι:

- *constructor*(): αρχικοποιεί τις ιδιότητες ή state του component.
- *componentWillMount*(): εκτελείται πριν φορτωθεί το component στο DOM.
- *render*()*:* επιστρέφει τη γραφική αναπαράσταση του component σε μορφή element, για το εκάστοτε στιγμιότυπο.
- *componentDidMount*(): εκτελείται αφού ολοκληρώσει η φόρτωση του component στο DOM.
- *componentWillReceiveProps*(): εκτελείται όταν το component αλλάξει props. Μπορεί να χρησιμοποιηθεί για να ανανεωθεί αναλόγως και το state ενός υπάρχοντως component σε σχέση με τα props που μόλις άλλαξαν.

- *shouldComponentUpdate*(): ορίζει πότε θα χρειαστεί το component να κάνει update και μπορεί να καλεί συνάρτηση.
- *componentWillUpdate*(): εκτελείτε πριν ανανεωθεί ένα υφιστάμενο component στο DOM
- *componentDidUpdate*(): εκτελείτε αφού ανανεωθεί ένα υφιστάμενο component στο DOM
- *componentWillUnmount*(): εκτελείται αφού αφαιρεθεί ένα component από το DOM

10. Routing

Η react αυτό που μας παρέχει είναι ένα περιβάλλον γραφικής διεπαφής στο οποίο είναι εύκολο να δημιουργήσουμε μία εφαρμογή μέσα σε μία σελίδα. Μπορεί ωστόσο να χρειάζεται να καταλαβαίνει η εφαρμογή τη διεύθυνση που εισάγει ο χρήστης και να τον καθοδηγεί ανάλογα, τότε πρέπει να συμπεριληφθεί μία βιβλιοθήκη που αναλαμβάνει το κομμάτι το οποίο λέγεται routing. Για την react η βιβλιοθήκη αυτή ονομάζεται react-router. Με την εγκατάστασή της επιτρέπεται να δημιουργηθούν οι συνδέσεις ανάμεσα στα component και στα url που τους αντιστοιχούν.

Επιπροσθέτως αυτά τα component μπορούν να περιέχουν άλλα εμφωλευμένα component, ενώ το κάθε ένα από αυτά εμφανίζεται μόνο εάν περιέχεται στο αντίστοιχο route. Η βιβλιοθήκη react-router επιπλέον παρέχει αντικείμενα τύπου link τα οποία επιτρέπουν την πλοήγηση στα διαθέσιμα route και ανανεώνουν τη υφιστάμενη διεύθυνση. Επίσης παρέχεται η κλάση history η οποία κρατάει το ιστορικό πλοήγησης της εφαρμογής.

Ο τρόπος με τον οποίο δουλεύει δουλεύει το router επιτρέπει να χτίσουμε ένα template ή layout με τα κύρια components που επιθυμούμε να εμφανίζονται ανά την εκάστοτε διεύθυνση, και αυτό είναι εφικτό διότι τα components που επιστρέφουν από τα routes μπορούν να εμφωλευτούν το ένα στο άλλο, αλλά επίσης και τελικά να

συνδυαστούν. Τέλος μπορούν να οριστούν μονοπάτια της εφαρμογής προστατευόμενα, περιορίζοντας έτσι την πρόσβαση των χρηστών σε αυτά.

Παράδειγμα του Router που χρησιμοποιείται στην εφαρμογή

```
import { BrowserRouter as Router, Route } from 'react-router-dom';

ReactDOM.render((
  <Router>
    <Route path="/" component={App} />
     </Router >
), document.getElementById('root'));
```

11. Redux

Προκειμένου να εξελιχθεί ακόμα περισσότερο η αρχιτεκτονική της διεπαφής μίας εφαρμογής, μπορεί να χρησιμοποιηθούν πλατφόρμες που διαχωρίζουν τις κλάσσεις της εφαρμογής ανάλογα με τον σκοπό που εξυπηρετούν, προσδιορίζοντας έτσι καλύτερα το σκοπό του κάθε αντικειμένου ξεχωριστά.

Για το λόγο αυτό δημιουργήθηκε το πρότυπο flux το οποίο περιγράφει την αρχιτεκτονική της διεπαφής πλατφόρμας. Αυτό ορίζει πως τα δεδομένα θα κινούνται προς μία κατεύθυνση στην εφαρμογή, και πως η εφαρμογή απαρτίζεται από τρία αντικείμενα που επικοινωνούν μεταξύ τους. Αυτά είναι τα dispatcher, store, view. Αυτό διαφοροποιείται από το μοντέλο MVC. Ωστόσο υπάρχουν και οι controller οι οποίοι ελέγχουν το τί πρέπει να εμφανιστεί στο view. Όταν ο χρήστης επικοινωνήσει με τη διεπαφή, μέσω του view, ενεργοποιείται ένα action το οποίο θα δώσει οδηγίες με τη σειρά του στο dispatcher. Αυτός με τη σειρά του θα ανανεώσει το state του εκάστοτε store ανάλογα με το action που το ενεργοποίησε. Επίσης είναι υπεύθυνος

να συντοντίζει τα δεδομένα στα εκάστοτε store. Οι controller, ακούνε για τις αλλαγές στο store και αναλαμβάνουν να ενημερώσουν αναλόγως το view, ονομάζονται και view- controller.

Τη φιλοσοφία της αρχιτεκτονικής flux στη react επεκτείνει η πλατφόρμα redux, για την οποία υπάρχουν βιβλιοθήκες σύνδεσης με την react, ενώ κανονικά ορίζεται ανεξάρτητα από την πλατφόρμα που την εφαρμόζει.

Και το redux υλοποιεί την έννοια του store για μια εφαρμογή σε react. Αυτό περιλαμβάνει το state της εφαρμογής μας την εκάστοτε στιγμή και είναι ικανό να τροφοδοτεί τα κατάλληλα δεδομένα προς παρουσίαση. Έτσι το store αποτελεί την μοναδική πηγή αλήθειας για την εφαρμογή μας.

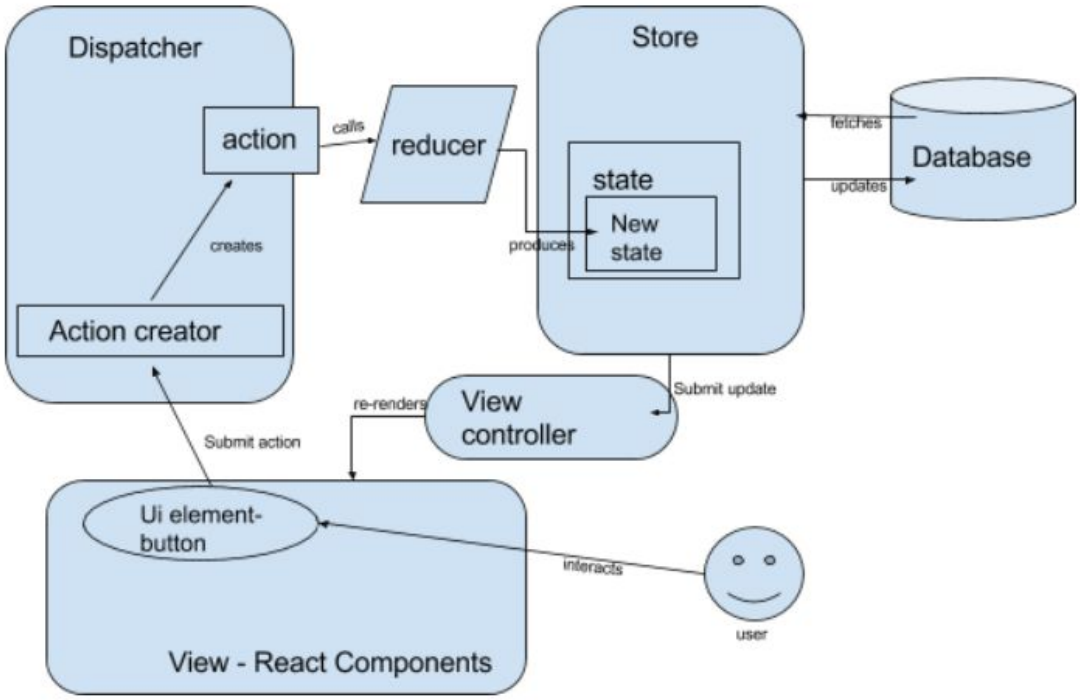
Το state που περιλαμβάνεται στο store δεν μπορεί να μεταβάλλεται άμεσα, πρέπει να ενεργοποιηθεί ο dispatcher με το ανάλογο action για να συμβεί αυτό. Για να ολοκληρωθεί η αλλαγή στο store ωστόσο, τροφοδοτείται το state από το εκάστοτε store με την αντίστοιχη action στο dispatcher, και αυτό οδηγεί στην εκτέλεση συγκεκριμένων reducer, ώστε τελικά να συμβεί μια ανανέωση στο state.

Οι reducer χειρίζονται τα action του dispatcher, και περιγράφονται ως αγνές συναρτήσης, pure function, το οποίο σημαίνει ότι δεν αλλάζουν τίποτα στην εφαρμογή και το αποτέλεσμα της εκτέλεσης τους είναι η επιστροφή του νέου state. Καλώντας έναν reducer με τις ίδιες παραμέτρους , αναμένεται να επιστραφεί πάντα το ίδιο αποτέλεσμα. Οι reducer αποτελούν χαρακτηριστικό του redux.

Οπότε το νέο state που επιστρέφεται από τους reducer θα αντικαταστήσει το παλιό στο store και αυτός είναι ο μόνος τρόπος να μεταβληθεί το state του store κατά το redux.

Επιπρόσθετα για να υλοποιήσουμε το view , στο redux , δημιουργούμε τα συστατικά που χρειάζονται για την εμφάνιση τους μέσω της react, και εμπεριέχονται από ένα component το οποίο συνδέεται με το store και χρησιμοποιεί για εμφάνιση τις ανάλογες κλάσσεις.

Το component αυτό που συνδέει τις λειτουργίες των component, εμφάνισης και συμπεριφοράς, ονομάζεται container στο redux, ενώ η έννοιά του περιλαμβάνεται και στη react με το όνομα High-order-component. Τα HOC αναλαμβάνουν να συνδυάσουν τις λειτουργίες και τις μεθόδους που χρειάζονται από ένα αντικείμενο για την γραφική του αναπαράσταση.



Λόγω του ότι με το redux επέρχεται περαιτέρω διάσπαση των components μας σε υπομέρους στοιχεία βάση των λειτουργιών, προκύπτει μία καλύτερα οργανωμένη εφαρμόγη με συστατικά πιο συγκεκριμένου και ξεκάθαρου σκοπού.

12. React Native

Η react native αποτελεί ένα ξεχωριστό κομμάτι της reactJs το οποίο επεκτείνει την εφαρμογή της , σε προγράμματα που τρέχουν εκτός φυλλομετρητή . Αυτό το πετυχαίνει διότι χρησιμοποιεί μια βιβλιοθήκη η οποία αναπαριστά το δέντρο δεδομένων που υπάρχουν στο virtual dom αντί για το dom του browser, στου αντίστοιχους ελεγκτές όψεις του λειτουργικού, δηλαδή το UIView για το ios και το android.view για το android.

Συνήθως όμως ο σχεδιασμός της όψης για κινητές συσκευές διαφέρει από αυτό του φυλλομετρητή, διότι στις κινητές συσκευές ο χώρος παρουσίασης είναι μικρός και η πληροφορία τίνει να είναι πιο λιτά δομημένη. Επίσης μερικά ακόμη χαρακτηριστικά διαφοροποιούνται, όπως για παράδειγμα η έλλειψη μπάρας συνδέσμου που υπάρχει στους φυλλομετρητές, ενώ η λειτουργία αυτή υλοποιείται διαφορετικά.

Εκτός από αυτές τις διαφορές, στο native περιβάλλον δεν μπορούμε να χρησιμοποιήσουμε html tags , τα οποία αναγνωρίζονται από τους φυλλομετρητές , ενώ τη θέση αυτών παίρνουν components που αναγνωρίζονται από τα native περιβάλλοντα και έχουν δημιουργηθεί με αυτό το σκοπό.

Με αυτές τις διαφορές κατά νου, μπορεί κάποιος να προχωρήσει στη δημιουργία μιας εφαρμογής με react native, βάση της αρχιτεκτονικής που διέπεται από το μοντέλο, με παρόμοια προσέγγιση με την διαδικτυακή εφαρμογή του browser, αφού όλες οι λειτουργίες της reactJs, έχουν τροποποιηθεί για να χρησιμοποιούνται και από τη react native.

13. Isomorphic react

Μέχρι τώρα μελετήθηκαν οι περιπτώσεις που ο server στέλνει την εκάστοτε εφαρμογή στο χρήση για εκτέλεση. Είναι δυνατόν ωστόσο να δημιουργηθεί αυτή η εφαρμογή και να εκτελείται στο server , ενώ θα ενημερώνει ανάλογα τον χρήστη βάση του στιγμιότυπου που επιθυμεί, υπάρχει δηλαδή συγχρονισμός . Το γεγονός ότι μία εφαρμογή υπάρχει και στο server και στο client σε δύο ταυτόχρονα μέρη και αυτά συγχρονίζονται μεταξύ τους ονομάζεται ισομορφισμός, isomorphism, ενώ αυτός ο τρόπος υλοποίησης ονομάζεται isomorphic.

Στην isomorphic react χρησιμοποιείται η εντολή ReactDOM.renderToString(), η οποία παίρνει ως παράμετρο ένα component και επιστρέφει το αποτέλεσμα της εκτέλεσής του σε κείμενο το οποίο θα σταλεί στο client ως η html προς εκτέλεση.

Έπειτα στο client θα γίνει άμεσα αναπαράσταση της εφαρμογής όπως αυτή υπάρχει στο server. Έτσι γλιτώνει πόρους ο client .

Ένας σημαντικός λόγος που προτιμάτε η isomorphic react είναι στη περίπτωση που η βάση δεδομένων υπάρχει στο ίδιο μέρος με τον server. Όταν η εφαρμογή τρέχει πρώτα στο server μπορεί αυτό να συλλέξει τα δεδομένα και να διαλέξει μόνο αυτά που χρειάζεται να αποστείλει στην απάντηση, γλιτώνοντας έτσι σημαντικά κίνηση των δεδομένων στο δίκτυο.

Η αρχιτεκτονική redux υποστηρίζεται επίσης στην isomorphic react, αφού χρειάζεται ένα μόνο store για να κάνει τις αλλαγές στο state και αυτό υπάρχει στο server και είναι υπεύθυνο για τα δεδομένα που θα προωθήσει στις εκάστοτε όψεις.

Οι όψεις επίσης παράγονται προτίστως στον server και αποστέλλονται έπειτα στον client έτοιμες να εμφανιστούν αναλόγως. Οι μορφοποιήσεις που μπορεί να έχουν, π.χ. css υλοποιούνται επίσης στον server, διευκολύντας έτσι τη διαδικασία αναπαράστασης στο φυλλομετρητή.

Τέλος περιορίζονται οι μέθοδοι που έχουν πρόσβαση στα δεδομένα να τρέχουν μόνο μέσα από το server χωρίς την παρέμβαση του χρήστη, αφήνωντας έτσι περισσότερα περιθώρια ελέγχου της ασφάλειας της εφαρμογής.

14. Styling components

Η μορφοποίηση της διεπαφής που εμφανίζεται, σίγουρα δεν επηρεάζει το μοντέλο ή τα δεδομένα άμεσα, αλλά είναι ο παράγοντας που κάνει την εφαρμογή πιο ελκυστική στο χρήστη. Οπότε αυτό το κομμάτι οφείλει να υλοποιείται όσο το δυνατό πιο απλοϊκά και αναπαραστατικά είναι δυνατόν, ώστε να είναι έυκολη η μετατροπή και συντήρησή του, βάση του σχεδιαστικού προτύπου.

Για το σκοπό αυτό υπάρχουν μέθοδοι αλλά και τεχνολογίες μορφοποίησης, με γνωστότερη και κυριότερη από αυτές να είναι η CSS (cascading style-sheets). Οι οδηγίες σε μορφή css μπορούν να περιλαμβάνονται και από αρχεία ή να δηλώνονται επί τόπου, inline, και επιδρούν στην διεπαφή αφού φορτωθεί το περιβάλλον dom. Για

να αναφερθούμε σε ένα component από το css μπορούμε να του δώσουμε κλάση, με τη λέξη κλειδί classname ως παράμετρο.

Styling στο αρχείο css με χρήση class name

```
.user-profile img {
  border-radius: 50%;
  max-width: 100%;
}

.card img{
  width: 100%;
  max-height: 120px;
}
```

Συχνά όμως προτιμάται και η αποθήκευση των παραμέτρων αυτών σε πίνακα javascript και η μορφοποίηση των component βάση αυτών. Αυτό συμβαίνει ώστε να υπάρχει σαφέστερη αναπαράσταση των component στα αρχεία μας.

Επιπροσθέτως πλατφόρμες που αναλαμβάνουν την γραφική αναπαράσταση των component, όπως το material-ui, συνήθως παρέχουν έτοιμες παραμέτρους για την μορφοποίηση της εμφάνισης αυτών.

Styling με χρήση μεταβλητών material

```
const styles = {
    errorStyle: {
      color: orange500,
    },
    floatingLabelStyle: {
      color: blue500,
    },
  };
```

```
<TextField
                  floatingLabelText="Created by:"
                  disabled={true}
                  floatingLabelStyle={styles.floatingLabelStyle}
              />
```

Ενώ άλλες, όπως ή η βιβλιοθήκη styled-components χρησιμοποιούν css για να μορφοποιηθούν τα αντίστοιχα αντικείμενα μέσα στις κλάσεις . Οι τιμές για τις παραμέτρους αυτές μπορούν να αποθηκευτούν σε μεταβλητές ή πίνακες, καθιστώντας τες πιο κατανοητές.

Styling με χρήση μεταβλητών inline

```
const cardStyles = {
    width : '32%',
    margin: '1% 1% 0% 0',
    float: 'left',
    minHeight: 120,
  };
```

```
<Card
    style={cardStyles}
    >
```

Εκτός από τα απλά αρχεία css, μπορούν να χρησιμοποιηθούν προηγμένες τεχνολογίες δημιουργίας css, οι css-preprocessors, οι οποίοι επεκτείνουν τη σύνταξη και λειτουργία της απλής css, αλλά τελικά παράγεται ένα απλό αρχείο css, εκτελέσιμο από τον φυλλομετρητή. Οι μορφοποιήσεις μέσω javascript επιπλέον υλοποιούνται τελικά με css.

Ωστόσο δεν χρειάζεται κάποιος να περιοριστεί σε μία από αυτές τις τεχνολογίες και συνήθως συνδυάζονται. Τα αρχεία css φέρουν πληροφορίες των γενικών μορφοποιήσεων της εφαρμογής, όπως οι πληροφορίες για τις γραμματοσειρές και για τη συμπεριφορά της εφαρμογής στα διάφορα μεγέθη παραθύρων, ενώ τα ξεχωριστά component περιέχουν τις πληροφορίες αυτές μέσα στις κλάσσεις τους.

15. Σύγκριση της ReactJs με άλλα framework

Η reactJs διαφοροποιείται από άλλες τεχνολογίες διότι έχει μοναδικά χαρακτηριστικλα όπως να αποτελεί σκελετό για την όψη της εφαρμογής, ενώ για να επεκταθεί στο μοντέλο χρειάζεται ο συνδυασμός του με το redux. Επιπλέον η προσέγγιση της reactJs στην εφαρμογή γίνετε βάση των component, component-based. Επιπλέον χαρακτηριστικό της είναι η σύνταξη jsx, που κάνει την ανάπτυξη ευκολότερη για τον προγραμματιστή σε πολύπλοκες διεπαφές. Η απόδοση της reactJs επίσης είναι πολύ καλή, διότι οι αλλάγες συμβαίνουν πρώτα στο virtual DOM και μετά ανανεώνεται μόνο ότι είναι απαραίτητο στο DOM του browser. Παρέχεται api, με την ίδια αρχιτεκτονική το οποίο δημιουργεί εφαρμογή για native περιβάλλον javascript, όπως το android. Τελικά η λογική της react είναι πολύ απλή, αφού τα component βασίζονται στο state και στα prop, τα οποία χαρακτηρίζουν και την εφαρμογή.

Τα προτερήματα της reactJs ποικίλουν περισσότερο ωστόσο σε σχέση με παρεμφερής τεχνολογίες.

Σε σχέση με την angular, αναλαμβάνουμε εμείς την ανάπτυξη του μοντέλου, ενώ η angular παρέχει όλο το μοντέλο MVC, με υλοποιημένες μεθόδους. Επίσης στην angular μπορούμε να έχουμε αμφίδρομη επικοινωνία των δεδομένων ανάμεσα στα component με τις μεθόδους αυτές , ενώ στη react η επικοινωνία είναι μονόδρομη και οι αλλαγές συμβαίνουν στη βάση του μοντέλου του redux. Στην angular ορίζεται η έννοια του template και χρειάζεται υλοποίηση, ενώ στη react το template είναι μία λογική που υλοποιείται πρόσθετα στη πλατφόρμα. Τελικά σε πιο σύνθετες εφαρμογές, η reactJs θα παράγει πιο απλή σελίδα σε σχέση με την angular διότι οι αλλαγές συμβαίνουν στο virtual DOM πριν εμφανιστούν, εξοικονώντας έτσι πόρους του browser.

Επιπλέον διαφέρει σε σχέση με άλλες βιβλιοθήκες javascript που αναλαμβάνουν αποκλειστικά την εμφάνιση της διεπαφής, όπως το bootstrap και η jquery. Στην react τα συστατικά της διεπαφής προκύπτουν από τα αντικείμενα, ενώ το bootstrap και η jquery εισάγουν κλάσσεις, και επεκτείνουν τις λειτουργίες ή

μορφοποιήσεις της ήδη υπάρχουσας σελίδας. Για το λόγο αυτό μπορούν να συνδυαστούν αυτές οι βιβλιοθήκες, η react με το bootstrap για παράδειγμα, ή να χρησιμοποιηθεί βιβλιοθήκη που παρέχει τα ουσιώδη γραφικά components μιας διεπαφής, κατευθείαν σε react όπως το material-ui, με παραμέτρους και μεθόδους, ανάλογα με την εκάστοτε περίπτωση χρήσης του component.

16. Debugging

Η αποσφαλμάτωση αποτελεί σημαντικό κομμάτι της ανάπτυξης όλων των εφαρμογών. Στις διαδικτυακές εφαρμογές η αποσφαλμάτωση γίνει μέσω τις κονσόλας του browser και της κονσόλας του server. Έπειτα μπορεί να μελετηθεί το περιβάλλον DOM, που αποτελείται από όλη την html που φαίνεται στο φυλλομετρητή. Αυτό γίνετε με τη χρήση του εργαλείου inspector που παρέχει ο κάθε φυλλομετρητής.

Για την βιβλιοθήκη reactjs ωστόσο, παρέχεται ειδική εργαλειοθήκη για τους φυλλομετρητές με σκοπό την αποσφαλμάτωση, η react developer tools. Αυτή η εργαλειοθήκη μας επιτρέπει να μελετήσουμε σε βάθος και να αναλύσουμε εφαρμογές που τρέχουν react. Μας εμφανίζει το virtual Dom της εφαρμογής μας , δηλαδή όλα τα component και τα ενδεχόμενα παιδιά τους, πριν πάνε στο DOM του φυλλομετρητή. Επίσης με αυτό το εργαλείο μπορούμε να παρατηρούμε διαρκώς τις μεταβολλές στα props και state των component , διευκολύνοντάς έτσι τον προγραμματιστή στη διαδικάσια εντοπισμού σφαλμάτων.

Επιπλέον για περαιτέρω έλεγχο του συστήματός μας, υπάρχει η βιβλιοθήκη Jest. Αυτή μας επιτρέπει να κρατάμε τιμές, ανά των κύκλο ζωής των component και να αποθηκεύουμε τις πληροφορίες που χρειαζόμαστε για έλεγχο σε αρχεία. Η βιβλιοθήκη jest εισάγει την έννοια του unit-testing στην react.

17. Ανάλυση project-template-functions

Το project που μελετήθηκε βασίζεται στην αρχιτεκτονική mvc, ωστόσο δεν χρησιμοποιήθηκε η γνωστή πλατφόρμα redux , ενώ είναι δυνατό αυτό να προστεθεί μελλοντικά. Την πηγή των δεδομένων στην πλατφόρμα μας, αποτελεί το firebase. Υπάρχουν τα τοπικά δεδομένα αποθηκευμένα στα state και props των component αλλά κατ'εντολή του χρήστη γίνετε ανανέωση στη βάση και έπειτα και στη διεπαφή που εμφανίζεται.

17.1 Περιγραφή Εφαρμογής

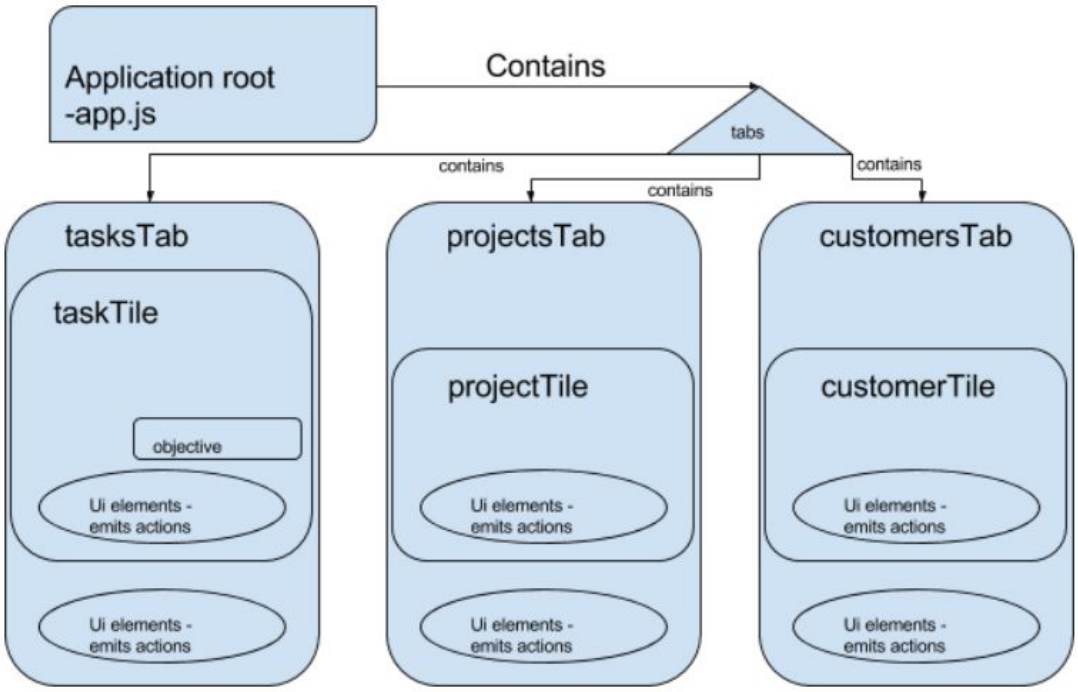
Όταν ο χρήστης επισκεφτεί την εφαρμογή, μέσω του URL της στο φυλλομετρητή, ο server που ανταποκρίνεται θα δημιουργήσει ένα στιγμιότυπο της εφαρμογής σε πακέτο, από τα αρχεία και τις κλάσσεις που χρειάζονται και θα το στείλει ως απόκριση στο χρήστη. Η διαχείριση των url γίνετε με την τεχνολογία routing η οποία μας επιτρέπει να ορίζουμε ποια component, απλά ή σύνθετα, θα ανταποκρινόνται στα εκάστοτε link. Συγκεκριμένα όταν ο χρήστης επισκέπτεται το server αυτόματα αποστέλλει μία απάντηση με το component που περιέχει την εφαρμογή. Αυτό μπορεί να επεκταθεί χωρίζοντας την εφαρμογή σε διαφορετικούς συνδέσμους ανά component. Επίσης αυτή η διαδικασία ονομάζεται routing, ενώ η βιβλιοθήκη που χρειάζεται να συμπεριληφθεί react-router. Για να γίνει η αλλαγή url, παρέχεται η κλάση link που αντιπροσωπεύει σύνδεσμο.

Κατά την μετάβαση του χρήστη στην αρχική σελίδα, εμφανίζονται η μπάρα πλοήγησης, με τον τίτλο της εφαρμογής και το κουμπί που επιτρέπει την είσοδο χρήστη , ενώ στο σώμα της εφαρμογής εμφανίζει μήνυμα που προτρέπει τον χρήστη να συνδεθεί και τον ενημερώνει για τον σκοπό της εφαρμογής.

Επόμενο βήμα είναι ο χρήστης να συνδεθεί με την εφαρμογή, αυτό επιτυγχάνεται με χρήση του δικτύου της google και μεθόδων που παρέχονται από το firebase api. Κατά την σύνδεση το μόνο που επιστρέφεται είναι ένα token το οποίο

επαληθεύει τον χρήστη από την google, χωρίς να υπάρχει μεσολάβηση της εφαρμογής στην διαδικασία ταυτοποίησης με την google.

Με τον χρήστη να έχει ταυτοποιηθεί εμφανίζεται η κύρια διεπαφή της εφαρμογής, η οποία περιλαμβάνει τρεις υποσελίδες, που υλοποιούνται ως tabs.Το κάθε ένα από αυτά τα tabs εμφανίζει τις εκάστοτε λίστες δεδομένων στον χρήστη, αυτές είναι τα tasks, τα projects και οι πελάτες - customers. Σε κάθε μία από αυτές τις λίστες δίνετε η δυνατότητα δημιουργίας καινούργιου αντικειμένου με ένα κουμπί πάνω δεξιά από τη λίστα το οποίο ανοίγει ένα παράθυρο διαλόγου το οποίο μας επιτρέπει να αρχικοποιήσουμε το εκάστοτε αντικείμενο προς δημιουργία. Επιπλέον πάνω αριστερά από τη λίστα εμφανίζονται τρεις επιλογές για φιλτράρισμα των δεδομένων που εμφανίζονται, ως προς τη σχέση που έχουν αυτά με τον χρήστη.



17.2 Οι κλάσσεις του template

Η κάθε λίστα αποτελείται από τα εκάστοτε παιδιά αντικείμενα, τα οποία αντιστοιχίζονται με τα αντικείμενα της βάσης. Αυτά εμφανίζουν αρχικά τις βασικές πληροφορίες, μπορούν κατ'επιλογή να εμφανίσουν όλες τις πληροφορίες τους, ενώ επίσης στην περίπτωση που δημιουργός τους είναι ο χρήστης, παρέχεται η επιλογή για τροποποίηση ή διαγραφή τους. Οι επιλογές που παρεχόνται σε κάθε τύπο και οι σχέσεις μεταξύ τους είναι διαφορετικές, αφού υλοποιήθηκαν για διαφορετικούς σκοπούς.

Έτσι στο κατάλογο αρχείων συναντιούνται τα επομένα αρχείο που αποτελούν και τις κλάσσεις των αντικειμένων:

```
root/src/
App.js
App.css
TasksTab.js
TaskTile.js
ProjectsTab.js
ProjectTile.js
CustomersTab.js
CustomerTile.js
Objective.js
Firebase.js
index.js
```

Επίσης έχουμε τους φακέλους

```
root/node_modules/
```

Και

```
root/public/
```

Όπου ο ένας περιέχει τις απαραίτητες βιβλιοθήκες που χρειάζεται ο server για εκτέλεση, όπως οι εγκατεστημένες react και react dom, ενώ ο άλλος το πρωτεύων

αρχείο index.html το οποίο φέρει το στοιχείο στο οποίο θα φορτωθεί η εφαρμογή μας.
Η κλασση app είναι το κυρίως container της εφαρμογής και περιέχει ένα αντικείμενο τύπου Tabs
Το οποίο γίνετε διαθέσιμο από τη βιβλιοθήκη material-ui

```
import {Tabs, Tab} from 'material-ui/Tabs';
```

Το αντικείμενο tabs με τη σειρά του περιέχει tab τα οποία επιστρέφουν τα αντικείμενα που ορίζονται:

```
<Tabs >
                <Tab label="Tasks" value="tasks" >
                  <div>
                  <h2>Tasks</h2>
                  <TasksTab user={this.state.user}/>
                  </div>
                </Tab>
                <Tab label="Projects" value="projects">
                  <div>
                    <h2>Projectjs</h2>
                  <ProjectsTab user={this.state.user}/>
                  </div>
                </Tab>
                <Tab label="Customers" value="customers">
                  <div>
                    <h2>Customers</h2>
                    <CustomersTab user={this.state.user}/>
                  </div>
                </Tab>
              </Tabs>
```

Στα tabs περνάει και ως πληροφορία ο συνδεδεμένος χρήστης ώστε να μπορεί να γνωρίζει η εφαρμογή αν το αντικείμενο που εμπεριέχεται είναι του χρήστη. Το αντικείμενό projectsTab, και τα υπόλοιπα, εισάγονται ως κλάσεις από άλλα αρχεία:

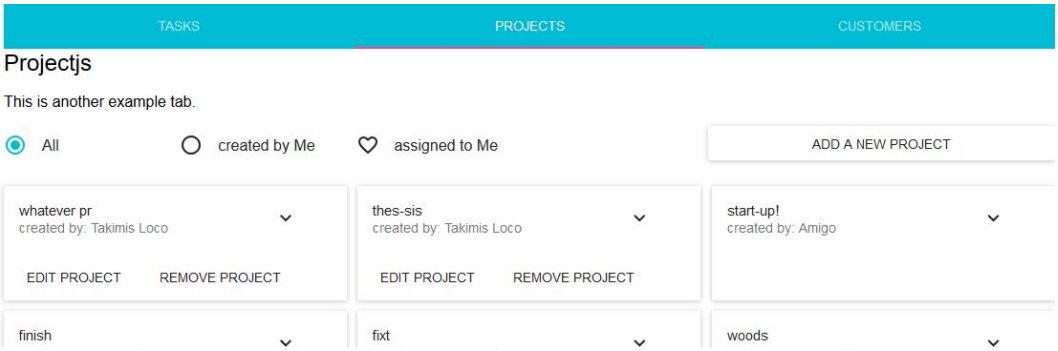
```
import TasksTab from './TasksTab.js'
import ProjectsTab from './ProjectsTab.js'
import CustomersTab from './CustomersTab.js'
```

Τα παραπάνω αρχεία εξάγουν κλάσσεις που υλοποιούν τα αντικείμενα που περικλείουν, στο ProjectsTab.js π.χ.

```
export default class ProjectsTab extends React.Component {
```

ώστε να μπορεί έπειτα να χρησιμοποιηθεί από άλλο component με τη σειρά του. Βλέπουμε ότι οι κλάση που εξάγεται επεκτείνει αντικείμενο της react, έτσι του προσδίδουμε τις ανάλογες δυνατότητες.

Τα tabs με τη σειρά τους εμφανίζουν το περιεχόμενό τους , το οποίο περιλαμβάνει κουμπιά και τα τετράγωνα των αντικειμένων της λίστας με τα project



Στο αρχείο projectsTab γίνετε εισαγωγή των απαραίτητων components, όπως το text απο τη βιβλιοθήκη material και το αντικείμενο projectTile που αντιπροσωπεύει το αντικείμενο στη λίστα:

```
import React from 'react';
import TextField from 'material-ui/TextField';
import ProjectTile from './ProjectTile.js'
```

Έπειτα θα δημιουργηθεί ένα τέτοιο αντικείμενο για κάθε παιδί στη λίστα:

```
<div className="cardscontainer">
          {this.state.projects.map((project) => {
          return (<ProjectTile
```

```
 project={project}
 user={this.props.user} />)
        })}
</div>
```

Σημειώνεται πως για κάθε project παιρνάνε τα περιεχόμενα στο αντίστοιχο παιδί ως props, ενώ στο στοιχείο που τα περιέχει δίνουμε όνομα κλάσσης css για την έυκολη μορφοποίησή τους.

Η λίστα με τα project προκύπτει κατά τη δημιουργία του αντικειμένου μετά από σύνδεση στο Api:

```
componentDidMount() {
    const projectsRef = firebase.database().ref('projects');
    projectsRef.on('value', (snapshot) => {
      let projects = snapshot.val();
      let newState = [];
      for (let project in projects) {
        newState.push({
          id: project,
          title: projects[project].title,
          creator: projects[project].creator,
          customerRel: projects[project].customerRel,
          manager: projects[project].manager
        });
      }
      this.setState({
        projects: newState
      });
    });
 }
```

Η μέθοδος render που αναπαριστά το projectTile, επιστρέφει σύνθετα html αντικείμενα με τη μορφή jsx:

```
render() {
```

```
return (
 <React.Fragment  key={this.props.project.id}>
   <Card >
{this.props.project.creator === this.props.user.displayName ||
this.props.project.creator === this.props.user.email ?
            <div>
             <Dialog
              title={ "Edit project: " + this.props.project.title
}
              actions={reactions}
              modal={true}
              open={this.state.open}
              onRequestClose={this.handleClose}
              contentStyle={customContentStyle}
            >
           ... </Dialog>
            </div>
             : null}
     </Card>
</React.Fragment>
  );}
```

Επειδή τα projectTile αποτελούν κομμάτια μίας λίστας αναφέρονται ως fragment ώστε να τα αναγνωρίζει η react ξεχωριστά και μέσω του κλειδιού key. Αυτά αποτελούνται από μία κάρτα που αναπαριστά το project και είναι κλάσση της materialUI. Μέσα εδώ μπορούμε να ορίσουμε τον τίτλο, το περιεχόμενο καθώς και την συμπεριφορά της. Επίσης με τον ίδιο τρόπο δηλώνουμε και δημιουργούμε το παράθυρο διαλόγου.

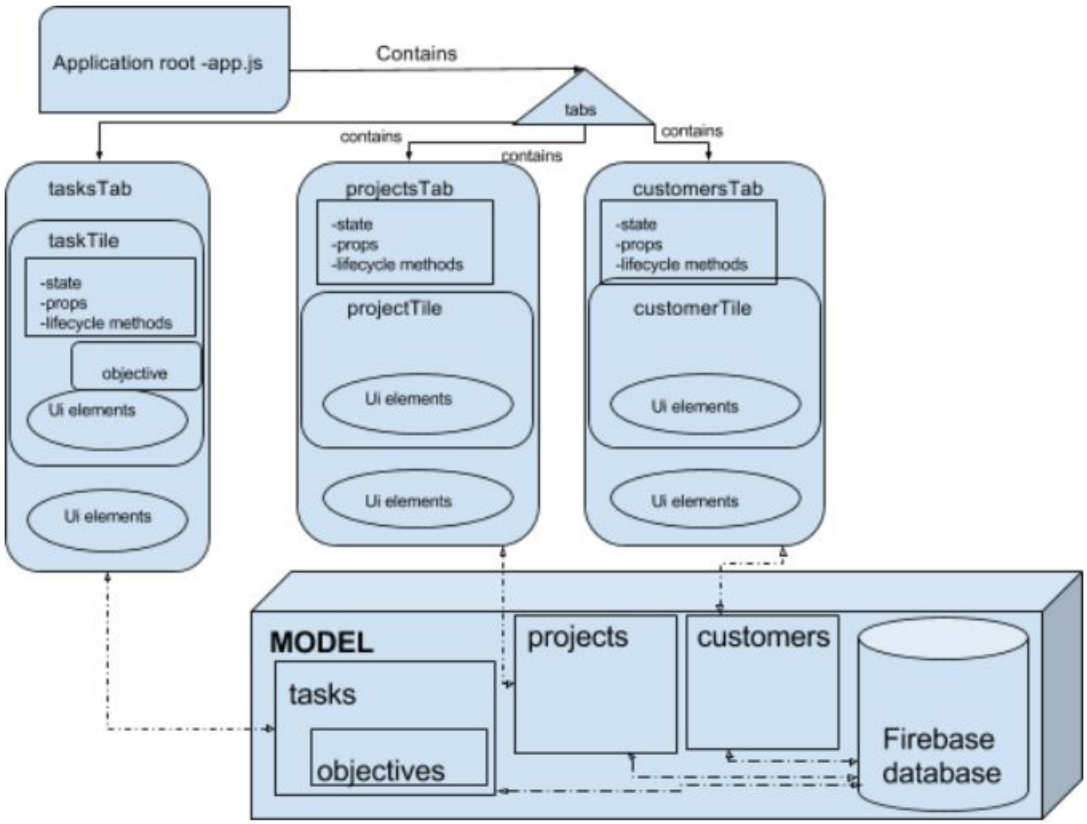
```
 <Card    style={cardStyles}    >
    <CardHeader
      title={this.props.project.title}
      subtitle={"created by: " + this.props.project.creator}
    />
    <CardText expandable={true}>
      Connected with customer : {this.state.customerName}
    </CardText>
```

```
    {this.props.project.creator === this.props.user.displayName ||
this.props.project.creator === this.props.user.email ?
    <CardActions>
      <FlatButton label="Edit Project" onClick={this.handleEdit}
/>
      <FlatButton label="Remove Project" onClick={() =>
this.removeProject(this.props.project.id)} />
    </CardActions>
</Card>
```

Το projectTile ως αντικείμενο έχει δομητή

```
constructor(props) {
    super(props);
    this.state = {
      projects: [],
      open: false,
      currentItem: this.props.project.title,
      currentCustomer: this.props.project.customerLink,
      customers: [],
      customerCon : this.props.project.customerRel
    };
    this.handleClose = this.handleClose.bind(this);
    this.handleEdit = this.handleEdit.bind(this);
    this.editProject = this.editProject.bind(this);
  }
```

Μέσα στο δομητή αρχικοποιείται το state του component βάση των props που περνιούνται ενώ επίσης συνδέονται και οι μέθοδοι που θα επηρεάζουν το state.



17.3 Ανάλυση μεθόδων που χρησιμοποιούνται

Κατά την φόρτωση του component projectTile, μετά το render θα οριστούν οι πρόσθετες τιμές για το state, ανάλογα της αναπαράστασης του εκάστοτε component με το αντικείμενο στη βάση.

```
componentDidMount() {

    const projectsRef = firebase.database().ref("projects");
    projectsRef.on('value', (snapshot) => {
      let projects = snapshot.val();
      let newStatee = [];
      for (let project in projects) {
        if(projects .projectsCon && projects
```

```
.projectsCon.indexOf(this.props.project.id))
        newStatee.push({
          id: projects ,
          title: projects[projects ].title,
        });
      }
      this.setState({
        projectsCon: newStatee
      });
    });
}
```

Έπειτα ακολουθεί η συνάρτηση ComponentWillReceiveProps η οποία ενεργοποιείται όταν ανανεωθούν τα props του component , και θέλουμε να μεταβάλλουμε και το state του component

```
componentWillReceiveProps(nextProps) {
  var newCustomerCon = nextProps.project.customerRel;
      this.setState({
        customerCon : newCustomerCon
        });

      if (newCustomerCon){
          var custRef =
firebase.database().ref(`/customers/${newCustomerCon}/`);
          custRef.once('value', (snapshot) => {
          var customer =  snapshot.val();
          if (customer){
          this.setState({
            customerName : customer.title
          });
          }
        });
      }
      else
        this.setState({ customerName : null  });
    }
```

Αυτή μας επιτρέπει να ανανεώνουμε μετέπειτα τις τιμές του αντικειμένου που εμφανίζονται σε σχέση με αυτές που υπάρχουν στη βάση. Στη συγκεκριμένη περίπτωση όταν αλλάξουν τα props που περιέχουν την σχέση project-customer και έρχονται από τον γονέα , θα ενεργοποιηθεί αυτή η συνάρτηση και θα ανανεώσει το όνομα του συσχετιζόμενου πελάτη βάση του αντικειμένου που επιστρέφεται από τη βάση.

Έπειτα στο αντικείμενο συναντάμε τις συναρτήσεις που εκτελούνται κατά την ενεργοποίηση μεθόδων μέσω στοιχείων της διεπαφής.

Κατά την αλλαγή του ονόματος ανανεώνουμε και την προσωρινή τιμή στο state πριν τη στείλουμε για αποθήκευση στη βάση, βάση της τιμής που προκάλεσε την ενεργοποίησή του,

```
handleChange = (event) => {
      this.setState({
        currentItem: event.target.value,
      });
    };
```

Κατά την αλλαγή του συνδεδεμένου πελάτη ανανεώνουμε και την ανάλογη προσωρινή τιμή στο state πριν τη στείλουμε για αποθήκευση στη βάση

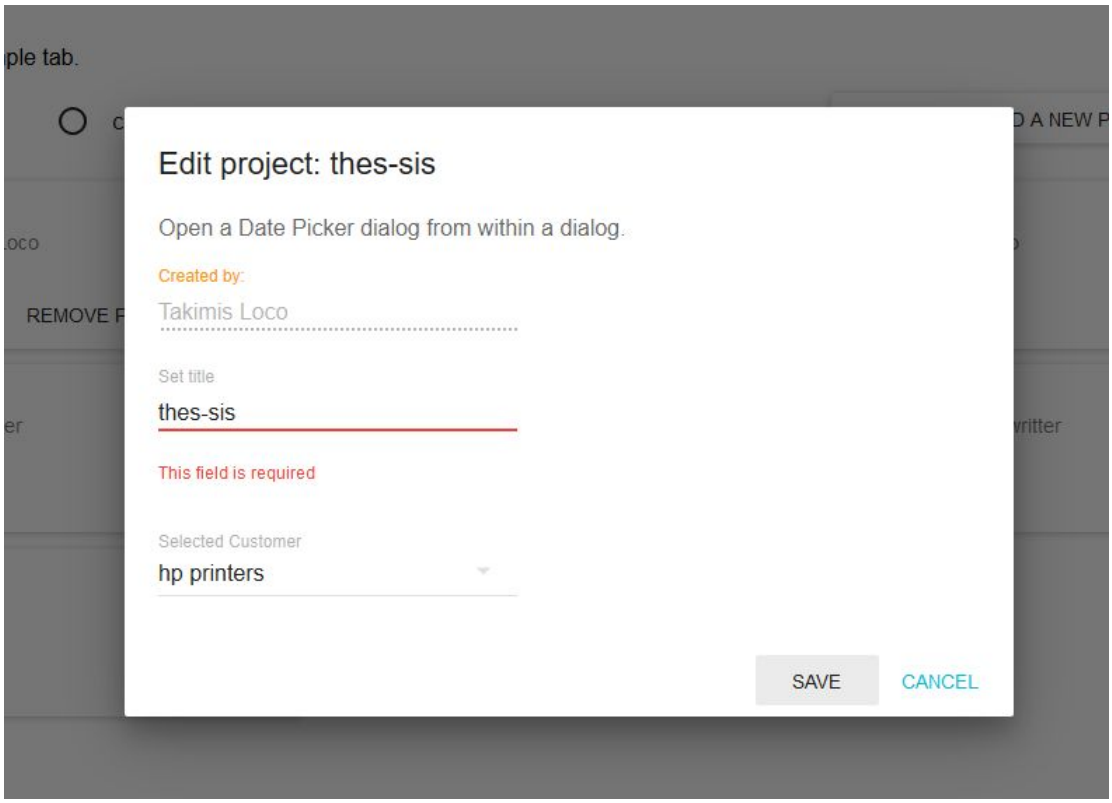
```
handleCustomerChange = (event, index, value) => {
            value ? this.setState({customerCon : value}) :
this.setState({customerCon : null})
    };
```

Η συνάρτηση handleClose η οποία όταν εκτελείται κλείνει το παράθυρο διαλόγου αλλάζοντας την τιμή της μεταβλητής στο state

```
handleClose(e ){
   this.setState({open: false}); };
```

Η συνάρτηση handleEdit αφορά τη μεταβλητή που ελέγχει τη κατάσταση για το παράθυρο διαλόγου.Αυτή υπάρχει στο state και ενεργοποιείται με το κουμπί που τη χρησιμοποιεί το εκάστοτε αντικείμενο

```
handleEdit = () =>
    this.setState({open: true});
  };
```



Όταν ανοίγει το παράθυρο διαλόγου για την επεξεργασία ενός project.

Η συνάρτηση removeProject όταν εκτελείται διαγράφει ένα αντικείμενο από την βάση δεδομένων, της περνάμε ως παράμετρο το projectId

```
 removeProject(projectId) {
    const projectRef =
firebase.database().ref(`/projects/${projectId}`);
    projectRef.remove();
  }
```

Η μέθοδος editProject μας επιτρέπει να ανανεώσουμε τις τιμές του Project στη βάση δεδομένων με βάση τις τιμές που φαίνονται στο παράθυρο διαλόγου και αναπαραστούν το state

```
editProject(e ) {

    const projectId = this.props.project.id;
    const projectRef =
firebase.database().ref(`/projects/${projectId}`);
 const project = {
      title: this.state.currentItem,
      creator: this.props.user.displayName ||
this.props.user.email,
      customerRel: this.state.customerCon ,    }
    projectRef.set(project);
    this.setState({open: false});
```

Επιπλέον στο σημείο αυτό ανανεώνουμε και τη σχέση του συνδεδεμένου με το project πελάτη στη βάση

```
const customersRef = firebase.database().ref(`/customers`);
    if(this.state.customerCon){
      const customerRelId = this.state.customerCon;
      const custProjRel =
firebase.database().ref(`/customers/${customerRelId}`);
      for(let customer in customersRef){
          var updatedRel = {projectRel : projectId};
          custProjRel.update(updatedRel);
      }
    }
  }
```

Οι παραπάνω μέθοδοι ενεργοποιούνται κατ'εντολή των element που αποτελούν το component είναι συνδεδεμένα με αυτό.

Το παράθυρο διαλόγου περιέχει δύο components κουμπιά τα οποία διαχειρίζονται την κατάστασή του :

```
    <FlatButton
      label="Save"
      primary={false}
      keyboardFocused={true}
      onClick={this.editProject}
    />,
    <FlatButton
      label="Cancel"
      primary={true}
      disabled={false}
      onClick={this.handleClose}
    />
```

Το ένα καλεί την συνάρτηση που σώζει το αντικείμενο στη βάση και κλείνει το παράθυρο , ενώ το άλλο απλά τον κλείνει.

Στην περίπτωση που ο συνδεδεμένος χρήστης είναι ίδιος με τον χρήστη που δημιούργησε το αντίστοιχο project τότε εμφανίζονται τα κουμπιά που του δίνουν τη δυνατότητα να μεταβάλλει μέσω του διαλόγου που ανοίγει ή να το διαγράψει.

```
{this.props.project.creator === this.props.user.displayName ||
this.props.project.creator === this.props.user.email ?
    <CardActions>
      <FlatButton label="Edit Project" onClick={this.handleEdit}
/>
      <FlatButton label="Remove Project" onClick={() =>
this.removeProject(this.props.project.id)} />
    </CardActions>
    :null}
```

Έπειτα ακολουθεί το σύνθετο component dialog , το οποίο παρέχεται από τη βιβλιοθήκη material και περιέχει τα γραφικά components που εμφανίζονται με την εμφάνιση του διαλόγου.

```
<Dialog
              title={"Edit project: " + this.props.project.title}
              actions={reactions}
              modal={true}
              open={this.state.open}
              onRequestClose={this.handleClose}
              contentStyle={customContentStyle}
            >

              <TextField
                  type="text"
                  name="username"
                  floatingLabelText="Created by:"
                  value={this.props.project.creator}
                  disabled={true}
              /><br />
            <TextField
                  type="text"
                  id="currentItem"
                  name="currentItem"
                  hintText="edit title"
                  floatingLabelText="Set title"
                  onChange={this.handleChange.bind(this)}
                  errorText="This field is required"
                  value={this.state.currentItem}
              /><br />
              <SelectField
                multiple={false}
                floatingLabelText="Selected Customer"
                value={this.state.customerCon}
                >
                <MenuItem key="none" value={null} primaryText=""
insetChildren={true}/>
                {this.menuCustomerItems(this.state.customers)}
              </SelectField>

            </Dialog>
```

Εδώ στο διάλογο φαίνονται δύο textfield ένα για τον τίτλο και ένα για τον δημιουργό καθώς επίσης και ένας επιλογέας που αναφέρεται σε μία λίστα με πελάτες που μπορεί να συνδεθεί.

Μέσα στη render μπορούμε να έχουμε και μεταβλητές αντικείμενα που φέρουν τιμές για μορφοποίηση της εμφάνισης.

```
render() {
  const styles = {
    errorStyle: {
      color: orange500,
    },
    underlineStyle: {
      borderColor: orange500,
    },
    floatingLabelStyle: {
      color: orange500,
    },
    floatingLabelFocusStyle: {
      color: blue500,
    },
  };
  const customContentStyle = {
    width: '33%',
    maxWidth: 'none',
  };
  const cardStyles = {
    width : '32%',
    margin: '1% 1% 0% 0',
    float: 'left',
    minHeight: 120,
  };
...
}
```

Αυτό περιέχει πεδία τύπου textfield ή selectfield τα οποία συνδέονται με τις αντίστοιχες μεταβλητές και μεθόδους του αντικειμένου που χρειάζονται όταν τους συμβεί μία αλλαγή. Τα πεδία αυτά έχουν εισαχθεί ως κλάσσεις από τη βιβλιοθήκη materialUI και αρχικοποιούνται μαζί με μερικές μεταβλητές που διατίθενται στο καθένα και περνάνε παράλληλα.

Εκτός από την εμφάνιση της λίστας αντικειμένων , το αντικείμενο projectsTab μας δίνει επιπλέον τη δυνατότητα να φιλτράρουμε τη λίστα αυτή αλλά και να προσθέσουμε αντικείμενα στη λίστα αυτή.

Τα φίλτρα αναπαριστώνται από αντικείμενα τύπου radioButton

```
<RadioButtonGroup name="filter" defaultSelected="all"
onChange={this.filterChanged}
style={filterStyles.radioButtonGroup}>
        <RadioButton
          value="all"
          label="All"
          style={filterStyles.radioButton}
        />
        <RadioButton
          value="createdMe"
          label="created by Me"
          style={filterStyles.radioButton}
        />
        <RadioButton
          value="assignedMe"
          label="assigned to Me"
          checkedIcon={<ActionFavorite style={{color: '#F44336'}}
/>}
          uncheckedIcon={<ActionFavoriteBorder />}
          style={filterStyles.radioButton}
        />
      </RadioButtonGroup>
```

Κατά την αλλαγή που συμβαίνει σε αυτή την ομάδα των κουμπιών , αλλάζει επίσης και η λίστα με αντίστοιχα αντικείμενα που υπάρχουν στο state, και φορτώνται τα νέα όπως προκύπτουν από τη βάση δεδομένων, βάση της τιμής του φίλτρου

```
filterChanged(event , value ){
    this.setState({filter: value});
    var fprojectsRef;
```

```
      if (value==='createdMe'){
        fprojectsRef =
firebase.database().ref('/projects/').orderByChild('creator').equa
lTo(this.props.user.displayName);
        }
      else if (value==='assignedMe'){
        fprojectsRef =
firebase.database().ref('/projects/').orderByChild('manager').equa
lTo(this.props.user.displayName);
        }
      else{
        fprojectsRef = firebase.database().ref('/projects/');
      }

        fprojectsRef.once('value', (snapshot) => {
          let projects = snapshot.val();
          let newStates = [];
          for (let project in projects) {
            newStates.push({
              id: project,
              title: projects[project].title,
              creator: projects[project].creator,
              customerRel: projects[project].customerRel
            });
          }
          this.setState({
            projects: newStates
          });
        });
    }
```

Στη συνέχεια υπάρχει το κουμπί που μας επιτρέπει να δημιουργήσουμε ένα νέο αντικείμενο και να το αποθηκεύσουμε στη βάση

```
<RaisedButton
 label="Add a new project"
 onClick={this.handleOpen}
 style={buttonStyles.newButton}/>
        <Dialog
          title="Add Project"
          actions={actions}
          modal={true}
```

```
          open={this.state.open}
          onRequestClose={this.handleClose}
        >
...
```

Το οποίο ενεργοποιεί με την σειρά του την αντίστοιχη συνάρτηση ή κλείνει το παράθυρο διαλόγου που άνοιξε με αυτό το σκοπό

```
handleAdd(e ){
    const projectsRef = firebase.database().ref('projects');
    const project = {
      title: this.state.currentItem,
      creator: this.props.user.displayName ||
this.props.user.email
    }
    projectsRef.push(project);
    this.setState({open: false});
  };
```

```
handleClose(e ){
           this.setState({open: false});
  };
```

Στα αντικείμενα τύπου Task ωστόσο συγκεκριμένα συμπεριέχεται και ένα πίνακας αντικειμένων objectives ο οποίος μπορεί να κρατάει παιδιά με όνομα και κατάσταση true/false , τα οποία απεικονίζουν του στόχους του εκάστοτε task

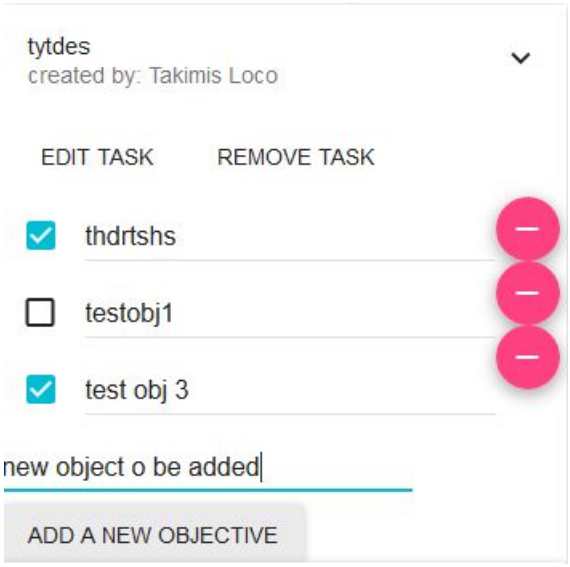
```
  componentWillReceiveProps(nextProps) {
var newObjectives = nextProps.task.objectives;
var objectivsTile=[];
for (let objective in newObjectives) {
    objectivsTile.push(<Objective
          objective={objective}
          user={nextProps.user}
          taskId={nextProps.task.id} key={objective}
          objectiveTitle={newObjectives[objective].title}
          objectiveStatus={newObjectives[objective].done}
          objectiveId={objective}
          />);
    }
```

```
    this.setState({objectivsTile: objectivsTile});
  }
```

Έτσι ορίζονται και δημιουργούνται τα αντικείμενα τύπου objective τα οποία ανανεώνονται όταν αλλάξει η τιμή τους στη βάση, με τη χρήση της μεθόδου `componentWillReceiveProps(nextProps)` . Αυτή χρησιμοποιείται γιατί πρέπει να μεταβληθεί το state, το οποίο περιέχει όλο το αντικείμενο, ανάλογα με τα νέα props που περιέχουν μόνο τη διεύθυνση.

Το objective έχει επίσης εισαχθεί ως κλάση , στο αρχείο TaskTile

```
import Objective from './Objective.js'
```



Εδώ φαίνεται πως παρουσιάζονται οι στόχοι, objectives , για κάθε task , το πεδίο και το κουμπί που εισάγουν νέο στόχο, ενώ επίσης τα δύο κουμπιά που μεταβάλλουν την κατάσταση ή διαγράφουν τον στόχο. Μέσα στα αντικείμενα objective υλοποιούνται και η μέθοδοι που ενεργοποιούνται για κάθε ένα από αυτά.

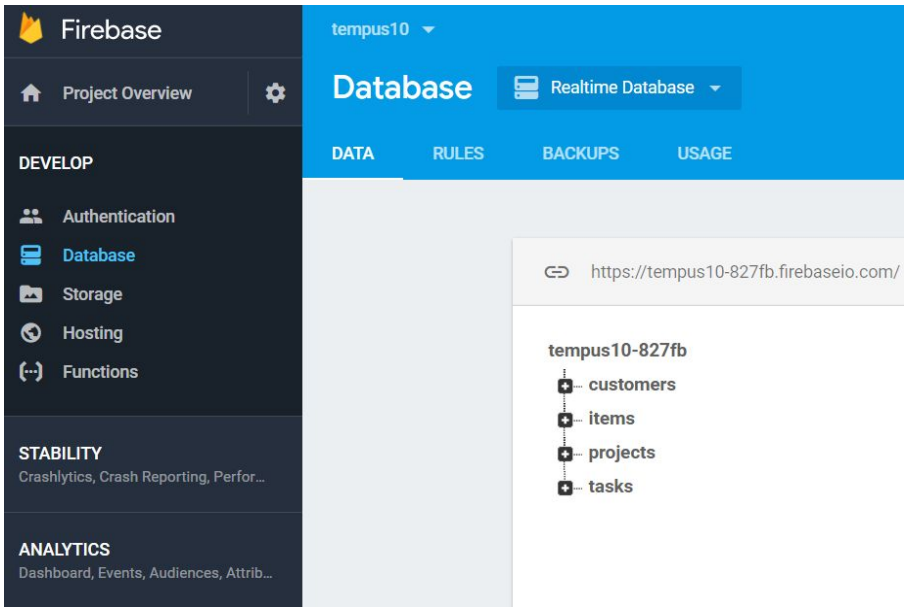
Με την ίδια φιλοσοφία που ακολουθείται στα component που παρουσιάζονται, προσεγγήστικαν και τα υπόλοιπα μέρη και μέθοδοι της εφαρμογής που δεν παρουσιάζονται.

18. Σύνδεση με API και βάση δεδομένων

Μία εφαρμογή σε react αποθηκεύει τα δεδομένα που χρειάζεται στο state και αυτό υπάρχει στη μνήμη. Σε περίπτωση που γίνει επανεκκινήση στην εφαρμογή τα υπάρχων δεδομένα στο state θα χαθούν. Για να αποθηκευτούν τα δεδομένα χρησιμοποιείται μια βάση δεδομένων, όπως σε κάθε διαδικτυακή εφαρμογή , ενώ η λογική της επικοινωνίας με αυτή γίνετε μέσω μιας διεπαφής, της μορφής REST. Με αυτή τη διεπαφή γίνετε εφικτή η υλοποίηση όλων λειτουργιών CRUD, create - read - update - delete, με τη διαφορά ότι ο σύνδεσμος μπορεί να επιστρέφει αντικείμενο, συνήθως σε μορφή json, και άρα υλοποιούνται πιο σύνθετες οδηγίες και μπορούν να αναπαρασταθούν πιο σύνθετες πληροφορίες όπως για παράδειγμα οι σχέσεις με τα συσχετιζόμενα ή συνδεδεμένα αντικείμενα στη βάση.

Για τον σκοπό αυτό έχουν δημιουργηθεί διάφορες διεπαφές που ενώνουν μία πλατφόρμα με την βάση δεδομένων της εκάστοτε τεχνολογίας. Η βάση δεδομένων επιπλέον τρέχει σε εξωτερικό ή εσωτερικό server, και για να συνδεθεί η πλατφόρμα με αυτή, πρέπει να γίνει παραμετροποίηση των ρυθμίσεων της διεπαφής επικοινωνίας με τη βάση δεδομένων.

Μία βάση δεδομένων που παρέχει έτοιμη διεπαφή για επικοινωνία είναι το firebase.



Η διεπαφή της πλατφόρμας firebase

Το firebase έχει δημιουργηθεί από την google και παρέχει τη δυνατότητα χρήσης αφηρημένης βάσης δεδομένων χωρίς τη χρήση server, ενώ η πρόσβαση στα δεδομένα αυτά είναι εφικτή εκτός από μορφή link , και μέσω της διεπαφής για την εκάστοτε πλατφόρμα, από την οποία μπορούν να εκτελεστούν όλες οι λειτουργίες που υποστηρίζονται . Το firebase επίσης παρέχει ξεχωριστό περιβάλλον διαχείρισης της βάσης, περιλαμβάνει σύστημα διαχείρισης χρηστών και αναγνώρισης χρηστών, και υποστηρίζει τη δημιουργία κανόνων για τη διατήρηση της λογικής του μοντέλου.

Η εγκατάσταση του Firebase γίνετε στη γραμμή εντολών στο path του server με την εντολή

```
yarn install firebase
```

Και έπειτα παρέχεται η κλάσση firebase και παρεμφερείς μεθόδοι στην εφαρμογή μας:

```
// src/firebase.js
import firebase from 'firebase'
```

Το υπόλοιπο αρχείο ρυθμίσεων του firebase

```
const config = {
  apiKey: "[redacted]",
    authDomain: "tempus[redacted].firebaseapp.com",
    databaseURL: "https://tempus[redacted].firebaseio.com",
    projectId: "tempus[redacted]",
    storageBucket: "tempus[redacted].appspot.com",
    messagingSenderId: "[redacted]"
};
firebase.initializeApp(config);

export const provider = new firebase.auth.GoogleAuthProvider();
export const auth = firebase.auth();

export default firebase;
```

Κλήση στο firebase

```
const itemsRef = firebase.database().ref('items');
    itemsRef.on('value', (snapshot) => {
```

Δημιουργία δεδομένων στο state του αντικειμένου από το api του firebase , με τη μέθοδο push

```
const itemsRef = firebase.database().ref('items');
    itemsRef.on('value', (snapshot) => {
      let items = snapshot.val();
      let newState = [];
      for (let item in items) {
        newState.push({
          id: item,
          title: items[item].title,
          user: items[item].user
        });
      }
      this.setState({
        items: newState
      });
```

Η εντολή push δημιουργεί ένα νέο αντικείμενο με τις τιμές που ορίζονται στη διεύθυνση που αναφέρεται. Το αντικείμενο που περιέχει τη μέθοδο push είναι από τη κλάση που μας παρέχει το firebase firebase.database().ref() .

Διαγραφή δεδομένων στο firebase , με τη μέθοδο remove

```
removeItem(itemId) {
    const itemRef = firebase.database().ref(`/items/${itemId}`);
    itemRef.remove();
  }
```

Ανανέωση δεδομένων στο firebase, , με τη μέθοδο update

```
const taskId = this.props.task.id;
const taskRef = firebase.database().ref(`/tasks/${taskId}`);
const task = {
      title: this.state.currentItem,
```

```
      creator: this.props.user.displayName ||
this.props.user.email,
      projectsCon: newProjectsCon
 }
    taskRef.update(task);
```

Αντικατάσταση δεδομένων στο firebase , με τη μέθοδο set

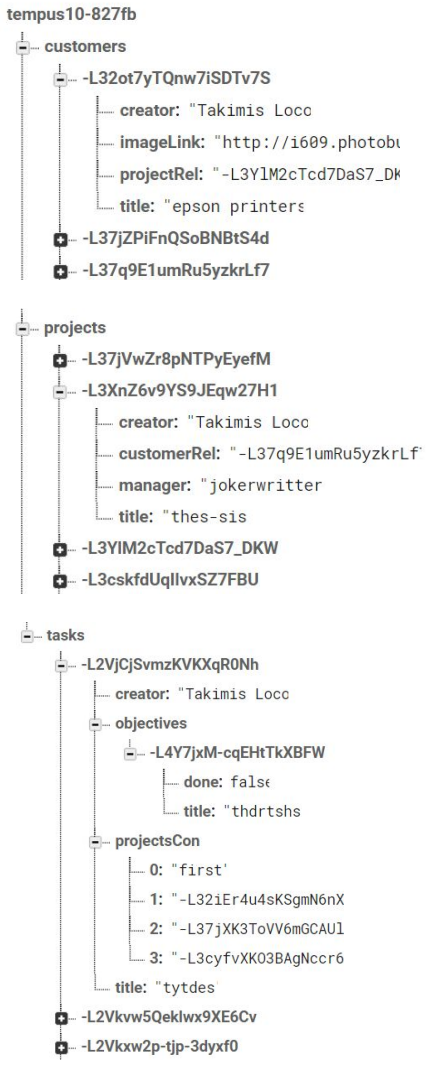
```
const projectId = this.props.project.id;
    const projectRef =
firebase.database().ref(`/projects/${projectId}`);
 const project = {
      title: this.state.currentItem,
      creator: this.props.user.displayName ||
this.props.user.email,       customerRel: this.state.customerCon ,
    }
    projectRef.set(project);
```

19. Εφαρμογή του template

Η πλατφόρμα που αναπτύχθηκε αφορά την διαχείριση καθηκόντων και έργων. Επιτρέπει τη σύνδεση χρηστών, τη δημιουργία και μεταβολή των αντικειμένων στη βάση δεδομένων που αποθηκεύονται, το firebase. Τα αντικείμενα στη μεριά των χρηστών, ανανεώνονται βάση των αλλαγών που συμβαίνουν στη βάση, οπότε η πλατφόρμα κρατάει τα δεδομένα που εμφανίζονται στον εκάστοστε χρήστη συντονισμένα .

Μπορεί να χρησιμοποιηθεί από τον απλό χρήστη ο οποίος θέλει να οργανώσει το πρόγραμμα της καθημερινότητάς του, ενώ επειδή περιλαμβάνει την έννοια οργάνωσης και ομαδοποίησης αρμοδιοτήτων μπορεί να εφαρμοστεί για να μοιραστούν τα επιμέρους καθήκοντα μεταξύ των μελών μίας ομάδας, και να γίνεται καταγραφή της προόδου των μερών ενός έργου σε πραγματικό χρόνο.

Σε περίπτωση που τα έργα αυτά αναφέρονται σε κάποιον εκτός τις ομάδας, όπως έναν πελάτη για παράδειγμα δίνετε η δυνατότητα αντιστοίχησης του εκάστοτε έργου και λεπτομερειών του, και η δυνατότητα καταγραφής επισημάνσεων της επικοινωνίας με των πελάτη και στοιχείων επικοινωνίας του, εξυπηρετώντας έτσι και σκοπούς πλατφόρμας διαχείρισης πελατειακών σχέσεων. Αυτό γίνετε γιατί η πληροφορία για το εκάστοτε έργο του πελάτη αποθηκεύεται και μπορεί να ενημερώνονται άμεσα οι υπεύθυνοι χρήστες για αυτό.



Τα δεδομένα της εφαρμογής στο firebase

Έτσι αυτή η εφαρμογή μπορεί να τροποποιηθεί και να χρησιμοποιηθεί στο εσωτερικό δίκτυο μιας επιχείρησης ως η πλατφόρμα συντονισμού και οργάνωσής της.

Επίσης τα δεδομένα αποθηκεύονται σε ξεχωριστό μέρος, οπότε εγγυάται η ασφάλεια και ακεραιότητά τους, ενώ παράλληλα επειδή αυτά υπάρχουν σε μορφή αντικειμένων είναι εύκολο να χρησιμοποιηθούν από άλλες πιθανές εφαρμογές που τα χρειαζόμαστε, να εισαχθούν νέα και να ανανεωθεί η βάση των υπαρχόντων δεδομένων, ή να εξαχθούν σε μορφή που βολεύει και να μορφοποιηθούν περαιτέρω, από προγράμματα όπως το excel για παράδειγμα.

Είναι πιθανό ωστόσο να αλλάξει τελείως η βάση δεδομένων firebase με μία βάση που θα τρέχει στον ίδιο server της εφαρμογής, η οποία πάλι θα μπορεί να επιστρέφει αντικείμενα ως δεδομένα. Για να συμβεί αυτό χρειάζεται η εγκατάσταση του ανάλογου λογισμικού στο server, η παραμετροποίηση του αλλά και η υλοποίηση των κλάσεων σύνδεσης με τη βάση, που υπάρχουν στη react και ενεργοποιούνται όταν θέλουμε να ανανεώσουμε τα δεδομένα μας, διότι μπορεί να υπάρχουν μικροδιαφορές σε αυτές τις βάσεις παρότι είναι ενδεχομένως παρόμοιας φιλοσοφίας.

20. Μελλοντικές επεκτάσεις

Λόγω της αρχιτεκτονικής που έχει εφαρμοσθεί και των τεχνολογιών που χρησιμοποιήθηκαν οι πιθανές επεκτάσεις που μπορούν να γίνουν είναι αρκετές, και δεν περιορίζονται παρά μόνο λόγω ασυμβατότητας τεχνολογιών, γεγονός δύσκολο αφού οι περισσότερες τεχνολογίες είναι συμβατές με τη javascript.

Αρχικά μπορεί να δημιουργηθεί αυτόνομο σύστημα αναγνώρισης χρηστών και ασφάλειας, διότι προς το παρόν χρησιμοποιείται το δίκτυο της google, ενώ παράλληλα μπορεί να προστεθεί και η αναγνώριση μέσω άλλων δικτύων που μας παρέχουν authenication api για τους χρήστες τους.

Έπειτα, θα ήταν πολύ εξυπηρετικό για τους χρήστες να τους δινόταν η δυνατότητα δημιουργίας ομάδων αλλά και η ιεραρχική οργάνωσή τους. Ωστόσο αυτός ο σχεδιασμός θα πρέπει να γίνει προσεκτικά διότι σε περίπτωση επέκτασης και διαφοροποίησης των δικαιωμάτων χρήστη ανά ομάδα ή βαθμό, επέρχεται τέτοια

πολυπλοκότητα που μπορεί να οδηγήσει σε δύσκολα συντηρίσιμη και περαιτέρω επεκτάσιμη πλατφόρμα. Και αυτό είναι λογικό αφού πλέον θα αρχίσει να αποκτά δυνατότητες κοινωνικού δικτύου.

Προς διευκόλυνση της επικοινωνίας μεταξύ των χρηστών μπορεί επίσης να δημιουργηθεί ένα component το οποίο θα περιέχει τη λογική της άμεσης συζήτησης, αλλά και η δυνατότητα ειδοποιήσεων στους χρήστες ενημερώνονται άμεσα για τα κρίσιμα ζητήματα. Παράλληλα μπορεί να επεκταθεί το φιλτράρισμα των δεδομένων και να επεκταθεί η δυνατότητα αναζήτησης βάση χρήστη, αλλά και να γίνει δυνατή η ταξινόμηση των δεδομένων στις λίστες βάση ποσοτικών παραμέτρων.

Σε ακόμη πιο εξειδικευμένες περιπτώσεις, η πλατφόρμα που εξετάστηκε μπορεί να συνδεθεί με τις διεπαφές που παρέχονται από υπάρχουσες πλατφόρμες που χρησιμοποιούν οι χρήστες, ώστε έτσι να συγχρονιστούν τα δεδομένα διαμέσου των πλατφορμών και να αποτελεί έτσι η εφαρμογή κεντρικό σημείο αναφοράς ανάμεσά τους, ενώ αυτές συνδυάζονται όπως ορίζουν οι απαιτήσεις του χρήστη. Έτσι μπορεί να συνδεθεί η εφαρμογή με πλατφόρμες προγραμματιστών που αναπαριστούν project, όπως, ή επιτρέπουν την αποθήκευση αρχείων στο διαδίκτυο και το διαμοιρασμό τους, όπως το dropbox.

Επιπρόσθετα είναι δυνατό να δοθεί στο χρήστη η δυνατότητα δημιουργίας δικών του οντολογιών ή κλάσεων προς εξυπηρέτηση των αποκλειστικών σκοπών του, καθώς επίσης και να προσδιοριστούν εύκολα η ιδιότητές των. Αυτό είναι ήδη εφικτό με την επέκταση του κώδικα, αλλά μπορεί επίσης να δημιουργηθεί γραφική διεπαφή για αυτό το σκοπό, φιλική στον απλό χρήστη. Λειτουργικότητες όπως αυτή συνήθως συναντάται στα συστήματα διαχείρισης περιεχομένου .

Παράλληλα μπορεί να εύκολα επεκταθεί με τη χρήση ενός component το οποίο θα είναι ικανό να μεταβάλλει παραμέτρους της γραφικής διεπαφής, όπως τα χρώματα ή το μέγεθος της γραμματοσειράς, βάση των προτιμήσεων του χρήστη, καθιστώντας έτσι τη διεπαφή πιο ελκυστική ανά κάθε ξεχωριστό χρήστη. Επίσης αυτό μπορεί να επιτρέψει τη χρήση χρώματος ώστε να προσδίδει έννοια στο αντίστοιχο περιεχόμενο, κατ' εντολή του χρήστη.

20.1 Τεχνικές αναβαθμίσεις

Για να επιτευχθούν πολλά από τα παραπάνω και η εφαρμογή να συνεχίσει να αποκρίνεται ακέραια είναι επίσης σημαντικό να γίνουν και βελτιώσεις από τεχνολογικής απόψεις . Πρώτη από αυτές είναι η επέκτασή της με την αρχιτεκτονική redux, ώστε να δημιουργηθεί ο σκελετός που θα υποστηρίξει όλη αυτή τη πολυπλοκότητα, και επόμενη είναι η τροποποίησή της ώστε να μπορεί να τρέχει ταυτόχρονα στο server, isomorphic. Τέλος μπορεί να επεκταθεί το σύστημα συνδέσμων routing, ώστε να αναπαριστώνται τα component με τα αντίστοιχα url και στο server της εφαρμογής πλέον, όχι μόνο στο api του firebase.

Τεχνολογικές βελτιώσεις που μπορούν να γίνουν είναι εγκατάσταση στο server ξεχωριστής τεχνολογίας βάσης δεδομένων, η οποία θα επεκτείνει το μοντέλο και μπορεί να συνδυαστεί με το firebase ή να το αντικαταστήσει. Στην περίπτωση που η τεχνολογία αυτή βασίζεται σε γράφους, τότε τα δεδομένα μας μπορούν να αναπαραστήσουν πολύπλοκες σχέσεις μεταξύ τους, στις οποίες είναι εύκολα δυνατή η αναφορά.

21. Συμπεράσματα - επίλογος

Η react εισήγαγε μία νέα προσέγγιση στην ανάπτυξη των διαδικτυακών εφαρμογών. Επειδή αυτή διαφέρει από την κλασσική προσέγγιση, μπορεί κάποιος να δυσκολευτεί στην αρχή μέχρι να την κατανοήσει σε επαρκή βαθμό. Η προσέγγιση αυτή λέγεται component-based, δηλαδή τη βάση της αποτελούν τα component, οι δομικοί λίθοι της εφαρμογής.

Ωστόσο η react, από μόνη της, δημιουργήθηκε μόνο για την υλοποίηση της όψης ή της της αναπαράστασης δεδομένων. Η λειτουργία της, επεκτείνεται με την χρήση πρόσθετων τεχνολογιών που προσδίδουν πλήρες αρχιτεκτονική MVC στην πλατφόρμα μας με το redux. Ο συνδυασμός του redux με την react , παρέχει ένα σύστημα ικανό να αναπαραστήσει και υλοποιήσει μία οσοδήποτε πολύπλοκη διεπαφή με την ίδια απλοϊκή προσέγγιση.

Με την κατανόηση του τρόπου σκέψης που επέρχεται με τη react, αλλάζει και ο τρόπος με τον οποίο σχεδιάζεται και υλοποιείται οι πλατφόρμα. Τα component αποτελούν αντικείμενα, τα οποία μπορούν να περιέχουν λειτουργίες αλλά και να συνδυάζονται μεταξύ τους, με σκοπό την επίτευξη πολύπλοκων διεπαφών . Παράλληλα κάποιος όχι απλά μπορεί να αναφερθεί στις μεταβλητές τους ανά πάσα στιγμή, αλλά μπορεί επιπροσθέτως να τις τροποποιήσει σε οποιαδήποτε στιγμή του κύκλου ζωής του.

Έτσι παρά της ενδεχόμενης δυσκολίας εκμάθησης που μπορεί να αντιμετωπίσει κάποιος ερχόμενος σε επαφή με όλες αυτές τις νέες, ελαφρώς αλληλοεξαρτώμενες με τη react, τεχνολογίες, ανταμείβεται τελικά σε βάθος χρόνο, αφού πλέον έχει στη διάθεσή του μία στοίβα, που του επιτρέπει την ανάπτυξη, αλλά και σχεδίαση σύγχρονων εφαρμογών που εξυπηρετούν τους περισσότερους σύνηθες σκοπούς μιας διαδικτυακής εφαρμογής, ενώ αυτή συνεχίζει να παραμένει επεκτάσιμη και συντηρίσιμη.

22. Βιβλιογραφία

Introduction to Web Applications: web.stanford.edu/~ouster/CS349W/lectures/webApps.html.

“10 Famous Apps Using ReactJS Nowadays.” *Blog Brainhub.eu*, 1 May 2017, brainhub.eu/blog/10-famous-apps-using-reactjs-nowadays/.

“React Redux: Architecture Overview – The Musings of Colt Pini.” *The Musings of Colt Pini*, The Musings of Colt Pini, 19 Nov. 2016, articles.coltpini.com/react-redux-architecture-overview-7b3e52004b6e.

“ReactJS vs Angular Comparison: Which Is Better? – Hacker Noon.” *Hacker Noon*, Hacker Noon, 19 Dec. 2016, hackernoon.com/reactjs-vs-angular-comparison-which-is-better-805c0b8091b1

“Angular vs React: Feature Comparison of JS Tools.” *Thinkmobiles*, 5 Oct. 2017, thinkmobiles.com/blog/angular-vs-react/.

“Babel · The Compiler for Writing next Generation JavaScript.” *Babel · The Compiler for Writing next Generation JavaScript*, babeljs.io/.

Deabes, Sameh. “Introduction to Web Application Performance - DZone Performance.” *Dzone.com*, 16 Sept. 2017, dzone.com/articles/introduction-to-web-applications-performance.

“Difference between REST and CRUD.” *Software Engineering Stack Exchange*, softwareengineering.stackexchange.com/questions/120716/difference-between-rest-and-crud.

Facebook. “Facebook/Create-React-App.” *GitHub*, github.com/facebook/create-react-app/blob/master/README.md#getting-started.

“Front and Back Ends.” *Wikipedia*, Wikimedia Foundation, 6 Feb. 2018, en.wikipedia.org/wiki/Front_and_back_ends.

“Hello World.” *React*, reactjs.org/docs/hello-world.html.

“In Depth Overview.” *Flux | Application Architecture for Building User Interfaces*, facebook.github.io/flux/docs/in-depth-overview.html#content.

“Intro to Firebase and React.” *CSS-Tricks*, 6 Sept. 2017, css-tricks.com/intro-firebase-react/.

“Introduction to Web Applications.” *Webapps*, cs.lmu.edu/~ray/notes/webapps/.

“JavaScript's History and How It Led To ReactJS.” *The New Stack*, 12 Dec. 2014, thenewstack.io/javascripts-history-and-how-it-led-to-reactjs/.

“Model–View–Controller.” *Wikipedia*, Wikimedia Foundation, 29 Jan. 2018, en.wikipedia.org/wiki/Model–view–controller.

“Modern Web App Architecture.” *MDN Web Docs*, developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture.

“Multitier Architecture.” *Wikipedia*, Wikimedia Foundation, 28 Jan. 2018, en.wikipedia.org/wiki/Multitier_architecture.

Patel, Ankit. “Why to Use ReactJS?” *XongoLab Blog*, 2 Feb. 2018, www.xongolab.com/blog/why-to-use-reactjs/.

“React (JavaScript Library).” *Wikipedia*, Wikimedia Foundation, 20 Jan. 2018, en.wikipedia.org/wiki/React_(JavaScript_library)#Architecture_beyond_HTML.

“React To The Future With Isomorphic Apps.” *Smashing Magazine*, 21 Apr. 2015, www.smashingmagazine.com/2015/04/react-to-the-future-with-isomorphic-apps/.

“React.Component.” *React*, reactjs.org/docs/react-component.html.

“React.js and How Does It Fit In With Everything Else?” *Funny Ant*, 22 Aug. 2014, www.funnyant.com/reactjs-what-is-it/.

Reactjs. “Reactjs/Redux.” *GitHub*, 3 Feb. 2018, github.com/reactjs/redux/tree/master/docs.

Severien, Tim. “Yarn vs Npm: Everything You Need to Know.” *SitePoint*, SitePoint, 8 May 2017, www.sitepoint.com/yarn-vs-npm/.

Sherman, Paul. “A Simple React Router v4 Tutorial – Paul Sherman – Medium.” *Medium*, Medium, 11 Mar. 2017,

medium.com/@pshrmn/a-simple-react-router-v4-tutorial-7f23ff27adf.

Silveira, Zach. “The Pain and the Joy of Creating Isomorphic Apps in ReactJS.” *ReactJS News*, reactjsnews.com/isomorphic-react-in-real-life.

“Thinking in React.” *React*, reactjs.org/docs/thinking-in-react.html.

“Try React.” *React*, reactjs.org/docs/try-react.html.

“Understanding REST Service Operations.” *Moved*, docs.oracle.com/cd/E41633_01/pt853pbh1/eng/pt/tibr/concept_UnderstandingRESTServiceOperations.html.

“Usage with React Router.” *Usage with React Router · Redux*, redux.js.org/docs/advanced/UsageWithReactRouter.html.

“Using Firebase with ReactJS.” *The Firebase Blog*, 1 May 2014, firebase.googleblog.com/2014/05/using-firebase-with-reactjs.html.

“What Is Web Application Architecture? How It Works, Trends, Best Practices and More.” *Stackify*, 21 Sept. 2017, stackify.com/web-application-architecture/.

“Why Did We Build React?” *React Blog*, reactjs.org/blog/2013/06/05/why-react.html.

Zadorozhnyi, Tim. “Read 5 Top Things We like about ReactJs and Building Web Applications with It.” *Better Code, Better Life*, 12 Aug. 2016, betterstack.org/2016/01/12/5-reasons-why-react-js-is-awesome/.

“Bundle Your Assets Scripts.” *Webpack Icon*, webpack.js.org/.

styled-components. “Styled-Components.” *Styled-Components*, www.styled-components.com/.