ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

" Κατασκευή Εφαρμογής με τεχνολογία MERN"

ΜΟΥΛΛΑΡΑΙ ΑΡΜΑΝΤΟ

ΕΠΙΒΛΕΠΩΝ: Σωτήρης Χριστοδούλου, ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

ΕΥΧΑΡΙΣΤΙΕΣ

Υπήρξαν πολλοί οι άνθρωποι που στάθηκαν δίπλα μου στην εκπόνηση της διπλωματικής μου εργασίας, παρέχοντας μου ηθική συμπαράσταση καθώς και πρακτικά. Καταρχήν θα ήθελα να ευχαριστήσω την οικογένειά μου για την υποστήριξη που μου προσέφερε σε όλη τη διάρκεια της φοίτησής μου στη σχολή. Επίσης, οφείλω να ευχαριστήσω τον επιβλέπων της εργασιας Κύριο Σωτήρη Π. Χριστοδούλου. Χωρίς τη βοήθειά του δεν θα είχε ολοκληρωθεί ποτέ αυτή η εργασία. Θα ήθελα να σας ευχαριστήσω πολύ για την υποστήριξη και την κατανόησή σας τα τελευταία χρόνια.

ΠΕΡΙΛΗΨΗ

Στο πλαίσιο της εργασίας αυτής θα ασχοληθούμε με την δημιουργία μιας εφαρμογής αγοράς που θα μπορεί να χρησιμοποιηθεί για να κάνουμε κρατήσεις σε ξενοδοχεία. Η εφαρμογή αυτή θα επιτρέπει στον χρήστη να ανεβάζει στην δικιά του πλατφόρμα διαθέσιμα ξενοδοχεία, δωμάτια και ημερομηνίες. Παράλληλα η εφαρμογή θα λειτουργεί και ως πλατφόρμα για κάποιον πελάτη ο οποίος θα θέλει να κάνει κράτηση στα διαθέσιμα ξενοδοχεία, δωμάτια και ημερομηνίες.

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΕΙΣΑΓΩΓΗ

Η τεράστια άνοδος του τουρισμού τις τελευταίες δεκαετίες, έχει καταστήσει δικαιολογημένα την Ελλάδα σε έναν από τους δυναμικότερους τομείς της διεθνούς οικονομίας. Οι λόγοι αυτής της αλματώδης ανάπτυξης ήταν η εξέλιξη της τεχνολογίας, η βελτίωση των μέσων μαζικής μεταφοράς και οι αλλαγές που παρουσιάζονταν στην κοινωνία, με τη μετάβαση από τη μία δεκαετία στην επόμενη. Επιπλέον, οι έντονες διαφοροποιήσεις αλλά και η συνεχής ανάδειξη νέων προορισμών, συνέβαλαν με τη σειρά τους θετικά στους υψηλούς ρυθμούς ανάπτυξης του διεθνούς τουρισμού. Στη σύγχρονη εποχή, κάνουμε λόγο για ένα κοινωνικό, πολιτιστικό και οικονομικό φαινόμενο της εποχής που αποτελεί αναπόσπαστο τμήμα του παγκόσμιου εμπορίου, με τις τουριστικές εισπράξεις να κατέχουν σημαντικό μερίδιο και να συναγωνίζονται εκείνες που προέρχονται από τις εξαγωγές καυσίμων, χημικών και προϊόντων της αυτοκινητοβιομηχανίας. Ειδικότερα για την Ελληνική οικονομία , έχει μεγάλη σπουδαιότητα καθώς επηρεάζει θετικά το Ακαθάριστο Εγχώριο Προϊόν, τις επενδύσεις και την απασχόληση ενώ συγχρόνως περιορίζει το έλλειμμα στο ισοζύγιο των πληρωμών.

Η πλούσια πολιτιστική κληρονομιά, η εκτεταμένη ακτογραμμή και το φυσικό περιβάλλον είναι ορισμένα από τα συγκριτικά πλεονεκτήματα που καθιστούν τη χώρα από τους σημαντικότερους τουριστικούς προορισμούς παγκοσμίως. Ταυτόχρονα, ο τουρισμός μαζί με την ναυτιλία αποτελούν τους πιο εξωστρεφείς τομείς της ελληνικής οικονομίας . Συγκεκριμένα, ο τουρισμός, συνεισφέρει το 16,4% στο ΑΕΠ, απασχολεί έναν στους 5 εργαζόμενους της χώρας ενώ τέλος καλύπτει το 51,2% του ελλείμματος του εμπορικού ισοζυγίου.

Με το πέρασμα των χρόνων παρατηρείται στον τομέα της εστίασης υψηλή άνοδος. Εφαρμογές όπως το booking.com, Airbnb.com κλπ. βοηθούν τις ελληνικές επιχειρήσεις να ακμάζουν και ταυτόχρονα να βοηθούν στην ελληνική οικονομία. Τα κοινά χαρακτηριστικά που έχουν αυτές οι εφαρμογές, πέραν του ότι βοηθούν τους Έλληνες πολίτες στην προώθηση του χώρου τους, είναι ότι όλες εδρεύουν σε χώρες του εξωτερικού.

1.1 ΠΕΡΙΓΡΑΦΗ MERN ΤΕΧΝΟΛΟΓΙΑΣ

Για την δημιουργία του προγράμματος θα χρησιμοποιήσουμε μια τεχνολογία αρκετά γνωστή που ονομάζεται MERN.

Τι είναι όμως η τεχνολογία MERN και γιατί είναι τόσο γνωστή;

Το MERN είναι ένα αρκτικόλεξο που χρησιμοποιείται για να περιγράψει ένα συγκεκριμένο σύνολο τεχνολογιών που βασίζονται σε JavaScript που χρησιμοποιούνται στη διαδικασία ανάπτυξης εφαρμογών Ιστού. Έχει σχεδιαστεί με μια ιδέα να κάνει τη διαδικασία ανάπτυξης όσο το δυνατόν πιο ομαλή και γρήγορη. Το MERN περιλαμβάνει τα ακόλουθα στοιχεία ανοιχτού κώδικα:

- **Mongo DB**
- **Express JS**
- **React JS/ Redux**
- **Node JS**

Κάθε ένα από αυτά τα στοιχεία διαδραματίζει κρίσιμο ρόλο στη διαδικασία ανάπτυξης εφαρμογών ιστού (web application). Όλα αυτά παρέχουν ένα πλαίσιο από άκρο σε άκρο για να εργαστούν οι προγραμματιστές. Για παράδειγμα, το Mongo DB είναι ένα σύστημα βάσης δεδομένων, το Node JS είναι ένα περιβάλλον εκτέλεσης back-end, το Express JS είναι ένα πλαίσιο web back-end και το React είναι ένα πλαίσιο front-end.

Λοιπόν, ας ρίξουμε μια πιο προσεκτική ματιά σε κάθε ένα από αυτά τα συστατικά:

1. Mongo DB

Είναι ένα δωρεάν πρόγραμμα βάσης δεδομένων ανοιχτού κώδικα, πολλαπλών πλατφόμων, προσανατολισμένο σε έγγραφα. Είναι ταξινομημένο ως πρόγραμμα βάσης δεδομένων No SQL, πράγμα που σημαίνει ότι τα δεδομένα αποθηκεύονται σε ευέλικτα έγγραφα με γλώσσα ερωτημάτων που βασίζεται σε JSON. Αυτό σημαίνει επίσης ότι το μέγεθος του αριθμού περιεχομένου των πεδίων στα έγγραφα τείνει να ποικίλλει. Ολόκληρα τα δεδομένα είναι δομημένα με τέτοιο τρόπο ώστε να είναι επιρρεπή σε αλλαγές με την πάροδο του χρόνου.

Το MongoDB είναι γνωστό ως μια ευέλικτη λύση που είναι πάντα εύκολο να κλιμακωθεί. Αυτό το συγκεκριμένο σύστημα βάσης δεδομένων αναπτύχθηκε από την Mongo DB Inc. και δημοσιεύεται με συνδυασμό της άδειας γενικής δημόσιας άδειας GNU Affero και της άδειας Apache.

2. Express JS

Αυτό είναι επίσης ένα δωρεάν λογισμικό ανοιχτού κώδικα. Κυκλοφορεί με την άδεια του MIT και μπορεί να ταξινομηθεί ως πλαίσιο εφαρμογής web για το Node.js. Για να είμαστε πιο ακριβείς, το Express JS έχει σχεδιαστεί για την ανάπτυξη εφαρμογών ιστού και API.

Αντί να γράφουν με μη αυτόματο τρόπο τον πλήρη κώδικα διακομιστή ιστού στο Node.js, οι προγραμματιστές χρησιμοποιούν αυτό το στοιχείο MERN για να απλοποιήσουν τη διαδικασία κωδικοποίησης. Το καλύτερο χαρακτηριστικό αυτού του πλαισίου είναι ότι οι προγραμματιστές δεν επαναλαμβάνουν τον ίδιο κώδικα ξανά και ξανά, όπως θα έκαναν με την εγγραφή κώδικα Node.js στη λειτουργική μονάδα HTTP.

Πολλοί προγραμματιστές το αναφέρουν συχνά ως «de facto τυπικό πλαίσιο διακομιστή για το Node.js».

3. React JS/ Redux

Η React είναι μια βιβλιοθήκη JavaScript που χρησιμοποιείται για τη δημιουργία διεπαφών χρήστη. Αρχικά δημιουργήθηκε από έναν μηχανικό λογισμικού που εργαζόταν για το Facebook, το React ήταν αργότερα σε ανοιχτό κώδικα. Αυτή τη στιγμή διατηρείται από το Facebook και μια ισχυρή κοινότητα που αποτελείται από διαφορετικές εταιρείες ανάπτυξης λογισμικού και ανεξάρτητους προγραμματιστές.

Αυτή η συγκεκριμένη βιβλιοθήκη χρησιμοποιείται συχνά για τη δημιουργία προβολών που αποδίδονται σε HTML. Οι προβολές που δημιουργείτε στο Reach declarative, πράγμα που σημαίνει ότι δεν χρειάζεται να αφιερώσετε επιπλέον χρόνο στη διαχείριση των αλλαγών και των αποτελεσμάτων που έχουν στα δεδομένα.

Το React χρησιμοποιεί μια γλώσσα προγραμματισμού με πλήρη χαρακτηριστικά (JavaScript) για την κατασκευή επαναλαμβανόμενων ή υπό όρους στοιχείων DOM, κάτι που είναι σίγουρα ένα τεράστιο πλεονέκτημα. Δεν χρειάζεται να βασίζεστε σε πρότυπο για να αυτοματοποιήσετε τη δημιουργία επαναλαμβανόμενων στοιχείων HTML και DOM.

Με το React, ο ίδιος κώδικας μπορεί να εκτελεστεί τόσο στον διακομιστή όσο και στο πρόγραμμα περιήγησης. Εξαιτίας αυτών, οι περισσότεροι προγραμματιστές αποκαλούν το React «η καρδιά και η ψυχή της στοίβας του MERN».

4. Node JS

Αρχικά δημιουργημένο για το Google Chrome και αργότερα σε ανοιχτό κώδικα, το Node JS είναι ένα περιβάλλον JavaScript χρόνου εκτέλεσης πολλαπλών πλατφορμών που χρησιμοποιείται για την εκτέλεση κώδικα JavaScript εκτός προγράμματος περιήγησης.

Όπως γνωρίζετε, η JavaScript χρησιμοποιήθηκε αρχικά για δέσμες ενεργειών στο front-end, αλλά το Node JS επέτρεψε στους προγραμματιστές να το χρησιμοποιούν για να γράφουν εργαλεία γραμμής εντολών και σενάρια back-end με σκοπό τη δημιουργία δυναμικού περιεχομένου ιστοσελίδας προτού σταλεί η σελίδα στο χρήστη φυλλομετρητής.

Το Node JS σχεδιάστηκε με την ιδέα να επιτρέπει στους προγραμματιστές να δημιουργούν επεκτάσιμες εφαρμογές δικτύου.

1.2 ΠΕΡΙΓΡΑΦΗ ΠΡΟΒΛΗΜΑΤΟΣ

Στο τομέα των ξενοδοχείων και στην online κράτηση τους , είναι πολλές οι εφαρμογές που θα κοιτάξουμε αμέσως για να κάνουμε μία κράτηση. Βέβαια πολλές φορές αυτές οι εφαρμογές έχουν αρκετά ελαττώματα. Λόγω της τεράστιας κλίμακας των χρηστών, μπορούν να γίνουν λάθη που οδηγούν σε απώλεια χρημάτων και κακής αξιολόγησης. Ενώ οι third-party ιστότοποι κρατήσεων μπορούν να αποτελέσουν χρήσιμο σημείο εκκίνησης για την έρευνα για τα ξενοδοχεία που είναι διαθέσιμα σε ορισμένους προορισμούς και σε ποια σημεία τιμών, αυτός είναι πιθανώς ο βαθμός στον οποίο θα πρέπει να χρησιμοποιηθούν. Η απευθείας κράτηση σε ξενοδοχείο είναι το ασφαλέστερο και καλύτερο στοίχημα για λόγους που κυμαίνονται από τη λήψη ειδικής μεταχείρισης έως την άμεση προσφυγή σε περίπτωση που κάτι πάει στραβά.

Υπευθυνότητα:

Όταν η σχέση κράτησης είναι μόνο μεταξύ του επισκέπτη και του ξενοδοχείου, είναι καθαρή και εύκολη γιατί εμπλέκονται μόνο δύο μέρη: ο πελάτης και η επιχείρηση. Όταν εμπλέκεται ένα τρίτο μέρος (ιστότοπος κρατήσεων) είναι όταν τα πράγματα μπορεί να γίνουν ακατάστατα. Για παράδειγμα, μερικές φορές οι κρατήσεις δεν υποβάλλονται σε επεξεργασία και ο ιστότοπος και το ξενοδοχείο μπορούν να μεταφέρουν την ευθύνη πέρα δώθε. Αυτό μπορεί ενδεχομένως να αφήσει τον πελάτη με το λάθος δωμάτιο ή χωρίς καθόλου δωμάτιο και υπάρχει μικρή ελπίδα για προσφυγή.

Αντιστοίχιση τιμών:

Το μεγάλο και πιθανότατα μοναδικό με την κράτηση μέσω ιστότοπων όπως για παράδειγμα το Kayak και το Travelocity είναι ότι φαίνεται να αυξάνουν τις χαμηλότερες τιμές, ακόμα κι αν είναι μόλις λίγα χρήματα. Ωστόσο, αυτό που δεν είναι ακόμη γνωστό (αρκετά) είναι ότι τα περισσότερα ξενοδοχεία - ειδικά τα ξενοδοχεία της αλυσίδας - θα προσπαθήσουν να κάνουν την απευθείας κράτησή και να ανταποκριθούν σε οποιαδήποτε τιμή εκεί έξω.

Επιλογή δωματίου:

Τα ξενοδοχεία διαθέτουν μόνο ένα συγκεκριμένο τμήμα δωματίων για πλατφόρμες τρίτων. Προφανώς, τα ξενοδοχεία διατηρούν τα καλύτερα δωμάτια και τη μεγαλύτερη ποικιλία για να πουλήσουν απευθείας. Αυτό μπορεί να κάνει μεγάλη διαφορά όταν πρόκειται για ένα ακίνητο όπου ένα τυπικό δωμάτιο μπορεί να ποικίλλει πολύ σε μέγεθος, κάτι που είναι ιδιαίτερα κοινό σε παλιά ακίνητα που κατασκευάστηκαν πριν, όταν τα δωμάτια (και, επίσης, οι άνθρωποι) ήταν μικρότερα. Μια μικρή διαδικτυακή έρευνα μπορεί συχνά να αποκαλύψει ποιος τύπος δωματίου μπορεί πραγματικά να συνδέεται με την κράτηση μέσω ιστότοπου τρίτου μέρους, όπως αυτό το μικροκαμωμένο δωμάτιο στη Βοστώνη από την Priceline.

Περισσότερη ευελιξία:

Όταν γίνεται απευθείας κράτηση, τα ξενοδοχεία θα είναι συχνά πιο πρόθυμα και πιο ικανά να βοηθήσουν εάν χρειαστεί κάποια αλλαγή της ημερομηνίας ή ακόμα και να ακυρωθεί, επειδή δεν δεσμεύονται από κάποια συμφωνία τρίτων. Η κράτηση δωματίου σε ξενοδοχείο μέσω ιστότοπου τρίτου μέρους μπορεί ουσιαστικά να δημιουργήσει και να κλειδώσει ένα συμβόλαιο διαμονής, το οποίο μπορεί να περιορίσει το δωμάτιο κουνίσματος για τον επισκέπτη και το προσωπικό του ξενοδοχείου.

Αυτό σημαίνει επίσης ότι εάν κάνει κάποιος απευθείας κράτηση, το ξενοδοχείο μπορεί να κάνει περισσότερα για να αποζημιώσει έναν επισκέπτη - όπως να παραιτηθεί από το κόστος μιας ολόκληρης διαμονής - σε περίπτωση που κάτι πάει στραβά ή δεν είναι έτοιμο.

Καλύτερη εξυπηρέτηση, συν προνόμια:

Το προσωπικό του ξενοδοχείου μπορεί να μην βγει και να το πει, αλλά έχει την τάση να αντιμετωπίζει καλύτερα τους επισκέπτες όταν αυτοί οι επισκέπτες τους δίνουν περισσότερα χρήματα. Οι απευθείας κρατήσεις είναι πιο κερδοφόρες γι' αυτούς από αυτές που πραγματοποιούνται μέσω ιστοτόπων τρίτων που περιορίζονται και το προσωπικό γενικά μπορεί να δει απευθείας στο σύστημα υπολογιστή πώς έγινε η

κράτηση ενός δωματίου. Ελέγχουν σίγουρα τον υπολογιστή όταν ένας επισκέπτης δίνει μια υπόδειξη σχετικά με μια αναβάθμιση.

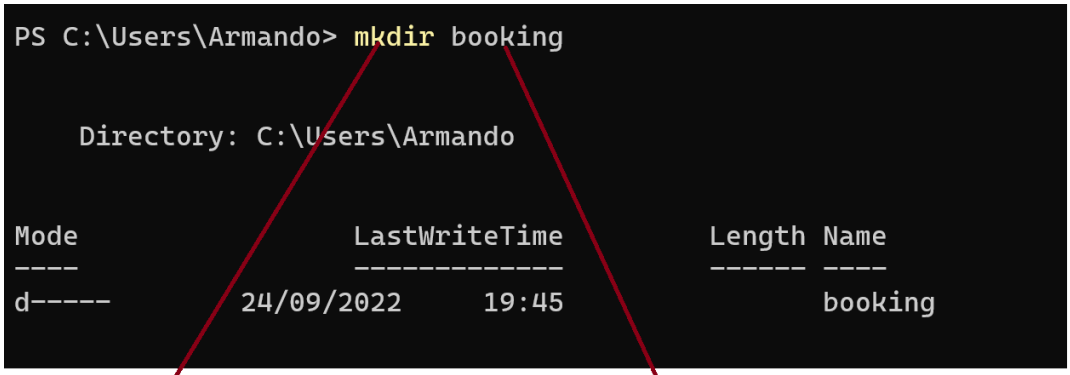
Επίσης, κάποιος που κάνει κράτηση δωματίου σε έναν ιστότοπο όπως η Expedia είναι η καλύτερη επιλογή για «περπάτημα», δηλαδή όταν ένας επισκέπτης στέλνεται σε άλλο κατάλυμα λόγω υπερκράτησης (μια συνηθισμένη πρακτική) σύμφωνα με τον Jacob Tomsky, συγγραφέα του Heads in Beds: Ένα απερίσκεπτο υπόμνημα ξενοδοχείων, φασαριών και λεγόμενης φιλοξενίας.

Τα ξενοδοχεία έχουν επίσης τη δύναμη και το απόθεμα να υπερβούν τις τυπικές κρατήσεις ξενοδοχείων για να δημιουργήσουν ειδικά πακέτα ή να προσφέρουν προνόμια όπως το δωρεάν πρωινό. Για παράδειγμα, η Starwood τρέχει αυτήν τη στιγμή μια προσφορά για τα θέρετρα του στο Μεξικό και στον Παναμά που περιλαμβάνει δωρεάν διανυκτερεύσεις, πιστώσεις για φαγητό και ποτό και αναβαθμίσεις. Οι πλατφόρμες τρίτων λειτουργούν μόνο με τα βασικά, τα οποία για τα ξενοδοχεία είναι μόνο δωμάτια.

Έχοντας αναφέρει τα παραπάνω, ας ξεκινήσουμε το στήσιμο της δικίας μας πλατφόρμας .

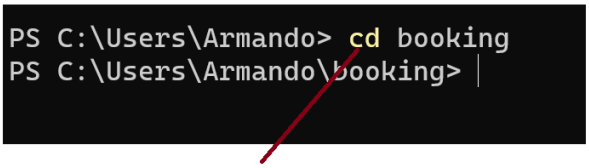
2 ΕΓΚΑΤΑΣΤΑΣΗ ΕΦΑΡΜΟΓΗΣ-BACKEND

Το πρώτο πράγμα που κάνουμε είναι να ανοίξουμε την γραμμή εντολών(terminal) του υπολογιστή μας. Ελέγχουμε εάν είναι εγκατεστημένο το node στον υπολογιστή μας με την εντολή “node -v”. Αν πάρουμε σαν αποτέλεσμα κάποια έκδοση του Node JS σημαίνει ότι υπάρχει στον υπολογιστή μας. Αν όχι τότε μπαίνουμε στην επίσημη σελίδα του https://nodejs.org/en/download/ και το κατεβάζουμε. Δημιουργούμε έναν καινούριο φάκελο με όποια ονομασία θέλουμε εμείς χρησιμοποιώντας την εντολή “mkdir folder” και τον ανοίγουμε με την εντολή “cd folder”.



Εντολή δημιουργίας φακέλου.

Όνομα του φακέλου μας



Εντολή που μας βοηθάει στην πλοήγηση των φακέλων

2.1 ΕΓΚΑΤΑΣΤΑΣΗ ΤΟΥ CLIENT

Για την δημιουργία ενός client χρησιμοποιούμε την εντολή “npm create-react-app client”. Μόλις αυτό γίνει, ανοίγουμε τον φάκελο μας και μέσα σε αυτόν πληκτρολογούμε την εντολή “npm start”. Έτσι το πρόγραμμα μας έχει ξεκινήσει να τρέχει και τώρα είμαστε έτοιμοι να κάνουμε αλλαγές. Για να έχουμε ένα καθαρό πρόγραμμα αποό την αρχή διαγράφουμε κάποια αρχεία που δεν θα τα χρησιμοποιήσουμε όπως και κώδικα.

Τα αρχεία που διαγράφουμε είναι τα εξής:

App.css
App.test
index.js(κώδικα)

logo.svg

2.1.1 ΧΡΗΣΗ VISUAL STUDIO CODE

Για να ξεκινήσουμε να γράφουμε κώδικα χρειαζόμαστε ένα πρόγραμμα που θα μας επιτρέπει αυτή την λειτουργία. Για το συγκεκριμένο project θα δουλέψουμε με το Visual Studio Code.

Τι είναι όμως το Visual Studio Code;

Το Visual Studio Code είναι ένα βελτιωμένο πρόγραμμα επεξεργασίας κώδικα με υποστήριξη για λειτουργίες ανάπτυξης όπως εντοπισμός σφαλμάτων, εκτέλεση εργασιών και έλεγχος έκδοσης. Στοχεύει να παρέχει μόνο τα εργαλεία που χρειάζεται ένας προγραμματιστής για έναν γρήγορο κύκλο δημιουργίας κώδικα-εντοπισμού σφαλμάτων και αφήνει πιο σύνθετες ροές εργασίας σε πληρέστερα IDE που εμφανίζονται, όπως το Visual Studio IDE.

Γιατί να χρησιμοποιήσουμε Visual Studio Code και όχι κάποιο άλλο editor;

Φιλικό προς νέους προγραμματιστές

Το Visual Studio Code επισημαίνει λέξεις-κλειδιά στον κώδικά με διαφορετικά χρώματα για την βοήθεια της αναγνώρισης εύκολων μοτίβων κωδικοποίησης και για γρηγορότερη εκμάθηση. Μπορείτε επίσης να ένας χρήστης να επωφεληθεί από λειτουργίες όπως το IntelliSense και το Peek Definition, που βοηθούν κατανόηση των λειτουργιών.

Διόρθωση σφαλμάτων

Καθώς κωδικοποιείτε, ο κώδικας του Visual Studio δίνει προτάσεις για να ολοκληρώσετε γραμμές κώδικα και γρήγορες επιδιορθώσεις για συνηθισμένα λάθη. Μπορείτε επίσης να χρησιμοποιήσετε το πρόγραμμα εντοπισμού σφαλμάτων στο VS Code για να περάσετε σε κάθε γραμμή κώδικα και να κατανοήσετε τι συμβαίνει.

2.1.2 ΔΙΑΦΟΡΑ NPM ΚΑΙ NPX

Όπως είδαμε και πρίν για να εγκαταστήσουμε τον client στο πρόγραμμα μας χρησιμοποιήσαμε την εντολή στον terminal “npm”. Ύπάρχει και ένας άλλος τρόπος για την εκτέλεση κάποιον εντολών που ονομάζεται “npx”. Ποια είναι η διαφορά όμως αυτών των 2;

NPM

Το NPM είναι ένας διαχειριστής πακέτων για τη γλώσσα προγραμματισμού JavaScript. Είναι ένα έργο ανοιχτού κώδικα που ξεκίνησε από τον Isaac Schlueter το 2009 και έκτοτε χρησιμοποιείται ως ο προεπιλεγμένος διαχειριστής πακέτων για το Node.js. Σε αυτό το μέρος, συζητάμε τι είναι το NPM και πώς λειτουργεί για να μας βοηθήσει να μάθουμε περισσότερα σχετικά με το γιατί είναι τόσο δημοφιλές εργαλείο:

1. Προσαρμόστε πακέτα κώδικα για τα προγράμματά σας ή ενσωματώστε τα πακέτα ως έχουν.
2. Εγκαταστήστε ανεξάρτητο λογισμικό που μπορείτε να χρησιμοποιήσετε άμεσα.
3. Λειτουργία πακέτων χρησιμοποιώντας npx χωρίς λήψη.
4. Μοιραστείτε τον κωδικό με οποιονδήποτε χρήστη του npm, ανά πάσα στιγμή.
5. Περιορίστε μεμονωμένους προγραμματιστές στον κώδικα.
6. Δημιουργήστε οργανισμούς για να συντονίζουν τη διαχείριση, την κωδικοποίηση και τους δημιουργούς πακέτων.
7. Χρησιμοποιώντας οργανισμούς, διαμορφώστε εικονικές ομάδες.
8. Διαχειρίζονται πολλές εκδόσεις κώδικα και εξαρτήσεις κώδικα.
9. Αναβαθμίστε εύκολα το λογισμικό αναβαθμίζοντας τον υποκείμενο κώδικα.
10. Ανακαλύψτε διαφορετικούς τρόπους επίλυσης του ίδιου παζλ.
11. Βρείτε άλλους προγραμματιστές που εργάζονται σε θέματα και έργα που είναι παρόμοια.

NPX

Το Npx μπορεί να χρησιμοποιηθεί για να εκτελέσει γρήγορα και εύκολα όλα τα πακέτα npm. Παρέχει επίσης μια εναλλακτική λύση στη χρήση των μεγάλων εντολών NPX ή NPM που χρησιμοποιούσαμε μέχρι τώρα.

Το ωραίο με το npx είναι ότι εκτελείται ως ξεχωριστή διαδικασία, πράγμα που σημαίνει ότι τυχόν σφάλματα δεν εμποδίζουν άλλες εργασίες να λειτουργούν όπως αναμένεται. Ας δούμε μερικά από τα χαρακτηριστικά του NPX:

1. Απλή και γρήγορη εκτέλεση τοπικών εντολών.
2. Επιτρέπει την εκτέλεση εντολών χωρίς να τις εγκαταστήσετε πρώτα.
3. Εκτελεί κάποιο κώδικα χρησιμοποιώντας διαφορετική έκδοση Node.
4. Εκτελεί αυθαίρετα αποσπάσματα κώδικα απευθείας από μια διεύθυνση URL.
5. Μπορείτε να εκτελέσετε εντολές npx με \$npm_ Μεταβλητές Περιβάλλοντος.
6. Μπορείτε επίσης να εκτελέσετε Κώδικα από απομακρυσμένο κλάδο GitHub με npx.
7. Το NPX χρησιμοποιείται επίσης για την εκτέλεση κώδικα από μια ουσία GitHub.

Με τα παραπάνω καταλαβαίνουμε πώς το npm είναι ένας διαχειριστής πακέτων Node.js σκοπός του οποίου είναι να χειρίζεται αυτοματοποιημένες εξαρτήσεις και πακέτα.
Αυτό σημαίνει ότι μπορούμε να καθορίσουμε όλες τις εξαρτήσεις (πακέτα) μας για ένα έργο στο αρχείο package.JSON και όταν κάποιος πρέπει να εγκαταστήσει τις εξαρτήσεις, μπορεί απλώς να εκτελέσει την εγκατάσταση npm και θα λάβει αυτό που ακριβώς ψάχνει.

Προσφέρει επίσης έκδοση εκδόσεων, δηλαδή μας επιτρέπει να ορίσουμε ποιες εκδόσεις των εξαρτήσεων βασίζεται το έργο, ώστε να μπορούμε να αποφύγουμε

αλλαγές από την καταστροφή του έργου μας ή τη χρήση της προτιμώμενης έκδοσης στο μεγαλύτερο μέρος.

Ενώ, το npx είναι ένα εργαλείο εκτέλεσης πακέτου Node και συνοδεύεται από npm ξεκινώντας από την έκδοση npm5.2 και μετά.

2.2 ΕΓΚΑΤΑΣΤΑΣΗ ΤΟΥ ROUTING

Ας υλοποιήσουμε το σύστημα routing που θα μας βοηθήσει στην πλοήγηση πολλαπλών σελίδων.

Για την ώρα θα χρειαστούμε 2 σελίδες. Μία login και μία register σελίδα. Δημιουργούμε μέσα στον φάκελο src έναν φάκελο που τον ονομάζουμε booking. Έπειτα φτιάχνουμε ένα νέο αρχείο που το ονομάζουμε Home.js. Αφού κάνουμε αυτά τα βήματα φτιάχνουμε και άλλον έναν φάκελο με όνομα auth, από το authorization. Ο λόγος που ονομάζεται έτσι ο φάκελος μας είναι γιατί μέσα σε αυτόν πρέπει να μπουν αρχεία που θα χρειάζεται ο χρήστης πιστοποίηση των στοιχείων του. Μέσα σε αυτόν τον φάκελο φτιάχνουμε τα αρχεία που θέλουμε (Login.js & Register.js). Για να βάλουμε το routing στην πράξη πρέπει να εγκαταστήσουμε τα κατάλληλα πακέτα.

Παρακάτω θα δούμε ποια εντολή θα χρειαστούμε για την εγκατάσταση των πακέτων αυτών.

npm i react-router-dom (εγκατάσταση του router)

Αφού γίνει αυτό κάνουμε import στο πρόγραμμά μας τα συστατικά για να δουλέψει το routing

import {BrowserRouter, Switch, Route} from "react-router-dom";

Στη συνέχεια καταχωρούμε ως components τα παρακάτω

<BrowserRouter>

<Switch>

<Route exact path="/" component={Home} />

**Το exact path είναι σημαντικό γιατί όλα θα μας καθοδηγήσουν στην αρχική σελίδα λόγω του “/” .

2.2.1 ΤΙ ΕΙΝΑΙ ΤΟ ROUTING

Πιο πάνω είδαμε πώς βάζουμε το routing στο πρόγραμμα μας. Όμως τι είναι ακριβώς το routing;

Το React Router είναι μια τυπική βιβλιοθήκη για δρομολόγηση στο React. Επιτρέπει την πλοήγηση μεταξύ προβολών διαφόρων στοιχείων σε μια εφαρμογή React, επιτρέπει την αλλαγή της διεύθυνσης URL του προγράμματος περιήγησης και διατηρεί τη διεπαφή χρήστη σε συγχρονισμό με τη διεύθυνση URL.

2.2.2 ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ

Client-side Routing

Η δρομολόγηση από την πλευρά του πελάτη γίνεται αποκλειστικά από JavaScript στη σελίδα. Κάθε φορά που ένας χρήστης κάνει κλικ σε έναν σύνδεσμο, η γραμμή URL αλλάζει και μια διαφορετική προβολή αποδίδεται στη σελίδα. Αυτή η προβολή μπορεί να είναι οτιδήποτε—JSX ή HTML. Οι εφαρμογές μιας σελίδας δίνουν μια ομαλή αίσθηση πλοήγησης καθώς δεν ανανεώνουν ολόκληρη τη σελίδα όταν εκτελείται μια διαδρομή. Ακόμη και όταν υποβάλλεται αίτημα στον διακομιστή για ανάκτηση δεδομένων, φαίνεται μόνο σαν να αποδίδονται στατικές σελίδες HTML στο frontend. Έτσι, οι εφαρμογές μιας σελίδας είναι άμεσοι δικαιούχοι της δρομολόγησης από την πλευρά του πελάτη και αυτός είναι ένας σημαντικός λόγος για την αυξανόμενη δημοτικότητά τους και την παροχή εξαιρετικής εμπειρίας χρήστη.

Ας δούμε τα πλεονεκτήματα και τα μειονεκτήματα της δρομολόγησης από την πλευρά του πελάτη.

Πλεονεκτήματα

- Η δρομολόγηση μεταξύ των στοιχείων είναι γρήγορη καθώς ο όγκος των δεδομένων που αποδίδει είναι μικρότερος. Τα υπόλοιπα δεδομένα αποδίδονται από το DOM και ακόμη και όταν υπάρχουν πολλοί τόνοι HTML και CSS για απόδοση, το DOM χειρίζεται αυτό το τμήμα εν ριπή οφθαλμού. Με τη χρήση αργής φόρτωσης, οποιαδήποτε καθυστέρηση στην απόδοση HTML αντισταθμίζεται.
- Για καλύτερη εμπειρία χρήστη, τα κινούμενα σχέδια και οι μεταβάσεις μπορούν εύκολα να εφαρμοστούν κατά την εναλλαγή μεταξύ διαφορετικών στοιχείων.
- Δίνει μια πραγματική αίσθηση μιας μονοσέλιδης εφαρμογής σε δράση. Δεν αποδίδονται ξεχωριστές σελίδες και η τρέχουσα σελίδα δεν ανανεώνεται για να φορτώσει μια νέα προβολή.

Μειονεκτήματα

- Ο αρχικός χρόνος φόρτωσης είναι αρκετά μεγάλος καθώς όλες οι διαδρομές, τα στοιχεία και το HTML πρέπει να φορτωθούν ταυτόχρονα κατά την πρώτη

προσάρτηση της εφαρμογής. Ολόκληρος ο ιστότοπος ή η εφαρμογή ιστού πρέπει να φορτωθεί με το πρώτο αίτημα.

- Υπάρχει περιττός χρόνος λήψης δεδομένων για μη χρησιμοποιήσιμες προβολές που δεν μπορεί να προβλεφθεί στην πρώτη απόδοση της εφαρμογής.
- Απαιτεί γενικά μια εξωτερική βιβλιοθήκη, που σημαίνει περισσότερο κώδικα και μεγαλύτερη εξάρτηση από εξωτερικά πακέτα, σε αντίθεση με τη δρομολόγηση από την πλευρά του διακομιστή.
- Η δρομολόγηση και η απόδοση από την πλευρά του πελάτη μετατρέπουν το JavaScript σε HTML, καθιστώντας την ανίχνευση των μηχανών αναζήτησης λιγότερο βελτιστοποιημένη.

Client-side Routing σε React

Το React αποδίδει τις κατάλληλες πληροφορίες για το DOM χρησιμοποιώντας τη δομή των στοιχείων του. Η δρομολόγηση από την πλευρά του πελάτη στο React βοηθά στη διατήρηση της απρόσκοπτης εμπειρίας χρήστη που υπόσχεται μια τυπική εφαρμογή μιας σελίδας. Αυτό επιτυγχάνεται μέσω μιας εξωτερικής βιβλιοθήκης React που ονομάζεται React Router.

Client-side Routing σε React

Το React Router χρησιμοποιεί δυναμική δρομολόγηση για να διασφαλίσει ότι η δρομολόγηση επιτυγχάνεται όπως ζητείται από τον χρήστη. Αυτό σημαίνει επίσης ότι όλα τα απαιτούμενα στοιχεία αποδίδονται επίσης χωρίς φλας λευκής οθόνης ή επαναφόρτιση σελίδας.

2.3 ΔΗΜΙΟΥΡΓΙΑ NAVIGATION

Σε αυτό το σημείο θα προσθέσουμε το navigation για να μπορούμε να μεταβούμε από την μία σελίδα στην άλλη με μεγαλύτερη ευκολία.

Ο κώδικας με τον οποίο θα ξεκινήσουμε είναι ο παρακάτω.

```
const TopNav = () => {
        return (
        )};
```

||Curly brace {} με return () βάζουμε όταν θέλουμε να γράψουμε πολλές γραμμές κώδικα.

||Μεταβλητή με () και χωρίς return όταν θέλουμε να κάνουμε ένα block κώδικα.

Στην συγκεκριμένη περίπτωση κάνουμε

```
const TopNav = () => (
        <div className="nav bg-light d-flex justify-content-between"></div>
        <Link className="nav-link" to="/">Home</Link>
```

Link είναι ουσιαστικά το “a”(HTML) και το "to" είναι το href(HTML) το οποίο πρέπει να το κάνουμε import.

Το TopNav είναι μεταβλητή που δεν την κάνουμε excecute. Για να γίνει excecute πρέπει να τη βάλουμε κάτω από το BrowserRouter μέσα σε { }. Επίσης όταν θέλουμε να κάνουμε excecute άμεσα, γράφουμε {TopNav()}. Όταν βάζουμε () χωρίς κάτι μέσα γίνεται αυτόματο excecute.

Αν τρέξουμε αυτό το κομμάτι κώδικα θα πάρουμε το εξής μήνυμα “error ./src/App.js”
“Attempted import error: 'react-router-dom' does not contain a default export (imported as 'Link').”
Αυτό σημαίνει ότι το Link δεν είναι default export. Όλα τα default export τα κάνουμε import κανονικά. Όλα τα άλλα export που δεν είναι default τα κάνουμε import με {}.

Τέλος κάνουμε το TopNav component και βάζουμε μέσα το function(μεταβλητή) που έχουμε έτοιμο και το αλλάζουμε από {TopNav()} σε <TopNav />.

2.3.1 ΤΙ ΕΙΝΑΙ ΤΟ BOOTSTRAP

Πιο πάνω είδαμε κάποιο κώδικα της bootstrap στο πρόγραμμα μας. Όμως τι είναι ακριβώς η bootstrap;

Το Bootstrap είναι ένα δωρεάν, ανοιχτού κώδικα πλαίσιο ανάπτυξης front-end για τη δημιουργία ιστότοπων και εφαρμογών ιστού. Σχεδιασμένο για να επιτρέπει την αποκριτική ανάπτυξη ιστοτόπων που προορίζονται για κινητά, το Bootstrap παρέχει μια συλλογή σύνταξης για σχέδια προτύπων.

Ως πλαίσιο, το Bootstrap περιλαμβάνει τα βασικά για την αποκριτική ανάπτυξη ιστού, επομένως οι προγραμματιστές χρειάζεται μόνο να εισαγάγουν τον κώδικα σε ένα προκαθορισμένο σύστημα πλέγματος. Το πλαίσιο Bootstrap είναι χτισμένο σε γλώσσα σήμανσης υπερκειμένου (HTML), σε φύλλα στυλ (CSS) και JavaScript. Οι προγραμματιστές ιστού που χρησιμοποιούν το Bootstrap μπορούν να δημιουργήσουν ιστότοπους πολύ πιο γρήγορα χωρίς να ξοδεύουν χρόνο ανησυχώντας για βασικές εντολές και λειτουργίες.

Σε τι χρησιμοποιείται το Bootstrap;

Το Bootstrap κάνει το responsive web design πραγματικότητα. Επιτρέπει σε μια ιστοσελίδα ή εφαρμογή να ανιχνεύει το μέγεθος και τον προσανατολισμό της οθόνης του επισκέπτη και να προσαρμόζει αυτόματα την οθόνη ανάλογα. Η προσέγγιση «πρώτα το κινητό» προϋποθέτει ότι τα smartphone, τα tablet και οι εφαρμογές για κινητές συσκευές για συγκεκριμένες εργασίες είναι τα κύρια εργαλεία των εργαζομένων για την ολοκλήρωση της εργασίας. Το Bootstrap αντιμετωπίζει τις απαιτήσεις αυτών των τεχνολογιών στο σχεδιασμό και περιλαμβάνει στοιχεία διεπαφής χρήστη, διατάξεις, εργαλεία JavaScript και το πλαίσιο υλοποίησης. Το λογισμικό διατίθεται προ μεταγλωττισμένο ή ως πηγαίος κώδικας.

Ο Mark Otto και ο Jacob Thornton ανέπτυξαν το Bootstrap στο Twitter για να βελτιώσουν τη συνοχή των εργαλείων που χρησιμοποιούνται στον ιστότοπο και να μειώσουν τη συντήρηση. Το λογισμικό ήταν παλαιότερα γνωστό ως Twitter Blueprint και μερικές φορές αναφέρεται ως Twitter Bootstrap.

Τι είναι το bootstrap σε έναν υπολογιστή;

Στους υπολογιστές, ο όρος bootstrap σημαίνει την εκκίνηση ή τη φόρτωση ενός προγράμματος σε έναν υπολογιστή χρησιμοποιώντας ένα πολύ μικρότερο αρχικό πρόγραμμα για φόρτωση στο επιθυμητό πρόγραμμα, το οποίο είναι συνήθως ένα λειτουργικό σύστημα.

Τι είναι το bootstrap στη CSS;

Το πιο δημοφιλές πλαίσιο CSS για την ανάπτυξη ιστοτόπων με ανταπόκριση και κινητά είναι το Bootstrap. Η πιο πρόσφατη έκδοση είναι το Bootstrap 5.

Τι είναι το bootstrap CDN;

Οι ιστοσελίδες με bootstrap συχνά χρειάζονται αύξηση της ταχύτητας. Ένα δίκτυο παράδοσης περιεχομένου βοηθά στην επίλυση αυτού του ζητήματος και παρέχει στατικό περιεχόμενο στους χρήστες πιο γρήγορα. Είναι η καλύτερη προσέγγιση για την ταυτόχρονη βελτίωση της αφοσίωσης των χρηστών και της ταχύτητας φόρτωσης της σελίδας.

Τι είναι το bootstrapping σε γενικές γραμμές;

Στον φυσικό κόσμο, το bootstrap είναι ένας μικρός ιμάντας ή θηλιά στο πίσω μέρος μιας δερμάτινης μπότας που επιτρέπει να τραβιέται η μπότα. Σε γενική χρήση, το bootstrapping αξιοποιεί μια μικρή αρχική προσπάθεια σε κάτι μεγαλύτερο και πιο σημαντικό. Η αλληγορία, «τράβηξε τον εαυτό σου από τις μπότες σου», σημαίνει να πετύχεις την επιτυχία από μια μικρή αρχή.

2.4 REDUX

Θα χρησιμοποιήσουμε το redux ως παγκόσμια κατάσταση για την εφαρμογή μας.

- Θα χρειαστεί να αποθηκεύσουμε τις συνδεδεμένες πληροφορίες χρήστη, το διακριτικό JWT σε κατάσταση redux, ώστε να μπορούμε να έχουμε εύκολη πρόσβαση σε αυτές τις πληροφορίες σε πολλές διαφορετικές σελίδες αργότερα.
- npm και redux react-redux redux-devtools-extension
- Φροντίζουμε να εγκαταστήσουμε επίσης την επέκταση chrome για redux.

2.4.1 ΓΙΑΤΙ REDUX

Με τον αριθμό των εργαλείων και των βιβλιοθηκών που υπάρχουν για την ανάπτυξη ιστού, μπορεί να μην είναι το πιο σοφό πράγμα να μεταβούμε σε κάθε νέο χωρίς να κατανοήσουμε πραγματικά τα οφέλη του ή γιατί πρέπει να το χρησιμοποιήσουμε.

Το Redux δεν είναι νέο, αλλά παραμένει αρκετά δημοφιλές. Παρακάτω, θα δείξουμε τι είναι το Redux, γιατί πρέπει να το χρησιμοποιούμε και πώς λειτουργεί.

Τι είναι REDUX

Το Redux είναι ένα προβλέψιμο κοντέινερ κατάστασης που έχει σχεδιαστεί για να σας βοηθά να γράφετε εφαρμογές JavaScript που συμπεριφέρονται με συνέπεια σε περιβάλλοντα πελάτη, διακομιστή και εγγενή περιβάλλοντα και είναι εύκολο να δοκιμαστούν.

Ενώ χρησιμοποιείται κυρίως ως εργαλείο διαχείρισης κατάστασης με το React, μπορείτε να το χρησιμοποιήσετε με οποιοδήποτε άλλο πλαίσιο ή βιβλιοθήκη JavaScript. Είναι ελαφρύ στα 2 KB (συμπεριλαμβανομένων των εξαρτήσεων), επομένως δεν χρειάζεται να ανησυχείτε για το ότι θα μεγαλώσει το μέγεθος του στοιχείου της εφαρμογής σας.

Με το Redux, η κατάσταση της εφαρμογής σας διατηρείται σε ένα κατάστημα και κάθε στοιχείο μπορεί να έχει πρόσβαση σε οποιαδήποτε κατάσταση χρειάζεται από αυτό το κατάστημα.

Πότε να χρησιμοποιήσετε το REDUX

Τον τελευταίο καιρό μια από τις μεγαλύτερες συζητήσεις στον κόσμο του frontend ήταν για το Redux. Λίγο μετά την κυκλοφορία του, το Redux έγινε ένα από τα πιο καυτά θέματα συζήτησης. Πολλοί το ευνόησαν ενώ άλλοι επεσήμαναν ζητήματα.

Το Redux σάς επιτρέπει να διαχειρίζεστε την κατάσταση της εφαρμογής σας σε ένα μόνο μέρος και να διατηρείτε τις αλλαγές στην εφαρμογή σας πιο προβλέψιμες και

ανιχνεύσιμες. Διευκολύνει τον συλλογισμό σχετικά με τις αλλαγές που συμβαίνουν στην εφαρμογή σας. Αλλά όλα αυτά τα οφέλη συνοδεύονται από συμβιβασμούς και περιορισμούς. Κάποιος μπορεί να αισθανθεί ότι προσθέτει κώδικα λέβητα, καθιστώντας τα απλά πράγματα λίγο συντριπτικά. αλλά αυτό εξαρτάται από τις αποφάσεις της αρχιτεκτονικής.

Μια απλή απάντηση σε αυτή την ερώτηση είναι ότι θα συνειδητοποιήσετε μόνοι σας πότε χρειάζεστε το Redux. Εάν εξακολουθείτε να μπερδεύεστε ως προς το αν το χρειάζεστε, δεν το χρειάζεστε. Αυτό συμβαίνει συνήθως όταν η εφαρμογή σας μεγαλώνει στην κλίμακα όπου η διαχείριση της κατάστασης της εφαρμογής γίνεται ταλαιπωρία. και αρχίζετε να ψάχνετε να το κάνετε εύκολο και απλό.

Πώς λειτουργεί το REDUX

Ο τρόπος που λειτουργεί το Redux είναι απλός. Υπάρχει ένα κεντρικό κατάστημα που κρατά όλη την κατάσταση της εφαρμογής. Κάθε στοιχείο μπορεί να έχει πρόσβαση στην αποθηκευμένη κατάσταση χωρίς να χρειάζεται να στέλνει στηρίγματα από το ένα στοιχείο στο άλλο.

Υπάρχουν τρία δομικά μέρη: ενέργειες, αποθήκευση και μειωτήρες. Ας συζητήσουμε εν συντομία τι κάνει το καθένα από αυτά. Αυτό είναι σημαντικό γιατί σας βοηθούν να κατανοήσετε τα οφέλη του Redux και τον τρόπο χρήσης του. Θα εφαρμόσουμε ένα παρόμοιο παράδειγμα με το στοιχείο σύνδεσης παραπάνω, αλλά αυτή τη φορά στο Redux.

Γιατί να χρησιμοποιήσετε το REDUX;

Όταν χρησιμοποιείτε το Redux με το React, οι καταστάσεις δεν θα χρειάζεται πλέον να ανυψώνονται. Αυτό σας διευκολύνει να εντοπίσετε ποια ενέργεια προκαλεί οποιαδήποτε αλλαγή.

Όπως μπορείτε να δείτε στο παραπάνω παράδειγμα, το στοιχείο δεν χρειάζεται να παρέχει καμία κατάσταση ή μέθοδο για τα θυγατρικά του στοιχεία για κοινή χρήση δεδομένων μεταξύ τους. Όλα τα χειρίζεται η Redux. Αυτό απλοποιεί σημαντικά την εφαρμογή και διευκολύνει τη συντήρησή της.

Αυτός είναι ο κύριος λόγος για τον οποίο πρέπει να χρησιμοποιήσετε το Redux.

2.4.2 ΧΡΗΣΗ JWT TOKEN

Το JWT γίνεται όλο και πιο δημοφιλές για την ασφάλεια των API. Τι είναι όμως ακριβώς το JWT; Και πώς λειτουργεί;

Τι είναι JWT;

Το JWT, ή JSON Web Token, είναι ένα ανοιχτό πρότυπο που χρησιμοποιείται για την κοινή χρήση πληροφοριών ασφαλείας μεταξύ δύο μερών — πελάτη και διακομιστή.

Κάθε JWT περιέχει κωδικοποιημένα αντικείμενα JSON, συμπεριλαμβανομένου ενός συνόλου αξιώσεων. Τα JWT υπογράφονται χρησιμοποιώντας έναν κρυπτογραφικό αλγόριθμο για να διασφαλιστεί ότι οι αξιώσεις δεν μπορούν να τροποποιηθούν μετά την έκδοση του διακριτικού.

Τι είναι το JSON;

Για αρχάριους προγραμματιστές, το JSON σημαίνει σημειογραφία αντικειμένου JavaScript και είναι μια μορφή κειμένου για τη μετάδοση δεδομένων σε εφαρμογές Ιστού. Αποθηκεύει πληροφορίες με τρόπο εύκολης πρόσβασης, τόσο για προγραμματιστές όσο και για υπολογιστές. Μπορεί να χρησιμοποιηθεί ως μορφή δεδομένων από οποιαδήποτε γλώσσα προγραμματισμού και γίνεται γρήγορα η προτιμώμενη σύνταξη για API, ξεπερνώντας την XML.

Τι είναι τα Tokens;

Τώρα που καταλαβαίνετε το JSON ως μορφή κειμένου δεδομένων, μπορεί να αναρωτιέστε Τι είναι τα διακριτικά; Για να το θέσω απλά, ένα διακριτικό είναι μια σειρά δεδομένων που αντιπροσωπεύει κάτι άλλο, όπως μια ταυτότητα. Στην περίπτωση του ελέγχου ταυτότητας, ένα διακριτικό που δεν βασίζεται στο JWT είναι μια σειρά χαρακτήρων που επιτρέπουν στον παραλήπτη να επικυρώσει την ταυτότητα του αποστολέα. Η σημαντική διάκριση εδώ είναι η έλλειψη νοήματος μέσα στους ίδιους τους χαρακτήρες.

Πώς λειτουργεί το JWT

Τα JWT διαφέρουν από άλλα διακριτικά ιστού στο ότι περιέχουν ένα σύνολο αξιώσεων. Οι αξιώσεις χρησιμοποιούνται για τη μετάδοση πληροφοριών μεταξύ δύο μερών. Το ποιοι είναι αυτοί οι ισχυρισμοί εξαρτάται από την περίπτωση χρήσης. Για παράδειγμα, μια αξίωση μπορεί να διευκρινίζει ποιος εξέδωσε το διακριτικό, για πόσο χρονικό διάστημα ισχύει ή ποιες άδειες έχει παραχωρηθεί στον πελάτη.

Ένα JWT είναι μια συμβολοσειρά που αποτελείται από τρία μέρη, που διαχωρίζονται με τελείες (.) και σειριοποιούνται χρησιμοποιώντας τη βάση64. Στην πιο κοινή σειριακή μορφή, τη συμπαγή σειριοποίηση, το JWT μοιάζει κάπως έτσι: xxxxx.yyyyy.zzzzz.

Μόλις αποκωδικοποιηθεί, θα λάβετε δύο συμβολοσειρές JSON:

1. Η κεφαλίδα και το ωφέλιμο φορτίο.
2. Η υπογραφή.

Η κεφαλίδα JOSE (JSON Object Signing and Encryption) περιέχει τον τύπο του διακριτικού — JWT σε αυτήν την περίπτωση — και τον αλγόριθμο υπογραφής.

Το ωφέλιμο φορτίο περιέχει τις αξιώσεις. Αυτό εμφανίζεται ως συμβολοσειρά JSON, που συνήθως δεν περιέχει περισσότερα από δώδεκα πεδία για να διατηρείται το JWT

συμπαγές. Αυτές οι πληροφορίες χρησιμοποιούνται συνήθως από τον διακομιστή για να επαληθεύσει ότι ο χρήστης έχει άδεια να εκτελέσει την ενέργεια που ζητά.

Δεν υπάρχουν υποχρεωτικές αξιώσεις για ένα JWT, αλλά τα επικαλυπτόμενα πρότυπα ενδέχεται να κάνουν τις αξιώσεις υποχρεωτικές. Για παράδειγμα, όταν χρησιμοποιείτε το JWT ως διακριτικό πρόσβασης φορέα στο OAuth2.0, πρέπει να υπάρχουν τα iss, sub, aud και exp. μερικά είναι πιο κοινά από άλλα.

Η υπογραφή διασφαλίζει ότι το διακριτικό δεν έχει τροποποιηθεί. Το μέρος που δημιουργεί το JWT υπογράφει την κεφαλίδα και το ωφέλιμο φορτίο με ένα μυστικό που είναι γνωστό τόσο στον εκδότη όσο και στον παραλήπτη ή με ένα ιδιωτικό κλειδί που είναι γνωστό μόνο στον αποστολέα. Όταν χρησιμοποιείται το διακριτικό, ο παραλήπτης επαληθεύει ότι η κεφαλίδα και το ωφέλιμο φορτίο ταιριάζουν με την υπογραφή.

Γιατί να χρησιμοποιήσετε το JWT;

Εν ολίγοις, τα JWT χρησιμοποιούνται ως ένας ασφαλής τρόπος για τον έλεγχο ταυτότητας των χρηστών και την κοινή χρήση πληροφοριών.

Συνήθως, ένα ιδιωτικό κλειδί ή μυστικό χρησιμοποιείται από τον εκδότη για την υπογραφή του JWT. Ο παραλήπτης του JWT θα επαληθεύσει την υπογραφή για να διασφαλίσει ότι το διακριτικό δεν έχει τροποποιηθεί μετά την υπογραφή του από τον εκδότη. Είναι δύσκολο για μη επαληθευμένες πηγές να μαντέψουν το κλειδί υπογραφής και να προσπαθήσουν να αλλάξουν τις αξιώσεις εντός του JWT.

Ωστόσο, δεν δημιουργούνται όλοι οι αλγόριθμοι υπογραφής ίσοι. Για παράδειγμα, ορισμένοι αλγόριθμοι υπογραφής χρησιμοποιούν μια μυστική τιμή που μοιράζεται μεταξύ του εκδότη και του μέρους που επαληθεύει το JWT. Άλλοι αλγόριθμοι χρησιμοποιούν δημόσιο και ιδιωτικό κλειδί. Το ιδιωτικό κλειδί είναι γνωστό μόνο στον εκδότη, ενώ το δημόσιο κλειδί μπορεί να διανεμηθεί ευρέως. Το δημόσιο κλειδί μπορεί να χρησιμοποιηθεί για την επαλήθευση της υπογραφής, αλλά μόνο το ιδιωτικό κλειδί μπορεί να χρησιμοποιηθεί για τη δημιουργία της υπογραφής. Αυτό είναι πιο ασφαλές από ένα κοινό μυστικό, επειδή το ιδιωτικό κλειδί χρειάζεται να υπάρχει μόνο σε ένα μέρος.

Εξαιτίας αυτού, ο διακομιστής δεν χρειάζεται να διατηρεί μια βάση δεδομένων με τις πληροφορίες που απαιτούνται για την αναγνώριση του χρήστη. Για τους προγραμματιστές, αυτά είναι εξαιρετικά νέα — ο διακομιστής που εκδίδει το JWT και ο διακομιστής που το επικυρώνει δεν χρειάζεται να είναι ο ίδιος.

2.5 ΔΗΜΙΟΥΡΓΙΑ ΠΕΡΙΒΑΛΛΟΝΤΟΣ NODE

Τώρα θα δουλέψουμε πάνω στον server μας. Ξεκινάμε το set up για το back-end. Για αυτό θα χρειαστούμε να δημιουργήσουμε μέσα στον booking φάκελο μας έναν νέο φάκελο με όνομα server.

2.5.1 ΑΡΧΕΙΑ JSON

npm init -y . Έτσι δημιουργείται το package.json αρχείο.

Τώρα που έχουμε το αρχείο package.json μπορούμε να εγκαταστήσουμε περισσότερα πακέτα που θα μας χρησιμεύσουν.

2.5.2 ΕΓΚΑΤΑΣΤΑΣΗ ΠΑΚΕΤΩΝ

Στην παρακάτω εικόνα βλέπουμε ποια εντολή θα τρέξουμε για να αποκτήσουμε όλα αυτά τα πακέτα.

```
npm i express dotenv mongoose cors morgan
```

Ας δούμε αναλυτικά τι κάνουν όλα αυτά τα πακέτα.

Dotenv:

Το Dotenv είναι μια λειτουργική μονάδα μηδενικής εξάρτησης που φορτώνει μεταβλητές περιβάλλοντος από ένα αρχείο .env στο process.env. Η αποθήκευση της διαμόρφωσης σε περιβάλλον χωριστά από τον κώδικα βασίζεται στη μεθοδολογία της εφαρμογής The Twelve-Factor App.

Mongoose:

Η μονάδα Mongoose είναι μια από τις πιο ισχυρές εξωτερικές μονάδες του NodeJS. Το Mongoose είναι ένα MongoDB ODM i.e (Μοντελοποίηση βάσης δεδομένων αντικειμένου) που χρησιμοποιείται για τη μετάφραση του κώδικα και της αναπαράστασής του από το MongoDB στον διακομιστή Node.js.

Πλεονεκτήματα της ενότητας Mongoose:

- Η επικύρωση συλλογής της βάσης δεδομένων MongoDB μπορεί να γίνει εύκολα.
- Η προκαθορισμένη δομή μπορεί να εφαρμοστεί στη συλλογή.

- Μπορούν να εφαρμοστούν περιορισμοί σε έγγραφα συλλογών που χρησιμοποιούν Mongoose.
- Η μονάδα Mongoose είναι χτισμένη στην κορυφή του προγράμματος οδήγησης MongoDB και παρέχει εύκολη αφαίρεση του ερωτήματος και ορίζει ένα ερώτημα.
- Πολλοί προγραμματιστές που συνηθίζουν να χρησιμοποιούν SQL αισθάνονται άβολα όταν εργάζονται στο MongoDB λόγω των βάσεων δεδομένων Nosql και της ευέλικτης δομής. Εδώ το Mongoose παίζει σημαντικό ρόλο και κάνει το σχήμα συλλογής παρόμοιο με τις βάσεις δεδομένων SQL.

Η ενότητα Mongoose παρέχει πολλές λειτουργίες για τον χειρισμό των εγγράφων της συλλογής της βάσης δεδομένων MongoDB.

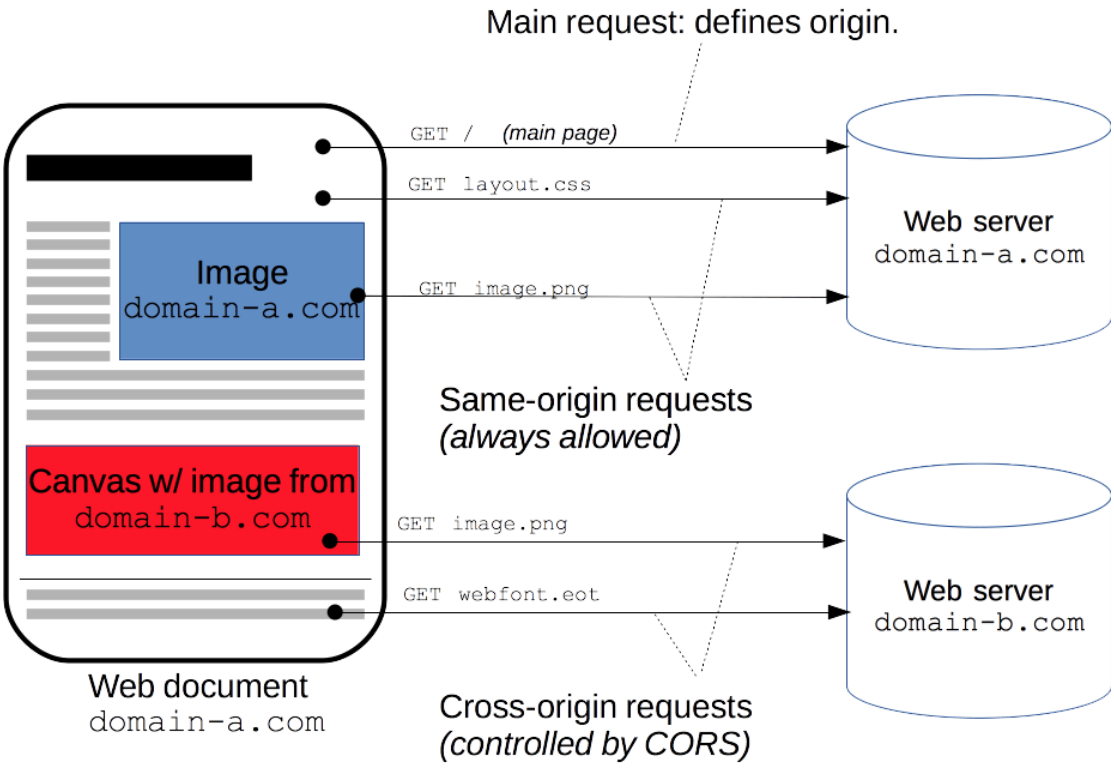
Υλοποίηση για καθορισμένη δομή και Επικύρωση συλλογής: Η ενότητα mongoose επέβαλε μια συγκεκριμένη δομή στη συλλογή και καθιστά τη συλλογή άκαμπτη.

Cors:

Το Cross-Origin Resource Sharing (CORS) είναι ένας μηχανισμός που βασίζεται σε κεφαλίδες HTTP που επιτρέπει σε έναν διακομιστή να υποδεικνύει οποιαδήποτε προέλευση (τομέα, σχήμα ή θύρα) εκτός από τη δική του από την οποία ένα πρόγραμμα περιήγησης θα πρέπει να επιτρέπει τη φόρτωση πόρων. Το CORS βασίζεται επίσης σε έναν μηχανισμό με τον οποίο τα προγράμματα περιήγησης κάνουν ένα αίτημα "προκαταρκτικής πτήσης" στον διακομιστή που φιλοξενεί τον πόρο πολλαπλής προέλευσης, προκειμένου να ελέγξει ότι ο διακομιστής θα επιτρέψει το πραγματικό αίτημα. Σε αυτήν την προκαταρκτική πτήση, το πρόγραμμα περιήγησης στέλνει κεφαλίδες που υποδεικνύουν τη μέθοδο HTTP και κεφαλίδες που θα χρησιμοποιηθούν στο πραγματικό αίτημα.

Ένα παράδειγμα αιτήματος πολλαπλής προέλευσης: ο κώδικας JavaScript διεπαφής που εξυπηρετείται από το https://domain-a.com χρησιμοποιεί το XMLHttpRequest για να υποβάλει αίτημα για https://domain-b.com/data.json.

Για λόγους ασφαλείας, τα προγράμματα περιήγησης περιορίζουν τα αιτήματα HTTP cross-origin που ξεκινούν από σενάρια. Για παράδειγμα, το XMLHttpRequest και το Fetch API ακολουθούν την ίδια πολιτική προέλευσης. Αυτό σημαίνει ότι μια εφαρμογή Ιστού που χρησιμοποιεί αυτά τα API μπορεί να ζητήσει πόρους μόνο από την ίδια προέλευση από την οποία φορτώθηκε η εφαρμογή, εκτός εάν η απάντηση από άλλες πηγές περιλαμβάνει τις σωστές κεφαλίδες CORS.



Ο μηχανισμός CORS υποστηρίζει ασφαλή αιτήματα πολλαπλής προέλευσης και μεταφορές δεδομένων μεταξύ προγραμμάτων περιήγησης και διακομιστών. Τα σύγχρονα προγράμματα περιήγησης χρησιμοποιούν CORS σε API όπως το XMLHttpRequest ή το Fetch για να μετριάσουν τους κινδύνους αιτημάτων HTTP πολλαπλής προέλευσης.

Morgan:

Το Morgan είναι ένα Middleware επιπέδου αιτήματος HTTP. Είναι ένα εξαιρετικό εργαλείο που καταγράφει τα αιτήματα μαζί με κάποιες άλλες πληροφορίες ανάλογα με τη διαμόρφωσή του και την προεπιλογή που χρησιμοποιείται. Αποδεικνύεται πολύ χρήσιμο κατά τον εντοπισμό σφαλμάτων και επίσης εάν θέλετε να δημιουργήσετε αρχεία καταγραφής.

Μαζί με τα παραπάνω θα χρειαστεί να κάνουμε εγκατάσταση και ένα ακόμη πακέτο που ονομάζεται nodemon.

Nodemon:

Όταν κάνουμε αλλαγές στον σέρβερ πρέπει συνέχεια να κάνουμε restart και να ξεκινάμε πάλι απο την αρχή. Το nodemon με ένα “npm start” κάνει αυτόματα τις αλλαγές όταν αποθηκεύουμε το πρόγραμμα μας.

Τέλος, αφού εγκαταστήσουμε όλα αυτά τα πακέτα θα δημιουργήσουμε ένα αρχείο με όνομα server.js. Αυτό θα είναι το σημείο εισόδου του διακομιστή μας, αυτό θα είναι το αρχείο που θα εκτελεστεί.

2.6 ΔΗΜΙΟΥΡΓΙΑ SERVER ΜΕ EXPRESS JS

Ανοίγουμε το αρχείο server.js που είναι μέσα στον φάκελο server. Πληκτρολογούμε την εντολή const express = require(“express”);

Δεν κάνουμε import γιατί γράφουμε σε node.js που δεν υποστηρίζει τέτοιες εντολές ακόμα.

Βάζουμε την express() μέσα σε μια μεταβλητή const app. Στη συνέχεια παίρνουμε την μεταβλητή app και κάνουμε το app.get δίνοντας του 2 arguments. Το πρώτο argument είναι το URL. Το δεύτερο είναι ένα call back function (req, res). Στην ουσία παίρνουμε το request και στέλνουμε το respond. Τέλος γράφουμε app.listen(8000, () => console.log(‘Server is running on port 8000’)); για να τρέξουμε τον κώδικα μας στο συγκεκριμένο port

Στην συνέχεια κάνουμε εγκατάσταση του πακέτου esm με την εντολή “npm i esm” για να κάνουμε imports αντί για var. Από τη στιγμή που γράφουμε σε React.js καλό είναι να μην διαφέρει ο κώδικας. Επίσης εδώ πρέπει να επισυνάψουμε ότι πρέπει να χρησιμοποιούμε τα backtricks `` για την ενσωμάτωση μεταβλητών π.χ. (`Here is your message: ${req.params.message}`);

2.6.1 ΓΙΑΤΙ EXPRESS JS

Το Express.js υποστηρίζει JavaScript, η οποία είναι μια ευρέως χρησιμοποιούμενη γλώσσα που είναι πολύ εύκολη στην εκμάθηση και υποστηρίζεται επίσης παντού. Επομένως, εάν γνωρίζετε ήδη JavaScript, τότε θα είναι πολύ εύκολο για εσάς να κάνετε προγραμματισμό χρησιμοποιώντας το Express.js.

Με τη βοήθεια του Express.js, μπορείτε εύκολα να δημιουργήσετε διαφορετικά είδη διαδικτυακών εφαρμογών σε σύντομο χρονικό διάστημα. Το Express.js παρέχει μια απλή δρομολόγηση για αιτήματα που υποβάλλονται από πελάτες. Παρέχει επίσης ένα ενδιάμεσο λογισμικό που είναι υπεύθυνο για τη λήψη αποφάσεων για να δώσει τις σωστές απαντήσεις για τα αιτήματα που υποβάλλονται από τον πελάτη.

Χωρίς Express.js, πρέπει να γράψετε τον δικό σας κώδικα για να δημιουργήσετε ένα στοιχείο δρομολόγησης που είναι μια χρονοβόρα και κουραστική εργασία. Το Express.js προσφέρει απλότητα, ευελιξία, αποτελεσματικότητα, μινιμαλισμό και επεκτασιμότητα στους προγραμματιστές. Έχει επίσης το πλεονέκτημα της ισχυρής απόδοσης καθώς είναι ένα πλαίσιο του Node.js.

Το Node.js πραγματοποιεί όλες τις εκτελέσεις πολύ γρήγορα με τη βοήθεια του Event Loop που αποφεύγει κάθε είδους αναποτελεσματικότητα. Η ισχυρή απόδοση του Node.js και η ευκολία κωδικοποίησης χρησιμοποιώντας το Express.js είναι τα πιο δημοφιλή χαρακτηριστικά που αγαπούν οι προγραμματιστές εφαρμογών ιστού. Εφόσον το Express.js είναι γραμμένο σε Javascript, μπορείτε να δημιουργήσετε ιστότοπους, εφαρμογές ιστού ή ακόμα και εφαρμογές για κινητά χρησιμοποιώντας το.

Ανάγκη της express.js

Το πιο πολύτιμο περιουσιακό στοιχείο σε κάθε επιχείρηση είναι ο χρόνος. Επιπλέον, πολλοί προγραμματιστές έχουν την πίεση να δημιουργήσουν αποτελεσματικές εφαρμογές Ιστού σε σύντομο χρονικό διάστημα. Αλλά η κωδικοποίηση εφαρμογών ιστού και η δοκιμή τους απαιτεί χρόνο. Αυτό είναι όπου το Express.js γίνεται σωτήριο για τους προγραμματιστές.

Το Express.js μπορεί να μειώσει το χρόνο κωδικοποίησης στο μισό και να μας βοηθήσει να δημιουργήσουμε αποτελεσματικές εφαρμογές web. Όχι μόνο μειώνει το χρόνο, αλλά μειώνει επίσης την προσπάθεια που απαιτείται για τη δημιουργία εφαρμογών ιστού με τη βοήθεια των διαφορετικών χαρακτηριστικών του.

Ένας άλλος λόγος για να χρησιμοποιήσετε το Express.js είναι η JavaScript. Το Express.js επιτρέπει ακόμη και σε αρχάριους να εισέλθουν στον τομέα της ανάπτυξης εφαρμογών ιστού επειδή υποστηρίζει JavaScript. Το Javascript είναι πολύ εύκολο στην εκμάθηση για οποιονδήποτε, ακόμα κι αν δεν έχει προηγούμενη γνώση οποιασδήποτε άλλης γλώσσας. Επομένως, το Express.js επιτρέπει σε νεαρά ταλέντα να εισέλθουν στον τομέα της ανάπτυξης εφαρμογών ιστού και να επιτύχουν.

Το Node.js βασίζεται σε συμβάντα και επομένως έχει τη δυνατότητα να χειρίζεται χιλιάδες αιτήματα πελατών ταυτόχρονα, κάτι που δεν είναι δυνατό με την PHP.

Στον σημερινό κόσμο, οι εφαρμογές και οι υπηρεσίες ιστού σε πραγματικό χρόνο αυξάνονται σε δημοτικότητα. Το Node.js έχει σχεδιαστεί αποκλειστικά για να υποστηρίζει πραγματικές εφαρμογές web. Το πιο συνηθισμένο παράδειγμα μιας εφαρμογής ιστού σε πραγματικό χρόνο θα ήταν η ζωντανή συνομιλία. Περιλαμβάνει χιλιάδες χρήστες και αλληλεπίδραση σε πραγματικό χρόνο που μπορεί να υποστηριχθεί εύκολα από το Node.js.

Ένα άλλο πλεονέκτημα κάθε επιχείρησης είναι τα χρήματα. Είναι σημαντικό να χρησιμοποιείτε αποτελεσματικά τα χρήματα για να μεγιστοποιήσετε τα κέρδη. Δεδομένου ότι το Express.js είναι μια εφαρμογή web ανοιχτού κώδικα και δωρεάν που παρέχει πολλές εξαιρετικές δυνατότητες, δεν υπάρχει λόγος να μην τη χρησιμοποιήσετε.

2.6.2ΧΡΗΣΗ ΤΩΝ ROUTES

Φτιάχνουμε έναν φάκελο μέσα στον server με όνομα routes και μέσα σε αυτόν τον φάκελο δημιουργούμε ένα auth.js αρχείο.

Αυτόματη φόρτωση διαδρομών

import fs from 'fs' (σημαίνει file system)

fs.readdirSync('./route').map((r) => app.use('/api', require(`./routes/${r}`))); (Με αυτόν τον κώδικα όλες οι διαδρομές (routes) θα εισαχθούν αυτόματα χωρίς να χρειαστεί να τα κάνουμε import.

Controllers

Κάνουμε νέο φάκελο μέσα στον server με όνομα controllers και φτιάχνουμε και εκεί ένα αρχείο auth.js. Στη συνέχεια προσθέτουμε τον εξής κώδικα

res.status(200).send(`Here is your message: ${req.params.message}`); };

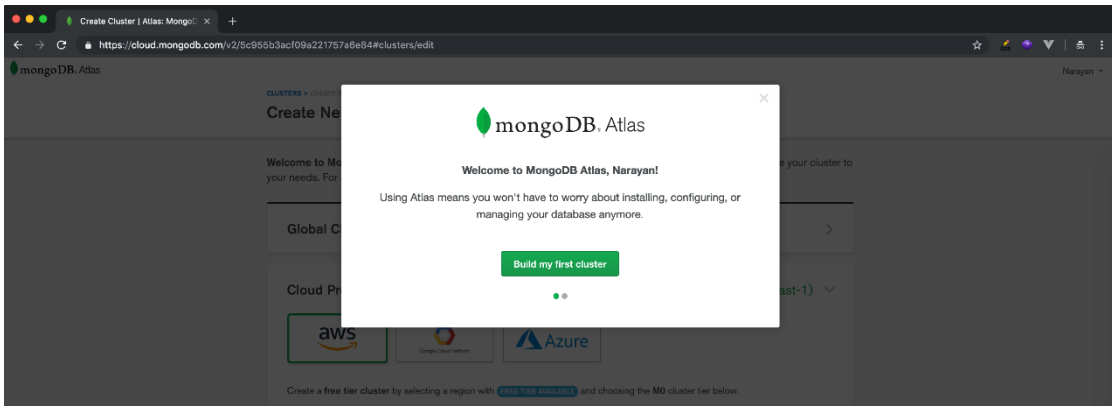
και τον ονομάζουμε showMessage

2.7 MONGODB ATLAS ΚΑΙ NOSQL ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

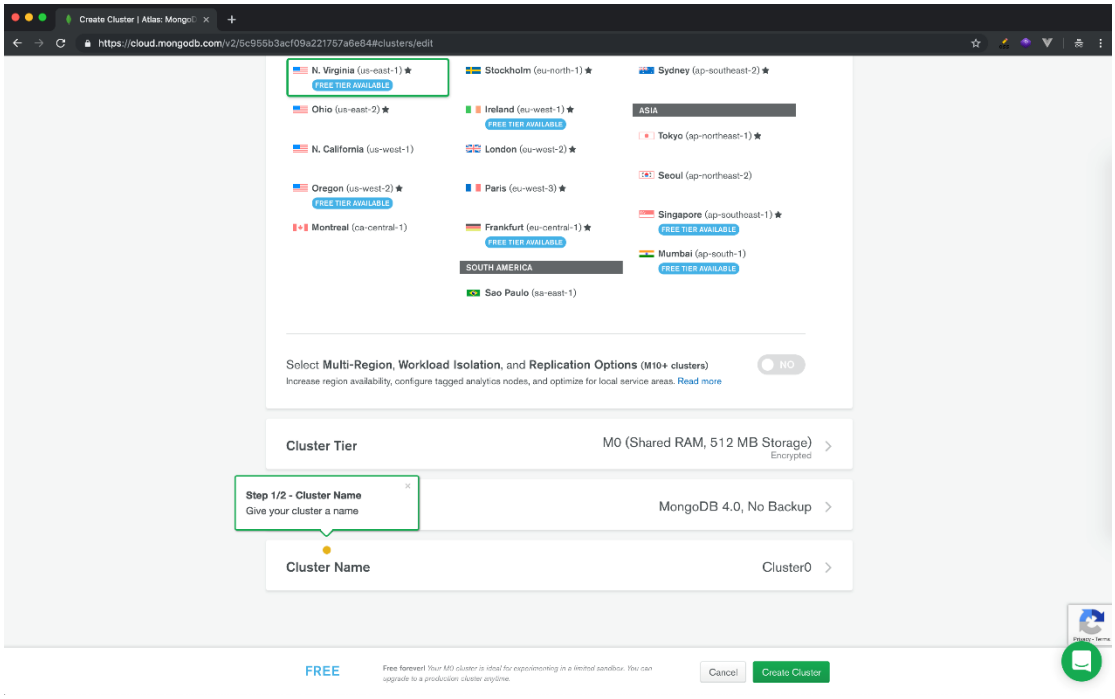
Τώρα θα συνδέσουμε το πρόγραμμα μας με την mongoDB. Πρώτα θα κάνουμε import την mongoose. Μπορούμε να χρησιμοποιήσουμε την mongoDB με 2 τρόπους.

Αν την έχουμε εγκαταστήσει τοπικά στον υπολογιστή μας είναι αρκετά εύκολο. Εάν όχι τότε μπορούμε να χρησιμοποιήσουμε μια online πλατφόρμα που θα μας επιτρέψει να τρέξουμε την βάση δεδομένων που ονομάζεται mongo atlas.

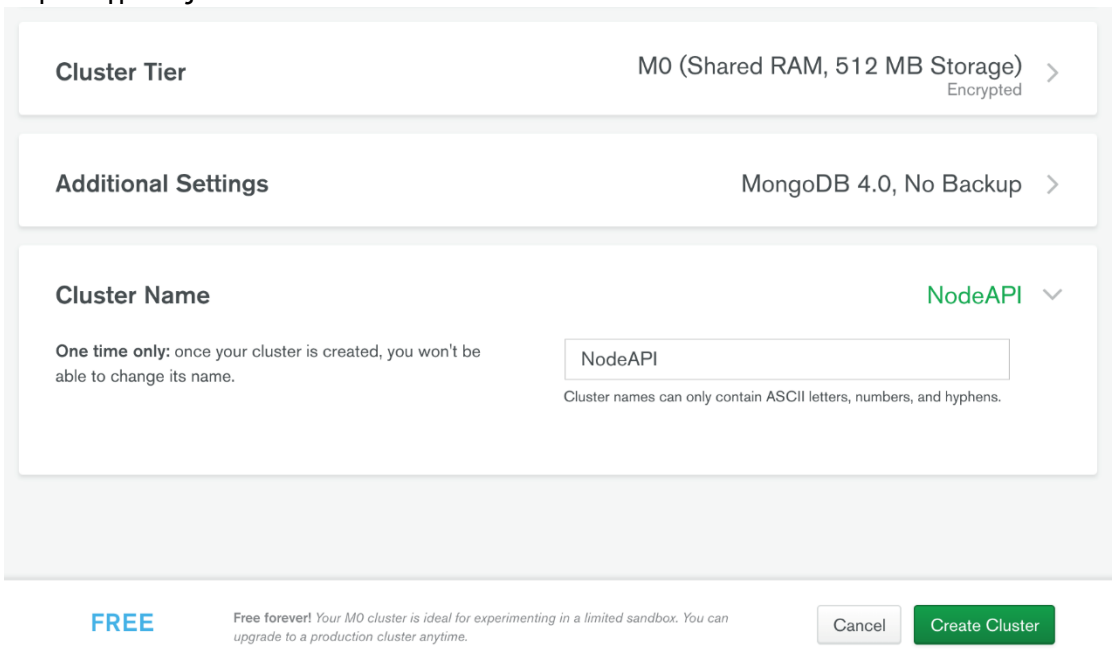
Η mongo altas επιτρέπει ένα σύμπλεγμα(clusture) για κάθε λογαριασμό. Ας ξεκινήσουμε, φτιάχνοντας έναν λογαριασμό. Στη συνέχεια θα μεταφερθείτε σε μια νέα σελίδα. Εκεί θα δείτε ένα κουμπί που λέει Build my first cluster.



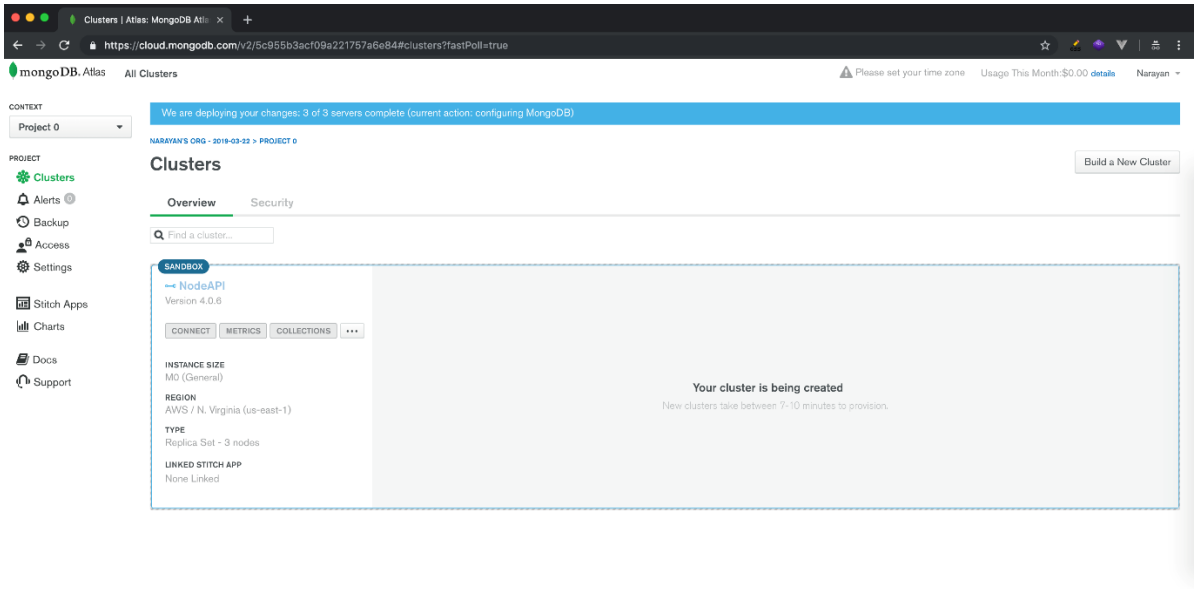
Προς το τέλος της σελίδας, θα δείτε τη σειρά Όνομα συμπλέγματος. Κάντε κλικ σε αυτό για να εισαγάγετε το όνομα για το νέο σας σύμπλεγμα/βάση δεδομένων.



Δώστε ένα όνομα για το σύμπλεγμα σας και πατήστε το κουμπί που λέει Δημιουργία συμπλέγματος.



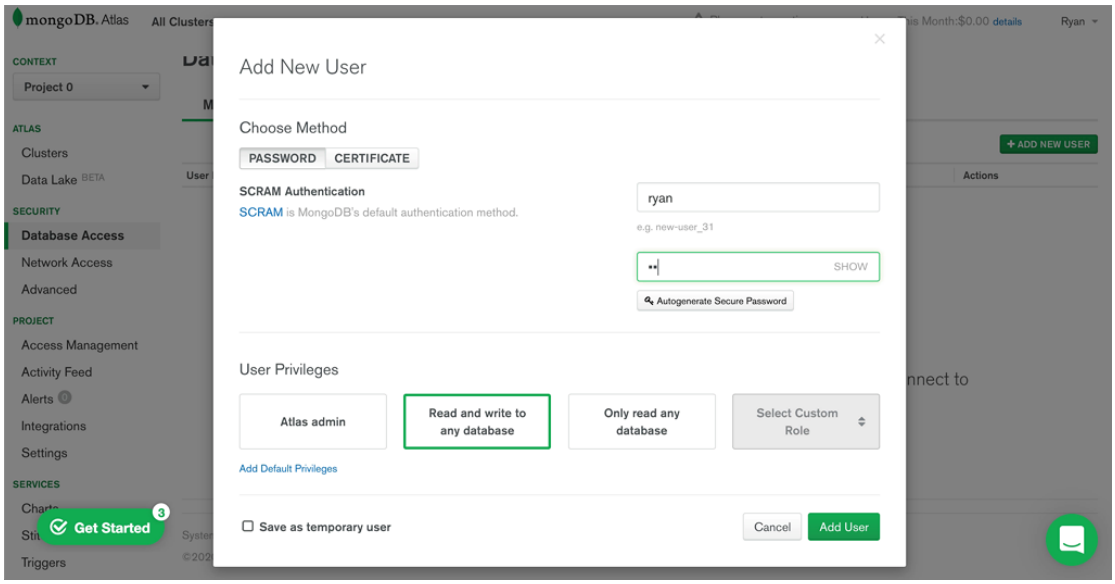
Στη συνέχεια θα δείτε την παρακάτω σελίδα



Δημιουργία χρήστη βάσης δεδομένων με όνομα και κωδικό πρόσβασης

Πρέπει να δημιουργήσετε χρήστη βάσης δεδομένων για να μπορείτε να χρησιμοποιήσετε αυτήν την υπηρεσία. Αυτός δεν είναι ο χρήστης σύνδεσης και ο κωδικός πρόσβασής σας, αλλά διαφορετικός. Θα χρησιμοποιηθεί για να επιτρέπεται η πρόσβαση στο συγκεκριμένο σύμπλεγμα mongo που μόλις δημιουργήσατε.

Κάντε κλικ στην Πρόσβαση στη βάση δεδομένων στην περιοχή Ασφάλεια στην αριστερή πλαϊνή γραμμή και στο επόμενο αναδυόμενο παράθυρο, εισαγάγετε το όνομα χρήστη και τον κωδικό πρόσβασης.

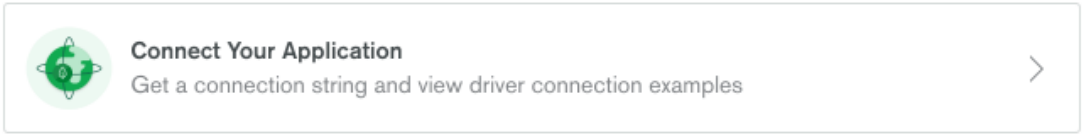


Αυτό το όνομα χρήστη και ο κωδικός πρόσβασης απαιτούνται αργότερα, επομένως πρέπει να το θυμάστε.

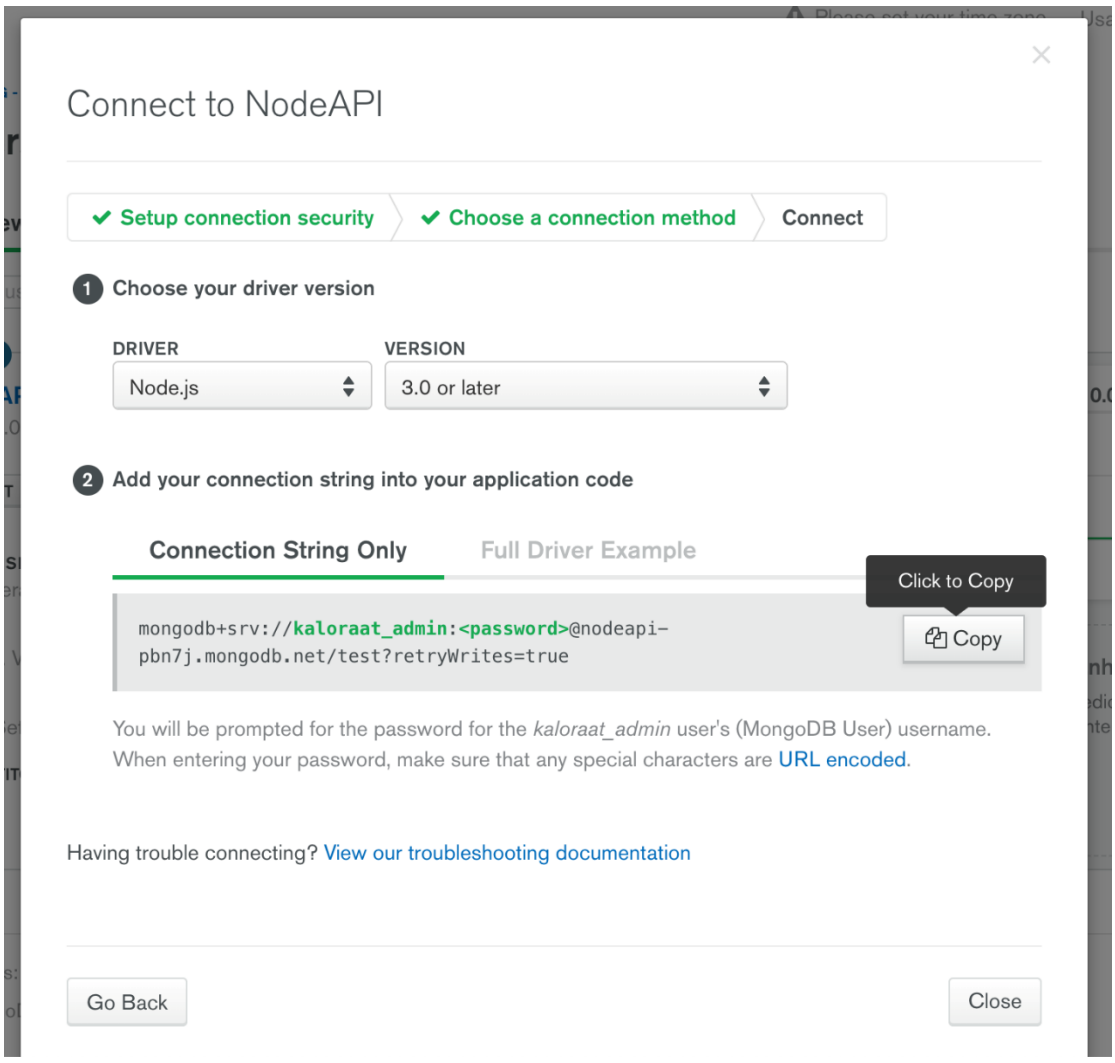
Λήψη συμβολοσειράς URI σύνδεσης

Τώρα είναι καιρός να αποκτήσουμε επιτέλους τη συμβολοσειρά URI σύνδεσης που θα επιτρέψει στην εφαρμογή μας να χρησιμοποιεί mongo atlas ως υπηρεσία βάσης δεδομένων στο cloud.

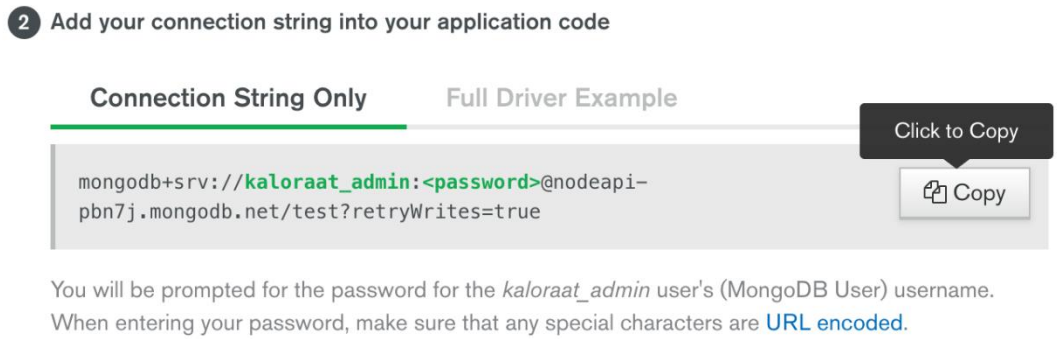
Κάντε κλικ στο Cluster στην αριστερή πλαϊνή γραμμή κάτω από τον Άτλαντα στην κορυφή. Θα δείτε ένα αναδυόμενο παράθυρο. Κάντε κλικ στην επιλογή που λέει Σύνδεση της εφαρμογής σας



Τότε θα δούμε το παρακάτω παράθυρο



Κάντε κλικ για να αντιγράψετε τη συμβολοσειρά σύνδεσης.



Θα χρειαστεί να αντικαταστήσουμε το εξάρτημα με τον πραγματικό κωδικό πρόσβασης που χρησιμοποιήσαμε νωρίτερα για να δημιουργήσουμε έναν χρήστη MongoDB.

Όταν τελειώσουμε όλα αυτά τα βήματα τότε θα ανοίξουμε τον φάκελο .env και θα καταχωρήσουμε όλα τα στοιχεία που πήραμε από την mongodb altas.

Σε αυτό το σημείο μπορεί να αναρωτηθούμε το εξής. Γιατί να χρησιμοποιήσουμε μια noSQL βάση δεδομένων και όχι μια mySQL βαση δεδομένων;

Οι βασικές διαφορές μεταξύ αυτών των δύο συστημάτων βάσεων δεδομένων είναι σημαντικές. Το να επιλέξετε ποιο να χρησιμοποιήσετε είναι πραγματικά ζήτημα προσέγγισης και όχι καθαρά τεχνικής απόφασης.

Η MySQL είναι ένα ώριμο σύστημα σχεσιακής βάσης δεδομένων, που προσφέρει ένα οικείο περιβάλλον βάσης δεδομένων για έμπειρους επαγγελματίες πληροφορικής.

Το MongoDB είναι ένα καθιερωμένο, μη σχεσιακό σύστημα βάσης δεδομένων που προσφέρει βελτιωμένη ευελιξία και οριζόντια επεκτασιμότητα, αλλά με κόστος ορισμένων χαρακτηριστικών ασφαλείας των σχεσιακών βάσεων δεδομένων, όπως η ακεραιότητα αναφοράς.

Παρακάτω, θα εξετάσουμε μερικές από τους λόγους που αποφασίζουμε μεταξύ MongoDB και MySQL.

MongoDB vs. MySQL user-friendliness

Το MongoDB είναι μια ελκυστική επιλογή για προγραμματιστές. Η φιλοσοφία της αποθήκευσης δεδομένων είναι απλή και άμεσα κατανοητή σε οποιονδήποτε έχει εμπειρία στον προγραμματισμό.

Το MongoDB αποθηκεύει δεδομένα σε συλλογές χωρίς επιβεβλημένο σχήμα. Αυτή η ευέλικτη προσέγγιση για την αποθήκευση δεδομένων την καθιστά ιδιαίτερα κατάλληλη για προγραμματιστές που μπορεί να μην είναι ειδικοί στη βάση δεδομένων, αλλά θέλουν να χρησιμοποιήσουν μια βάση δεδομένων για να υποστηρίξουν την ανάπτυξη των εφαρμογών τους.

Σε σύγκριση με τη MySQL, αυτή η ευελιξία είναι ένα σημαντικό πλεονέκτημα: για να αξιοποιήσετε στο έπακρο μια σχεσιακή βάση δεδομένων, πρέπει πρώτα να κατανοήσετε τις αρχές της κανονικοποίησης, της ακεραιότητας αναφοράς και του σχεδιασμού της σχεσιακής βάσης δεδομένων.

Με τη δυνατότητα αποθήκευσης εγγράφων διαφορετικών σχημάτων, συμπεριλαμβανομένων μη δομημένων συνόλων δεδομένων, η MongoDB παρέχει μια ευέλικτη διεπαφή προγραμματιστή για ομάδες που δημιουργούν εφαρμογές που δεν χρειάζονται όλα τα χαρακτηριστικά ασφαλείας που προσφέρουν τα σχεσιακά συστήματα. Ένα συνηθισμένο παράδειγμα μιας τέτοιας εφαρμογής είναι μια εφαρμογή Ιστού που δεν εξαρτάται από δομημένα σχήματα. μπορεί εύκολα να εξυπηρετήσει μη δομημένα, ημι-δομημένα ή δομημένα δεδομένα, όλα από την ίδια συλλογή MongoDB.

Η MySQL είναι μια κοινή επιλογή για χρήστες που έχουν μεγάλη εμπειρία στη χρήση παραδοσιακών scripting SQL, σχεδίασης λύσεων για σχεσιακές βάσεις δεδομένων ή που τροποποιούν ή ενημερώνουν υπάρχουσες εφαρμογές που ήδη λειτουργούν με ένα σχεσιακό σύστημα. Οι σχεσιακές βάσεις δεδομένων μπορεί επίσης να είναι καλύτερη επιλογή για εφαρμογές που απαιτούν πολύ περίπλοκες αλλά άκαμπτες δομές δεδομένων και σχήματα βάσεων δεδομένων σε μεγάλο αριθμό πινάκων.

Ένα συνηθισμένο παράδειγμα ενός τέτοιου συστήματος θα μπορούσε να είναι μια τραπεζική εφαρμογή που απαιτεί πολύ ισχυρή ακεραιότητα αναφοράς και εγγυήσεις συναλλαγών για να διατηρηθεί η ακριβής ακεραιότητα των δεδομένων κατά τη διάρκεια του χρόνου.

Ωστόσο, είναι σημαντικό να διευκρινιστεί ότι το MongoDB υποστηρίζει επίσης ιδιότητες ACID των συναλλαγών (ατομικότητα, συνέπεια, απομόνωση και ανθεκτικότητα). Αυτό επιτρέπει μεγαλύτερη ευελιξία στη δημιουργία ενός μοντέλου δεδομένων συναλλαγών που μπορεί να κλιμακωθεί οριζόντια σε ένα κατανεμημένο περιβάλλον και δεν έχει καμία επίδραση στην απόδοση για συναλλαγές πολλών εγγράφων.

MongoDB vs. MySQL performance

Η αξιολόγηση της απόδοσης δύο τελείως διαφορετικών συστημάτων βάσεων δεδομένων είναι πολύ δύσκολη, καθώς και τα δύο συστήματα διαχείρισης προσεγγίζουν το έργο της αποθήκευσης και ανάκτησης δεδομένων με εντελώς διαφορετικούς τρόπους. Ενώ είναι δυνατό να συγκριθούν απευθείας δύο βάσεις δεδομένων SQL με ένα σύνολο τυπικών σημείων αναφοράς SQL, η επίτευξη του ίδιου μεταξύ μη σχεσιακών και σχεσιακών βάσεων δεδομένων είναι πολύ πιο δύσκολη και υποκειμενική.

Για παράδειγμα: Η MySQL είναι βελτιστοποιημένη για συνδέσεις υψηλής απόδοσης σε πολλούς πίνακες που έχουν ευρετηριαστεί κατάλληλα. Στο MongoDB, οι συνδέσεις υποστηρίζονται με τη λειτουργία $lookup, αλλά είναι λιγότερο απαραίτητες λόγω του τρόπου με τον οποίο τείνουν να χρησιμοποιούνται τα έγγραφα MongoDB. ακολουθούν ένα ιεραρχικό μοντέλο δεδομένων και διατηρούν τα περισσότερα από τα δεδομένα σε ένα έγγραφο, εξαλείφοντας επομένως την ανάγκη για ενώσεις σε πολλά έγγραφα.

Το MongoDB είναι επίσης βελτιστοποιημένο για απόδοση εγγραφής και διαθέτει ένα συγκεκριμένο API insertMany() για γρήγορη εισαγωγή δεδομένων, δίνοντας προτεραιότητα στην ταχύτητα έναντι της ασφάλειας συναλλαγών, όπου τα δεδομένα MySQL πρέπει να εισάγονται σειρά προς σειρά.

Παρατηρώντας ορισμένες από τις συμπεριφορές ερωτημάτων υψηλού επιπέδου των δύο συστημάτων, μπορούμε να δούμε ότι η MySQL είναι ταχύτερη στην επιλογή μεγάλου αριθμού εγγραφών, ενώ η MongoDB είναι σημαντικά ταχύτερη στην εισαγωγή ή ενημέρωση μεγάλου αριθμού εγγραφών.

MongoDB vs. MySQL security

Το MongoDB αξιοποιεί το δημοφιλές μοντέλο ελέγχου πρόσβασης που βασίζεται σε ρόλους με ένα ευέλικτο σύνολο αδειών. Οι χρήστες εκχωρούνται σε έναν ρόλο και αυτός ο ρόλος τους παρέχει συγκεκριμένα δικαιώματα για σύνολα δεδομένων και λειτουργίες βάσης δεδομένων. Όλη η επικοινωνία είναι κρυπτογραφημένη με TLS και είναι δυνατή η εγγραφή κρυπτογραφημένων εγγράφων σε συλλογές δεδομένων MongoDB χρησιμοποιώντας ένα κύριο κλειδί που δεν είναι ποτέ διαθέσιμο στο MongoDB, επιτυγχάνοντας κρυπτογράφηση δεδομένων σε κατάσταση ηρεμίας.

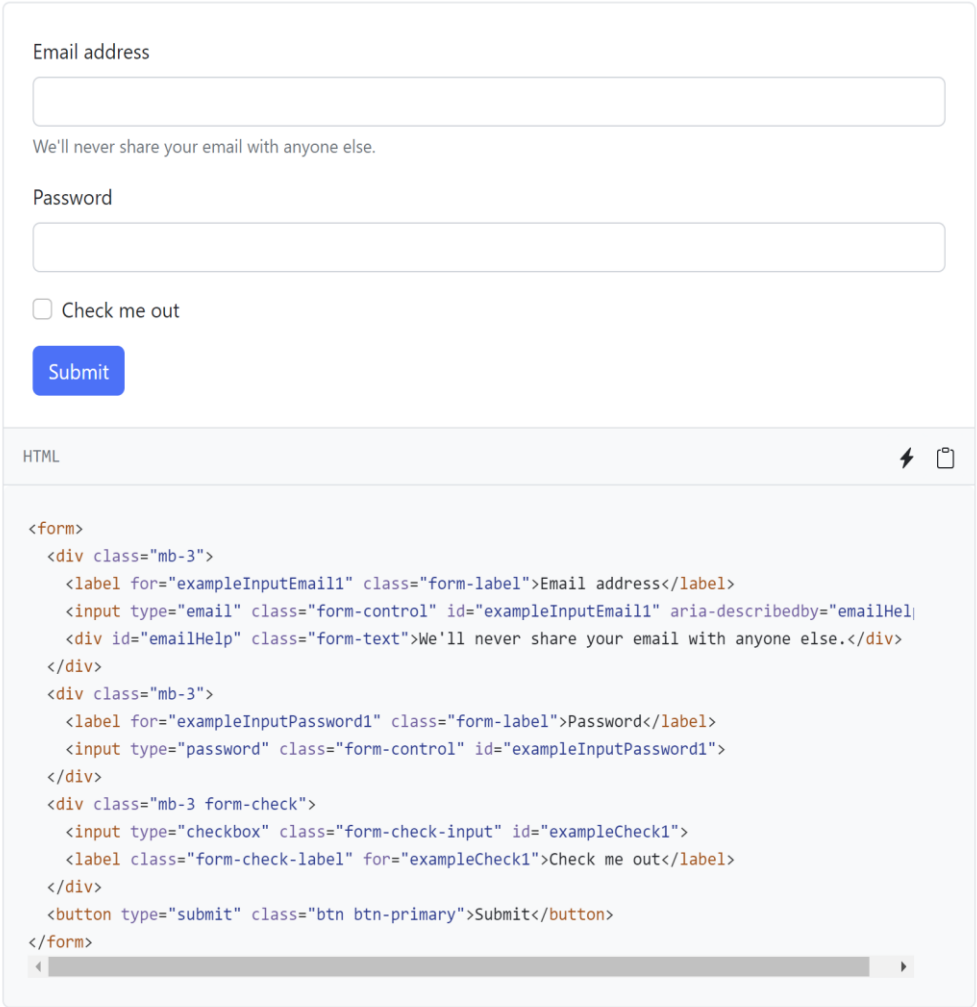
Η MySQL υποστηρίζει τις ίδιες δυνατότητες κρυπτογράφησης με το MongoDB. Το μοντέλο ελέγχου ταυτότητας είναι επίσης παρόμοιο. Στους χρήστες μπορούν να παραχωρηθούν ρόλοι αλλά και προνόμια, δίνοντάς τους δικαιώματα για συγκεκριμένες λειτουργίες βάσης δεδομένων και για συγκεκριμένα σύνολα δεδομένων.

3 ΔΗΜΙΟΥΡΓΙΑ ΕΦΑΡΜΟΓΗΣ – FRONTEND

Τώρα που τελειώσαμε με το στήσιμο της εφαρμογής μας είναι ώρα να ξεκινήσουμε να την φτιάξουμε. Θα ξεκινήσουμε με την δημιουργία της σελίδας εγγραφής του χρήστη. Σε αυτή τη σελίδα θα εμφανίσουμε μια φόρμα για να εισάγει ο χρήστης όνομα και κωδικό πρόσβασης. Αυτές τις πληροφορίες θα τις στείλουμε χρησιμοποιώντας το axios και θα τις λάβουμε στο backend μας στο req.body. Στη συνέχεια, το backend μας θα αποθηκεύσει ή θα δημιουργήσει αυτόν τον χρήστη στη βάση δεδομένων.

3.1 ΣΕΛΙΔΑ ΕΓΓΡΑΦΗΣ

Για την δημιουργία μιας φόρμας εγγραφής μας βοηθάει πολύ η bootstrap. Μπορούμε να βρούμε πολλές διαφορετικές φόρμες στο επίσημο site της bootstrap https://getbootstrap.com/ Για την φόρμα που θα φτιάξουμε θα χρησιμοποιήσουμε τον εξής κώδικα.



Προς το παρών όταν πατάμε το κουμπί “submit” δεν συμβαίνει κάτι γιατί δεν έχουμε αναθέσει κάποιο event “e”

```
1  import { useState } from "react";
2
3  const Register = () => {
4    const [name, setName] = useState("");
5    const [email, setEmail] = useState("");
6    const [password, setPassword] = useState("");
7
8    const handleSubmit = (e) => {
9      e.preventDefault();
10     console.table({ name, email, password });
11   };
12
13   const registerForm = () => (
14     <form onSubmit={handleSubmit} className="mt-3">
15       <div className="form-group mb-3">
16         <label className="form-label">Your name</label>
17         <input
18           type="text"
19           className="form-control"
20           placeholder="Enter name"
21           value={name}
22           onChange={(e) => setName(e.target.value)}
23         />
24       </div>
```

Με το e.preventDefault() κάνουμε σίγουρο ότι η σελίδα μας δεν θα ανανεωθεί όταν θα κλικάρουμε το submit κουμπί. Στην συνέχεια θα δημιουργήσουμε ένα Component για την φόρμα εγγραφής. Θα μιλήσουμε στην πορεία με περισσότερες λεπτομέρειες για τα React Components.

3.1.1 ΟΡΙΣΜΟΣ ΚΑΙ ΧΡΗΣΗ ΤΟΥ AXIOS

Τώρα αντί να κάνουμε console.log συνέχεια τις πληροφορίες του χρήστη, θα αρχίσουμε να τις στέλνουμε στο backend. Για αυτό θα χρησιμοποιήσουμε ένα σημαντικό εργαλείο που ονομάζεται axios. Το κάνουμε εγκατάσταση από την γραμμή εντολών μας με την εντολή “npm i axios”.

Ενώ ασχολούμαστε με εφαρμογές Ιστού στο React, η πιο συνηθισμένη εργασία είναι η επικοινωνία με διακομιστές υποστήριξης. Αυτό γίνεται συνήθως μέσω πρωτοκόλλου HTTP.

Υπάρχει ένας άλλος τρόπος επικοινωνίας με το backend στη React και εδώ είναι που θα μάθουμε για την εκπληκτική βιβλιοθήκη γνωστή ως Axios και μερικά από τα βασικά χαρακτηριστικά του Axios που έχουν συμβάλει στη δημοτικότητά του μεταξύ των προγραμματιστών frontend.

- Το Axios χρησιμοποιείται για την επικοινωνία με το backend και υποστηρίζει επίσης το Promise API που είναι εγγενές στο JS ES6.
- Είναι μια βιβλιοθήκη που χρησιμοποιείται για την υποβολή αιτημάτων σε ένα API, την επιστροφή δεδομένων από το API και, στη συνέχεια, την εκτέλεση εργασιών με αυτά τα δεδομένα στην εφαρμογή React.
- Το Axios είναι ένας πολύ δημοφιλής (πάνω από 95 χιλιάδες αστέρια στο Github) πελάτης HTTP, ο οποίος μας επιτρέπει να κάνουμε αιτήματα HTTP από το πρόγραμμα περιήγησης.

Γιατί χρειαζόμαστε το Axios στο React;

Το Axios μας επιτρέπει να επικοινωνούμε εύκολα με API στις εφαρμογές React. Αν και αυτό μπορεί να επιτευχθεί και με άλλες μεθόδους όπως το fetch ή το AJAX, το Axios μπορεί να προσφέρει λίγη περισσότερη λειτουργικότητα που πηγαίνει πολύ μακριά με εφαρμογές που χρησιμοποιούν React.

Το Axios είναι μια βιβλιοθήκη βασισμένη σε υποσχέσεις που χρησιμοποιείται με το Node.js και το πρόγραμμα περιήγησής σας για την πραγματοποίηση ασύγχρονων αιτημάτων JavaScript HTTP.

Όταν γνωρίζουμε ότι πρέπει να εφαρμόσουμε ορισμένα αιτήματα ασύγχρονης HTTP που βασίζονται σε υποσχέσεις στην εφαρμογή μας, με JavaScript, συνήθως σκεφτόμαστε μόνο το AJAX του jQuery ή τη μέθοδο ανάκτησης για να ολοκληρώσουμε τη δουλειά. Και ενώ το κάνουμε αυτό, στην πραγματικότητα πάμε ενάντια στη React

Έτσι, ενώ η React δεν ενδιαφέρεται καθόλου για τη χειραγώγηση του DOM, γιατί χρειαζόμαστε το jQuery για την εφαρμογή React;

Δεδομένου ότι το React χειρίζεται τα πάντα μέσα στο δικό του εικονικό DOM, ίσως δεν χρειαζόμαστε καθόλου το jQuery.

Και ως εκ τούτου, το Axios γίνεται μια ελαφρύτερη/βελτιστοποιημένη λύση για παιχνίδι με τα αιτήματά μας HTTP.

Εκτός από τα παραπάνω σημεία, γιατί προσωπικά μου αρέσει το AXIOS είναι επειδή έχει μια πολύ καθαρή και ακριβή σύνταξη.

3.1.2EVENTS

Στην ανάπτυξη Ιστού, τα συμβάντα αντιπροσωπεύουν ενέργειες που συμβαίνουν στο πρόγραμμα περιήγησης ιστού. Απαντώντας σε συμβάντα με

προγράμματα χειρισμού συμβάντων (handle events), μπορούμε να δημιουργήσουμε δυναμικές εφαρμογές JavaScript που ανταποκρίνονται σε οποιαδήποτε ενέργεια του χρήστη, όπως το κλικ με το ποντίκι, η κύλιση κατά μήκος μιας ιστοσελίδας, το άγγιγμα μιας οθόνης αφής και άλλα.

Στις εφαρμογές React, μπορούμε να χρησιμοποιήσουμε προγράμματα χειρισμού συμβάντων για να ενημερώσουμε δεδομένα κατάστασης, να ενεργοποιήσουμε αλλαγές στη βάση ή να αποτρέψουμε τις προεπιλεγμένες ενέργειες του προγράμματος περιήγησης. Για να γίνει αυτό, η React χρησιμοποιεί ένα περιτύλιγμα SyntheticEvent αντί για την εγγενή διεπαφή συμβάντος. Το SyntheticEvent προσομοιάζει στενά το τυπικό συμβάν προγράμματος περιήγησης, αλλά παρέχει πιο συνεπή συμπεριφορά για διαφορετικά προγράμματα περιήγησης ιστού. Η React παρέχει επίσης εργαλεία για την ασφαλή προσθήκη και κατάργηση μιας συσκευής ακρόασης συμβάντων παραθύρου όταν ένα στοιχείο προσαρτάται και αποσυνδέεται από το Μοντέλο αντικειμένου εγγράφου (DOM), δίνοντάς σας τον έλεγχο των συμβάντων του παραθύρου ενώ αποτρέπετε τις διαρροές μνήμης από ακροατές που αφαιρέθηκαν ακατάλληλα.

3.1.3REACT COMPONENTS

Ένα Component είναι ένα από τα βασικά δομικά στοιχεία της React. Με άλλα λόγια, μπορούμε να πούμε ότι κάθε εφαρμογή React αποτελείται από κομμάτια που ονομάζονται συστατικά. Τα εξαρτήματα κάνουν το έργο της δημιουργίας διεπαφής χρήστη πολύ πιο εύκολο. Μπορούμε να δούμε μια διεπαφή χρήστη αναλυμένη σε πολλά μεμονωμένα κομμάτια που ονομάζονται συστατικά και να τα εργαστείτε ανεξάρτητα και να τα συγχωνεύσετε όλα σε ένα μητρικό στοιχείο που θα είναι το τελικό σας περιβάλλον χρήστη.

Τα στοιχεία στο React βασικά επιστρέφουν ένα κομμάτι κώδικα JSX που λέει τι πρέπει να αποδοθεί στην οθόνη. Στη React, έχουμε κυρίως δύο τύπους στοιχείων:

Class Components

Τα στοιχεία κλάσης είναι λίγο πιο περίπλοκα από τα λειτουργικά στοιχεία. Τα λειτουργικά στοιχεία δεν γνωρίζουν τα άλλα στοιχεία του προγράμματός σας, ενώ τα στοιχεία της κλάσης μπορούν να λειτουργήσουν μεταξύ τους. Μπορούμε να περάσουμε δεδομένα από ένα στοιχείο κλάσης σε άλλα στοιχεία κλάσης. Μπορούμε να χρησιμοποιήσουμε κλάσεις JavaScript ES6 για να δημιουργήσουμε στοιχεία που βασίζονται σε κλάσεις στη React. Το παρακάτω παράδειγμα δείχνει ένα έγκυρο στοιχείο που βασίζεται σε κλάσεις στη React:

```
class Democomponent extends React.Component
{
   render(){
        return <h1>Welcome Message!</h1>;
   }
}
```

Τα στοιχεία που δημιουργήσαμε στα παραπάνω δύο παραδείγματα είναι ισοδύναμα και έχουμε επίσης δηλώσει τη βασική διαφορά μεταξύ ενός λειτουργικού στοιχείου και ενός στοιχείου κλάσης. Θα μάθουμε για περισσότερες ιδιότητες των στοιχείων που βασίζονται σε τάξη σε περαιτέρω σεμινάρια. Προς το παρόν, έχετε υπόψη σας ότι θα χρησιμοποιήσουμε λειτουργικό στοιχείο μόνο όταν είμαστε σίγουροι ότι το στοιχείο μας δεν απαιτεί αλληλεπίδραση ή εργασία με οποιοδήποτε άλλο στοιχείο. Δηλαδή, αυτά τα στοιχεία δεν απαιτούν δεδομένα από άλλα στοιχεία, ωστόσο μπορούμε να συνθέσουμε πολλαπλά λειτουργικά στοιχεία κάτω από ένα μόνο λειτουργικό στοιχείο. Μπορούμε επίσης να χρησιμοποιήσουμε στοιχεία που βασίζονται σε κλάσεις για αυτόν τον σκοπό, αλλά δεν συνιστάται, καθώς η χρήση στοιχείων που βασίζονται σε κλάση χωρίς ανάγκη θα καταστήσει την εφαρμογή σας μη αποδοτική.

Σε αυτήν την ανάρτηση, θα γράψουμε κυρίως λειτουργικά στοιχεία για να κάνουμε τα πράγματα πιο κατανοητά.

Rendering Components

Η React έχει επίσης τη δυνατότητα απόδοσης στοιχείων που ορίζονται από τον χρήστη. Για να αποδώσουμε ένα στοιχείο στο React, μπορούμε να αρχικοποιήσουμε ένα στοιχείο με ένα στοιχείο που ορίζεται από το χρήστη και να περάσουμε αυτό το στοιχείο ως πρώτη παράμετρο στη ReactDOM.render() ή να περάσουμε απευθείας το στοιχείο ως πρώτο όρισμα στη μέθοδο ReactDOM.render().

Η παρακάτω σύνταξη δείχνει πώς να αρχικοποιήσετε ένα στοιχείο σε ένα στοιχείο:

```
const elementName = <ComponentName />;
```

Στην παραπάνω σύνταξη, το ComponentName είναι το όνομα του στοιχείου που ορίζεται από το χρήστη. Σημείωση: Το όνομα ενός στοιχείου πρέπει πάντα να ξεκινά με κεφαλαίο γράμμα. Αυτό γίνεται για να διαφοροποιηθεί μια ετικέτα συστατικού με τις ετικέτες html. Το παρακάτω παράδειγμα αποδίδει ένα στοιχείο με το όνομα Hello World στην οθόνη:

```
import React from 'react';
import ReactDOM from 'react-dom';

// This is a functional component
const Welcome=()=>
{
      return <h1>Hello World!</h1>
}

ReactDOM.render(
   <Welcome />,
   document.getElementById("root")
);
```

3.1.4ΔΗΜΙΟΥΡΓΙΑ USER & ENDPOINTS

Μπορούμε τώρα να ξεκινήσουμε να φτιάξουμε έναν χρήστη(user). Αρχικά, ξεκινάμε φτιάχνοντας ένα endpoint/route και έναν controller method για να χειριστούμε την εγγραφή ενός χρήστη. Πάμε λοιπόν στον φάκελο server → routes → auth.js και δημιουργούμε εκεί ένα endpoint (τελικό σημείο) router.post('/register', register); Πρέπει να είναι post γιατί θα λάβουμε ένα post request από τον client. Δεν χρειάζεται να γράψουμε /api/register γιατί τα έχουμε ταξινομήσει στον server (//route middleware).

Για να πάρουμε πίσω τα δεδομένα και να φανούν στην βάση δεδομένων χρησιμοποιούμε το εξής κομμάτι κώδικα.

```
export const register = async (req, res) => {
                        console.log(req.body);
```

Τι είναι όμως το req.body;

Είναι ο κώδικας που μας δίνει όλες πληροφορίες για τον χρήστη.

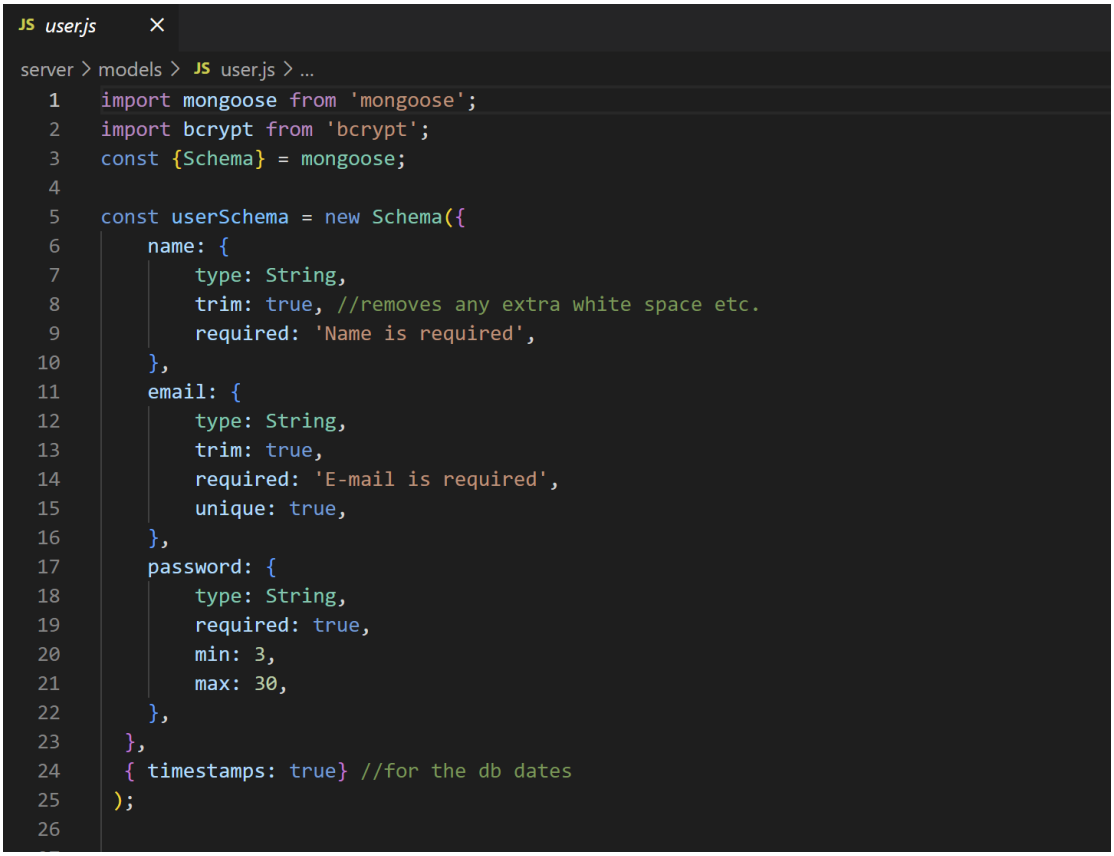
```
const res = await axios.post(`http://localhost:8000/api/register`, {name, email, password,}
```

Για να πάρουμε όμως τα json δεδομένα από το request, πρέπει να κάνουμε apply το middleware app.use(express.json());

Μπορούμε να το καταφέρουμε με ένα πακέτο npm αλλά δεν χρειάζεται να επιβαρύνομε το πρόγραμμα μας ενώ έχουμε ήδη την express.

3.1.5USER SCHEMA & PASSWORD HASH

Τώρα είναι η ώρα να αποθηκεύσουμε τα δεδομένα από το req.body στην βάση δεδομένων. Θα το υλοποιήσουμε αυτό δημιουργώντας ένα user schema(σχήμα χρήστη) που θα μας βοηθήσει να ορίσουμε τις ιδιότητες ενός χρήστη, που θα αποθηκεύονται στη βάση δεδομένων. Όλο το σχήμα θα στο κάνουμε στο αρχείο user.js που θα είναι στον καινούριο φάκελο που θα φτιάξουμε μέσα στον server και θα λέγεται models. Ένα user schema θα έχει την εξής μορφή κώδικα



Παρατηρούμε στην παραπάνω εικόνα πως αποθηκεύονται τα στοιχεία του χρήστη (όνομα, email, κωδικός πρόσβασης).

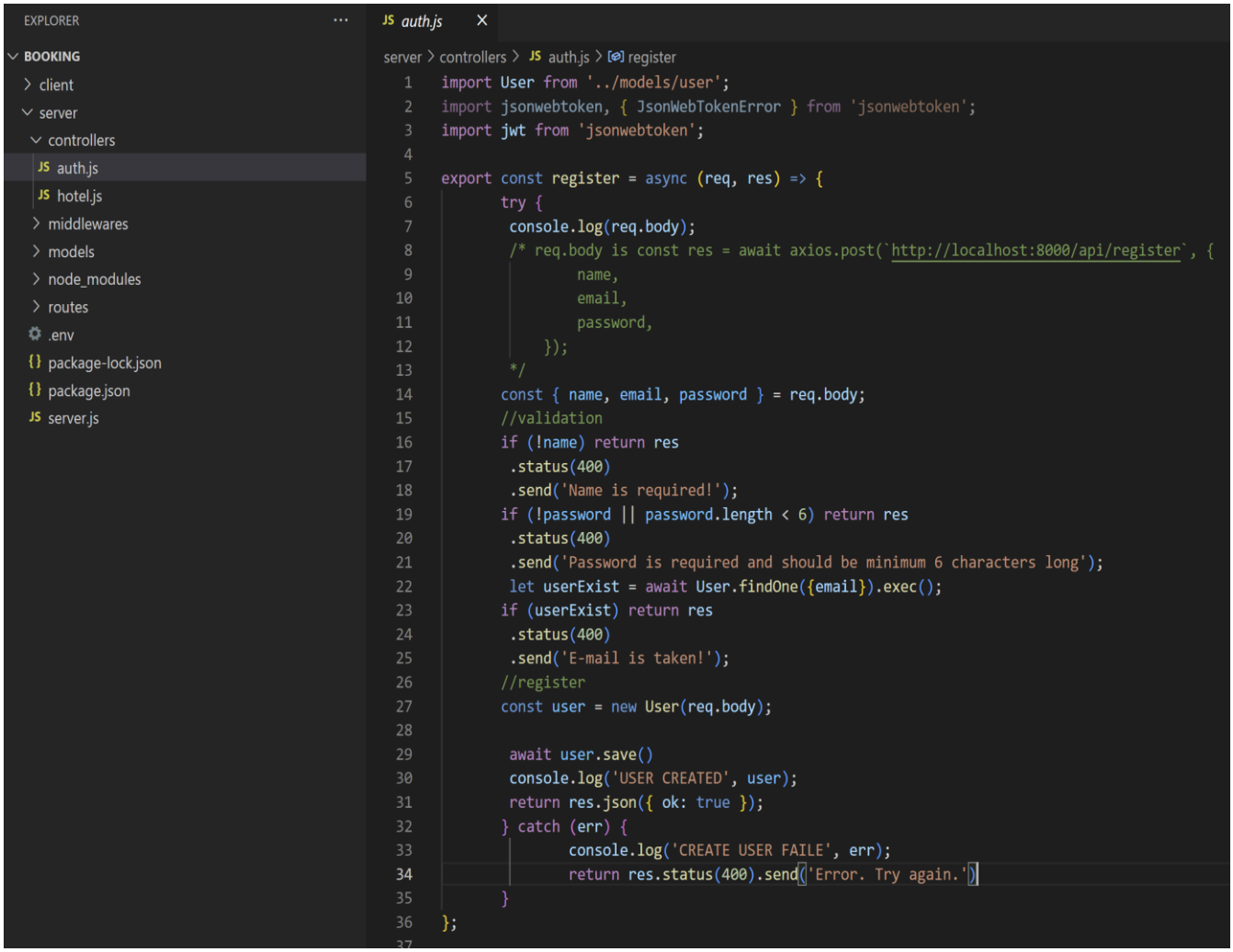
Για λόγους ασφαλείας θα πρέπει να κάνουμε τον κωδικό πρόσβασης hashed. Αυτό γίνεται με την λήψη του πακέτου bcrypt. Κάνουμε λοιπόν εγκατάσταση αυτό το πακέτο (“npm install bcrypt”). Καθώς θα αποθηκεύεται ο χρήστης στην βάση δεδομένων θα πρέπει να σιγουρευτούμε ότι ο κωδικός είναι hashed(κατακερματισμένος).

Το hashing θα πρέπει να γίνεται μόνο σε 2 περιπτώσεις

- Εάν είναι η πρώτη φορά που αποθηκεύεται ο χρήστης.
- Εάν ο χρήστης έχει ενημερώσει/τροποποιήσει τον υπάρχοντα κωδικό πρόσβασης.

Για αυτό θα πρέπει να δημιουργήσουμε ένα pre middleware στο schema μας που, θα τρέχει κάθε φορά ο χρηστής θα αποθηκεύεται/δημιουργείται, ή/και ένας κωδικός θα επεξεργάζεται/ενημερώνεται.

Στη συνέχεια θα πρέπει να αποθηκευτεί ο χρήστης στην βάση μας. Για να γίνει αυτό γράφουμε τον εξής κώδικα στο server → controllers → auth.js. Ο κώδικας μας θα δείχνει κάπως έτσι



3.1.6REACT TOASTIFY

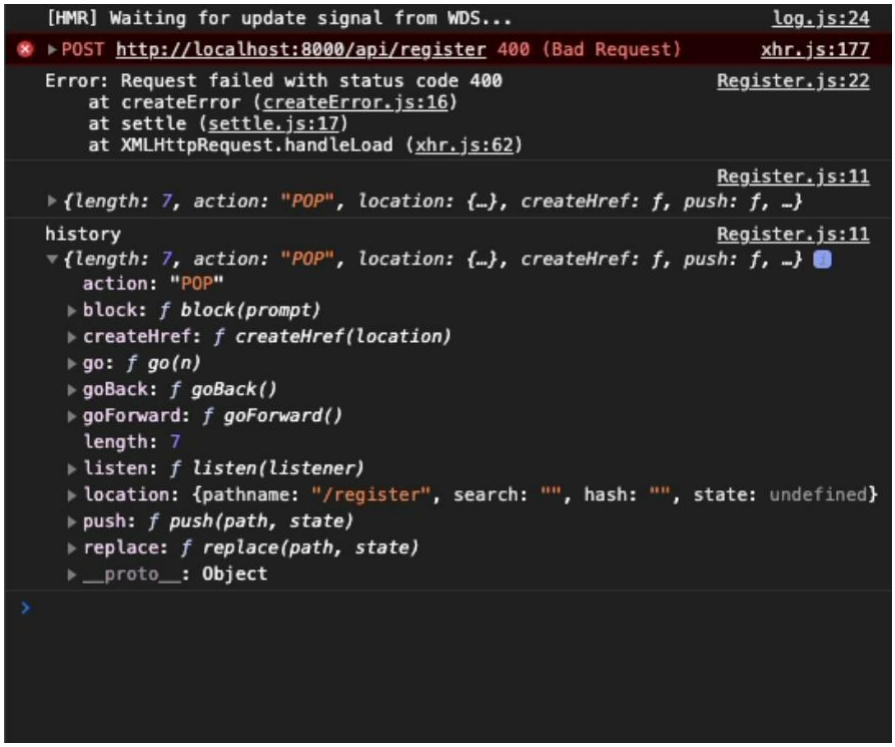
Στις περιπτώσεις που ο χρήστης έχει καταγραφεί με επιτυχία στην βάση δεδομένων, ή υπάρχει κάποιο σφάλμα με τα στοιχεία του, θέλουμε να του δείξουμε με ένα pop-up alert το αποτέλεσμα του. Για τον λόγο αυτό θα χρειαστεί να εγκαταστήσουμε μια γνωστή βιβλιοθήκη της React που ονομάζεται react toastify. Τι ακριβώς όμως είναι αυτό;

Το React-Toastify είναι μία από τις κορυφαίες βιβλιοθήκες toast React που είναι διαθέσιμες. Αυτό το εργαλείο επιτρέπει να προσθέτουμε ειδοποιήσεις toast στην εφαρμογή μας με ευκολία.

Τέλος, αφού κάνουμε εγγραφή του χρήστη με επιτυχία, πρέπει να τον κατευθύνουμε αυτόματα στην σελίδα σύνδεσης. Η react μας δίνει την δυνατότητα να κάνουμε πρόσβαση στο ιστορικό του αντικειμένου από το react-router-dom. Ολόκληρη η εφαρμογή μας είναι βασισμένη γύρω από το Component <BrowserRouter /> και αυτό μας δίνει την δυνατότητα να κάνουμε “διάρθρωση”. Στην συγκεκριμένη περίπτωση κάνουμε διάρθρωση αυτό το ιστορικό {history}. Άρα

Const Register = ({ history }) => { . Μετά αν θέλουμε κάνουμε console.log("history", history);

Και βλέπουμε ποια μεταβλητή θα χρησιμοποιήσουμε.



Σε αυτή την περίπτωση βλέπουμε ότι έχουμε αποτέλεσμα της μεταβλητής push : Άρα σβήνουμε το console.log και πάμε από κάτω από το μήνυμα success και γράφουμε history.push(‘/login’); Με αυτό τον τρόπο ανακατευθύνουμε τον χρήστη στην σελίδα εισόδου, μετά από μια επιτυχημένη εγγραφή του.

3.1.7ΜΕΤΑΒΛΗΤΕΣ ENV

Οι μεταβλητές μπορούν να παίξουν διαφορετικούς ρόλους στο React Native. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε μεταβλητές CSS στο React Native για να αποθηκεύσουμε και να εφαρμόσουμε προσαρμοσμένα στυλ.

Μια μεταβλητή περιβάλλοντος — ή όπως είναι ευρέως γνωστή, μια μεταβλητή env — είναι μια μεταβλητή της οποίας η τιμή ορίζεται εκτός προγράμματος. Χρησιμοποιείται για την αποθήκευση πληροφοριών όπως τελικά σημεία API, διαδρομές καταλόγου και ούτω καθεξής.

Εάν θέλουμε να μειωθεί η επανάληψη στον κώδικά μας, η χρήση μιας μεταβλητής env βοηθά να κωδικοποιήσουμε DRY. Το μόνο που έχουμε να κάνουμε είναι να γράψουμε ή να ορίσουμε τη μεταβλητή μας στον ριζικό σας φάκελο. Στη συνέχεια, μπορούμε να εισάγουμε τη μεταβλητή στο στοιχείο μας πριν την καλέσουμε. Ενώ οι μεταβλητές env είναι πολύ χρήσιμες στο React Native, δεν συνιστάται η χρήση τους για την αποθήκευση ευαίσθητων πληροφοριών όπως κλειδιά API, κλειδιά ελέγχου ταυτότητας ή κωδικούς πρόσβασης. Δεν είναι ασφαλή για διάφορους λόγους.Για παράδειγμα, οι μεταβλητές env δεν κρυπτογραφούν τις τιμές τους. Είναι επίσης ενσωματωμένα στο

build της εφαρμογής, πράγμα που σημαίνει ότι ο καθένας μπορεί να τις δει ελέγχοντας τα αρχεία της εφαρμογής σας.

Προς το παρών θα κάνουμε από `http://localhost:8000/api/register` σε `${process.env.REACT_APP_API}/register`

3.2 ΣΕΛΙΔΑ ΣΥΝΔΕΣΗΣ

Σε αυτό το κομμάτι θα ασχοληθούμε με τις εντολές (actios) της React. Ξεκινάμε κάνοντας νέο φάκελο μέσα στο src με όνομα actions. Μέσα σε αυτό θα δημιουργήσουμε το αρχείο με όνομα auth.js και εκεί μετακινούμε το axios.post που είχαμε στο register.js. Αφού κάνουμε import το axios, η πρώτη μεταβλητή που θα δηλώσουμε είναι το async ().

Όσο για την σελίδα σύνδεσης θα ακολουθήσουμε τα ίδια βήματα με αυτά που κάναμε στη σελίδα εγγραφής, με μόνη διαφορά ότι θα βάλουμε τα state του email και του κωδικού πρόσβασης.

Το ίδιο θα κάνουμε και με το login request.

Τώρα είναι ώρα για να συγκρίνουμε τους κωδικούς πρόσβασης από το front-end στο back-end. Αλλά πριν κάνουμε οτιδήποτε ας δούμε πώς ακριβώς λειτουργεί το login.

3.2.1ΟΡΙΣΜΟΣ ΤΟΥ LOG IN

Αρχικά πρέπει να βρούμε τον χρήστη στην βάση δεδομένων και να συγκρίνουμε τον κωδικό ο οποίος είναι ήδη κατακερματισμένος. Αυτό σημαίνει ότι πρέπει να φτιάξουμε μια μεταβλητή middleware που θα μας δίνει την δυνατότητα να συγκρίνει τον κατακερματισμένο κωδικό που είναι αποθηκευμένος στην βάση. Εάν ταιριάζουν, τότε ο χρήστης θα μπορεί να συνδεθεί με επιτυχία. Από την στιγμή που ο χρήστης θα είναι συνδεδεμένος μπορούμε να στείλουμε τον JWT(JSON Web Token).

Ο κώδικας που θα κάνει αυτή την σύγκριση θα δείχνει κάπως έτσι

auth.js

server > controllers > auth.js > register

```
37
38  export const login = async (req, res) => {
39          console.log(req.body);
40          const {email, password} = req.body;
41          try {
42            //check if user with that email exist
43            let user = await User.findOne({email}).exec();
44            // console.log('USER EXIST', user);
45            if(!user) res.status(400).send("User with that email not found");
46            // compare password
47            user.comparePassword (password, (err, match) => {
48                  console.log('COMPARE PASSORD IN LOGIN ERR', err);
49                  if(!match || err) return res.status(400).send("Wrong password")
50                  // generate a TOKEN then send as response to the client
51                  let token = jwt.sign({_id: user._id }, process.env.JWT_SECRET, {
52                          expiresIn: '7d'
53                  });
54                  res.json({token, user: {
55                          _id : user._id,
56                          name: user.name,
57                          email: user.email,
58                          createdAt: user.createdAt,
59                          updatedAt: user.updatedAt,
60                  } });
61            });
62          } catch (err) {
63                  console.log('LOGIN ERROR', err)
64                  res.status(400).send("Signin failed");
65          }
66  };
```

Για να παράγουμε τον token κατεβάζουμε ένα npm με όνομα json web token. Μετά πάμε στο .env του server και φτιάχνουμε ένα JWT_SECRET που το χρησιμοποιούμε όταν θέλουμε να επαληθεύσουμε το token αυτό.

Τώρα στο login το token και ο user στέλνονται από το backend σαν απάντηση στο frontend. Τώρα αυτό που έχουμε να κάνουμε είναι να κάνουμε save αυτό το token και να χρησιμοποιήσουμε τις πληροφορίες που μας δίνει στο response.Άρα πρέπει να το κάνουμε αποθήκευση στο REDUX state έτσι ώστε να μπορούμε να έχουμε πρόσβαση σε κάθε σελίδα αλλά και στο local storage, γιατί αν το κάνουμε αποθήκευση μόνο στο REDUX κάθε φορά που θα κάνουμε ανανέωση την σελίδα θα χάνεται.

Στη συνέχεια κάνουμε το REDUX state για κάθε ανανέωση της σελίδας. Γίνεται με τον τρόπο

```
let userState;
if(window.localStorage.getItem('auth')) {
  userState = JSON.parse(window.localStorage.getItem("auth"));
```

} else { userState = null; }

Τέλος για να μπει σε πλήρη λειτουργία η σύνδεση χρήστη πρέπει να γίνει με επιτυχία και αποσύνδεση. Αυτό το βλέπουμε με μια δομή της React με όνομα conditional rendering. Δηλαδή, κάθε φορά που είσαι σε μια σελίδα να μην δείχνει συνέχεια το navigation bar και να φαίνονται όλες οι σελίδες μας. Αυτό γίνεται με την βοήθεια της REDUX.

JS TopNav.js

client > src > components > JS TopNav.js > ...

```
import { Link } from 'react-router-dom';
import { useSelector, useDispatch } from 'react-redux';
import { useHistory } from 'react-router-dom';

const TopNav = () => {
    const dispatch = useDispatch();
    const { auth } = useSelector((state) => ({ ...state }));
    const history = useHistory()

    const logout = () => {
        dispatch({
            type: 'LOGOUT',
            payload: null,
        });
        window.localStorage.removeItem("auth");
        history.push("/login");
    };

    return (
        <div className="nav bg-light d-flex justify-content-between">
            <Link className="nav-link" to="/">
                Home
            </Link>

            {auth !== null && (
            <Link className="nav-link" to="/dashboard">
                Dashboard
            </Link>
            )}

            {auth !== null && (
                <a className="nav-link pointer" onClick={logout}>
                    Logout
                </a>
            )}

            {auth === null && (
            <>
            <Link className="nav-link" to="/about-us">
                About Us
            </Link>
            <Link className="nav-link" to="/login">
                Login
            </Link>
            <Link className="nav-link" to="/register">
                Register
            </Link>
            </>
            )}
        </div>
    );

}

export default TopNav;
```

3.2.2PRIVATE ROUTING

Σε αυτό το σημείο του προγράμματος μας θα δημιουργήσουμε προστατευόμενες σελίδες που είναι προσβάσιμες για συνδεδεμένους χρήστες. Οι ιδιωτικές διαδρομές στο δρομολογητή React (ονομάζονται επίσης προστατευμένες διαδρομές) απαιτούν από έναν χρήστη να έχει εξουσιοδότηση να επισκεφθεί μια διαδρομή (ανάγνωση: σελίδα). Έτσι, εάν ένας χρήστης δεν είναι εξουσιοδοτημένος για μια συγκεκριμένη σελίδα, δεν μπορεί να έχει πρόσβαση σε αυτήν. Το πιο συνηθισμένο παράδειγμα είναι ο έλεγχος ταυτότητας σε μια εφαρμογή React όπου ένας χρήστης μπορεί να έχει πρόσβαση στις προστατευμένες σελίδες μόνο όταν είναι εξουσιοδοτημένες (πράγμα που σημαίνει ότι σε αυτήν την περίπτωση έχει γίνει έλεγχος ταυτότητας). Ωστόσο, η εξουσιοδότηση υπερβαίνει τον έλεγχο ταυτότητας. Για παράδειγμα, ένας χρήστης μπορεί επίσης να έχει ρόλους και δικαιώματα που δίνουν σε έναν χρήστη πρόσβαση σε συγκεκριμένες περιοχές της εφαρμογής.

Στον φάκελο components φτιάχνουμε ένα αρχείο με όνομα PrivateRoute.js. Μπορούμε επίσης να δείχνουμε το ID του χρήστη και πότε φτιάχτηκε ο λογαριασμός.

3.3 ΔΗΜΙΟΥΡΓΕΙΑ ΞΕΝΟΔΟΧΕΙΩΝ

Τώρα θα ξεκινήσουμε να δημιουργούμε ξενοδοχεία τα οποία θα καταχωρούνται στη βάση δεδομένων. Επειδή θέλουμε να φτιάξουμε ένα interface παρόμοιο του booking.com θα πάρουμε την API algolia. Πριν όμως ξεκινήσουμε, ας εντρυφήσουμε σε ένα θέμα που ακόμα και κάποιοι προγραμματιστές δυσκολεύονται να κατανοήσουν. Τι ακριβώς είναι ένα API και γιατί είναι τόσο σημαντική;

3.3.1APIs

Το API είναι το ακρωνύμιο του Application Programming Interface, το οποίο είναι ένας ενδιάμεσος λογισμικού που επιτρέπει σε δύο εφαρμογές να συνομιλούν μεταξύ τους. Κάθε φορά που χρησιμοποιείτε μια εφαρμογή όπως το Facebook, στέλνετε ένα άμεσο μήνυμα ή ελέγχετε τον καιρό στο τηλέφωνό σας, χρησιμοποιείτε ένα API.

Ένα παράδειγμα API;

Όταν χρησιμοποιείτε μια εφαρμογή στο κινητό σας τηλέφωνο, η εφαρμογή συνδέεται στο Διαδίκτυο και στέλνει δεδομένα σε έναν διακομιστή. Στη συνέχεια, ο διακομιστής ανακτά αυτά τα δεδομένα, τα ερμηνεύει, εκτελεί τις απαραίτητες ενέργειες και τα στέλνει πίσω στο τηλέφωνό σας. Στη συνέχεια, η εφαρμογή ερμηνεύει αυτά τα δεδομένα και σας παρουσιάζει τις πληροφορίες που θέλετε με ευανάγνωστο τρόπο. Αυτό είναι ένα API - όλα αυτά συμβαίνουν μέσω API.

Για να το εξηγήσουμε καλύτερα, ας πάρουμε ένα γνωστό παράδειγμα.

Φανταστείτε ότι κάθεστε σε ένα τραπέζι σε ένα εστιατόριο με ένα μενού με επιλογές για να παραγγείλετε. Η κουζίνα είναι το μέρος του «συστήματος» που θα προετοιμάσει την παραγγελία σας. Αυτό που λείπει είναι ο κρίσιμος σύνδεσμος για να επικοινωνήσετε την παραγγελία σας στην κουζίνα και να παραδώσετε το φαγητό σας πίσω στο τραπέζι σας. Εκεί μπαίνει ο σερβιτόρος ή το API. Ο σερβιτόρος είναι ο αγγελιοφόρος - ή API - που δέχεται το αίτημα ή την παραγγελία σας και λέει στην κουζίνα - στο σύστημα - τι να κάνει. Στη συνέχεια, ο σερβιτόρος σας παραδίδει την απάντηση. σε αυτή την περίπτωση, είναι το φαγητό.

Ακολουθεί ένα πραγματικό παράδειγμα API. Μπορεί να είστε εξοικειωμένοι με τη διαδικασία αναζήτησης πτήσεων στο διαδίκτυο. Ακριβώς όπως το εστιατόριο, έχετε μια ποικιλία επιλογών για να διαλέξετε, όπως διαφορετικές πόλεις, ημερομηνίες αναχώρησης και επιστροφής και πολλά άλλα. Ας φανταστούμε ότι κάνετε κράτηση για πτήση σε έναν ιστότοπο αεροπορικής εταιρείας. Επιλέγετε πόλη και ημερομηνία αναχώρησης, πόλη και ημερομηνία επιστροφής, κατηγορία καμπίνας, καθώς και άλλες μεταβλητές. Για να κλείσετε την πτήση σας, αλληλεπιδράτε με τον ιστότοπο της αεροπορικής εταιρείας για να αποκτήσετε πρόσβαση στη βάση δεδομένων της και να δείτε εάν υπάρχουν διαθέσιμες θέσεις σε αυτές τις ημερομηνίες και ποιο μπορεί να είναι το κόστος.

Ωστόσο, τι γίνεται αν δεν χρησιμοποιείτε τον ιστότοπο της αεροπορικής εταιρείας – ένα κανάλι που έχει άμεση πρόσβαση στις πληροφορίες; Τι γίνεται αν χρησιμοποιείτε μια διαδικτυακή ταξιδιωτική υπηρεσία, όπως το Kayak ή το Expedia, που συγκεντρώνει πληροφορίες από διάφορες βάσεις δεδομένων αεροπορικών εταιρειών;

Η ταξιδιωτική υπηρεσία, σε αυτήν την περίπτωση, αλληλεπιδρά με το API της αεροπορικής εταιρείας. Το API είναι η διεπαφή στην οποία, όπως και ο χρήσιμος σερβιτόρος σας, μπορεί να ζητηθεί από αυτήν την ηλεκτρονική ταξιδιωτική υπηρεσία να λάβει πληροφορίες από τη βάση δεδομένων της αεροπορικής εταιρείας για κράτηση θέσεων, επιλογές αποσκευών κ.λπ. Στη συνέχεια, το API λαμβάνει την απάντηση της αεροπορικής εταιρείας στο αίτημά σας και την παραδίδει σωστά πίσω στην ηλεκτρονική ταξιδιωτική υπηρεσία, η οποία σας εμφανίζει στη συνέχεια τις πιο ενημερωμένες, σχετικές πληροφορίες.

Αυτό που παρέχει επίσης ένα API είναι ένα επίπεδο ασφάλειας

Τα δεδομένα του τηλεφώνου σας δεν εκτίθενται ποτέ πλήρως στον διακομιστή και, ομοίως, ο διακομιστής δεν εκτίθεται ποτέ πλήρως στο τηλέφωνό σας. Αντίθετα, το καθένα επικοινωνεί με μικρά πακέτα δεδομένων, μοιράζοντας μόνο ό,τι είναι απαραίτητο, όπως η παραγγελία σε πακέτο. Λέτε στο εστιατόριο τι θα θέλατε να φάτε, σας λένε τι χρειάζονται σε αντάλλαγμα και, στο τέλος, παίρνετε το γεύμα σας.

Τα API έχουν γίνει τόσο πολύτιμα που αποτελούν μεγάλο μέρος των εσόδων πολλών επιχειρήσεων. Μεγάλες εταιρείες όπως η Google, το eBay, η Salesforce.com, η Amazon και η Expedia είναι μερικές μόνο από τις εταιρείες που κερδίζουν χρήματα από τα API τους. Αυτό στο οποίο αναφέρεται η «οικονομία API» είναι αυτή η αγορά των API.

Το σύγχρονο API

Με τα χρόνια, το τι είναι ένα "API" έχει συχνά περιγράψει οποιοδήποτε είδος γενικής διεπαφής συνδεσιμότητας σε μια εφαρμογή. Πιο πρόσφατα, ωστόσο, το σύγχρονο API έχει λάβει ορισμένα χαρακτηριστικά που τα καθιστούν εξαιρετικά πολύτιμα και χρήσιμα:

Τα σύγχρονα API συμμορφώνονται με πρότυπα (συνήθως HTTP και REST), τα οποία είναι φιλικά προς προγραμματιστές, εύκολα προσβάσιμα και κατανοητά ευρέως

Αντιμετωπίζονται περισσότερο σαν προϊόντα παρά με κωδικό. Είναι σχεδιασμένα για κατανάλωση για συγκεκριμένο κοινό (π.χ. προγραμματιστές κινητών), είναι τεκμηριωμένα και έχουν εκδοθεί με τρόπο ώστε οι χρήστες να μπορούν να έχουν συγκεκριμένες προσδοκίες για τη συντήρηση και τον κύκλο ζωής τους.

Επειδή είναι πολύ πιο τυποποιημένα, έχουν πολύ ισχυρότερη πειθαρχία για την ασφάλεια και τη διακυβέρνηση, καθώς και παρακολουθούνται και διαχειρίζονται για απόδοση και κλίμακα

Όπως κάθε άλλο κομμάτι του παραγόμενου λογισμικού, το σύγχρονο API έχει τον δικό του κύκλο ζωής ανάπτυξης λογισμικού (SDLC) σχεδιασμού, δοκιμής, κατασκευής, διαχείρισης και έκδοσης. Επίσης, τα σύγχρονα API είναι καλά τεκμηριωμένα για κατανάλωση και έκδοση εκδόσεων.

3.3.2HOTEL FORMS

Τώρα που είδαμε τι σημαίνει API, ας ξεκινήσουμε να την δουλεύουμε για το πρόγραμμα μας.

New Hotel Form 1

Σε αυτό το μέρος θα δώσουμε το δικαίωμα στον χρήστη να ανεβάζει το δικό του δωμάτιο ξενοδοχείου. Πριν ξεκινήσουμε όμως χρειαζόμαστε την API της algolia. Πληκτρολογούμε στο terminal “npm i algolia-places-react”.

New Hotel Form 2

Σε αυτή την φόρμα ο χρήστης θα μπορεί να δηλώνει τον τίτλο δωματίου, χαρακτηριστικά, εικόνα, τιμή και χωρητικότητα.

Ας χρησιμοποιήσουμε τώρα ένα event handler που θα μας δίνει το δικαίωμα να αλλάζουμε το state και να δούμε πώς θα ανεβάζει ο χρήστης την φωτογραφία για το δωμάτιο του.

Στο αρχείο NewHotel.js μέσα στον φάκελο hotels γράφουμε

```
JS NewHotel.js ×

client > src > hotels > JS NewHotel.js > ...

import { useState } from 'react';
import { toast } from 'react-toastify';
import {DatePicker, Select } from 'antd';
import { createHotel } from '../actions/hotel';
import {useSelector} from 'react-redux';
import HotelCreateForm from '../components/forms/HotelCreateForm';

const { Option } = Select;

const NewHotel = () => {
    //redux
    const {auth} = useSelector((state) => ({ ...state }));
    const {token} = auth;
    // state
    const [values, setValues] = useState({
        title: "",
        content: "",
        image: "",
        location: "",
        price: "",
        from: "",
        to: "",
        bed: ""
    });
    const [preview, setPreview] = useState('https://via.placeholder.com/100x100.png?text=PREVIEW');
    // destructuring variebles from state
    const { title, content, image, location, price, from, to, bed } = values;

    //here we have to send all the data as a "form" data to our backend, including the image
    const handleSubmit = async (e) => {
        e.preventDefault();
        //console.log(values);
        //console.log(location);

        let hotelData = new FormData()
        hotelData.append('title', title)
        hotelData.append('content', content)
        hotelData.append('location', location)
        hotelData.append('price', price)
        image && hotelData.append('image', image)
        hotelData.append('from', from)
        hotelData.append('to', to)
        hotelData.append('bed', bed)

        try {
            let res = await createHotel(token, hotelData)
        console.log('HOTEL CREATE RES', res)
        toast.success('New hotel posted!')
        setTimeout(() => {
            window.location.reload();
        }, 1000);
        }catch (err) {
            console.log(err)
            toast.error(err.response.data);
        }
```

```
    const handleImageChange = (e) => {
        setPreview(URL.createObjectURL(e.target.files[0]));
        setValues({...values, image: e.target.files[0] });
    };

    const handleChange = (e) => {
        setValues({...values, [e.target.name]: e.target.value });
    };

    return (
        <>
            <div className="container-fluid bg-secondary p-5 text-center">
                <h2>Add Hotel</h2>
            </div>
            <div className="container-fluid">
                <div className="row">
                    <div className="col-md-10">
                        <br />
                        <HotelCreateForm
                            values={values}
                            setValues={setValues}
                            handleImageChange={handleImageChange}
                            handleSubmit={handleSubmit}
                            handleChange={handleChange}
                            location={location}
                        />
                    </div>
                    <div className="col-md-2">
                        <img src={preview} alt="preview_image" className="img img-fluid m-2" />
                    </div>
                </div>
            </div>
        </>
    );
};

export default NewHotel;
```

Μετά από αυτό θα δουλέψουμε πάνω στις ημερομηνίες from – to. Κάνουμε import το {DatePicker} από την βιβλιοθήκη antd. Με αυτό, κάθε φορά που ο χρήστης θα ορίζει μια ημερομηνία θα του εμφανίζεται ένα ημερολόγιο. Πρέπει να είμαστε προσεκτικοί διότι υπάρχει η δυνατότητα λάθους του χρήστη, δηλαδή να ορίσει μια παλιά ημερομηνία. Γι' αυτό θα χρειαστεί να γράψουμε μια javascript μεταβλητή που θα παίρνει από την σημερινή ημερομηνία και έπειτα και θα δείχνει μόνο αυτές. Τις παλιές ημερομηνίες θα τις εμφανίζει, απλά ως μη εφικτές να οριστούν.

Τώρα είμαστε έτοιμοι να στείλουμε δεδομένα για να δημιουργήσουμε/ανεβάσουμε ένα δωμάτιο ξενοδοχείου.

Ας δούμε τι έχουμε να κάνουμε στο backend

Δημιουργία ενός hotel schema

Όπως και με τον χρήστη έτσι και με το ξενοδοχείο θα χρειαστούμε ένα schema που θα μας δίνει την δυνατότητα να αποφασίζουμε ποια πεδία θα αποθηκεύονται στη βάση.

Στα models του server θα κάνουμε ένα αρχείο με όνομα hotel.js και θα γράψουμε μέσα :

JS hotel.js

server > models > JS hotel.js > ...

```
import mongoose from 'mongoose';

const {Schema} = mongoose
const {ObjectId} = mongoose.Schema

const hotelSchema = new Schema({
    title: {
        type: String,
        required: 'Title is required',
    },
    content: {
        type: String,
        required: 'Content is required',
        maxlength: 10000,
    },
    location: {
        type: String,
        required: 'Location is required',
    },
    price: {
        type: Number,
        required: 'Pleace provide a price',
        trim: true,
    },
    postedBy: {
        type: ObjectId,
        ref: 'User',
    },
    image: {
        data: Buffer,
        contentType: String,
    },
    from: {
        type: Date,
    },
    to: {
        type: Date,
    },
    bed: {
        type: Number,
    },
}, {timestamps: true}
);

export default mongoose.model("Hotel", hotelSchema);
```

Μετά θα χρειαστεί να φτιάξουμε ένα route. Κάνουμε λοιπόν “npm i express-formidable. Αυτό μας βοηθάει να χειριστούμε τα δεδομένα της φόρμας. Το χρειαζόμαστε γιατί στέλνουμε δεδομένα από την πλευρά του client. Εάν δεν έχουμε το ενδιάμεσο λογισμικό δεν θα πάρουμε πίσω κάποιο από τα δεδομένα μας. Όταν γίνει η εγκατάσταση το προσθέτουμε στο middleware. Τότε θα μπορούμε να έχουμε τις εξής ιδιότητες

req.fields; // εδώ λαμβάνονται όλες τις πληροφορίες όπως τίτλος, περιεχόμενο κ.λπ.

req.files; // εδώ λαμβάνεται η εικόνα.

Δημιουργία ενός controller

Αντί να ορίσουμε όλη τη λογική διαχείρισης αιτημάτων σας clousure στα αρχεία διαδρομής, θέλουμε να οργανώσουμε αυτήν τη συμπεριφορά χρησιμοποιώντας έναν controller.

Αποστολή μηνύματος επιτυχίας μετά την αποθήκευση ξενοδοχείου

Μετά από κάθε επιτυχημένη καταγραφή κάθε ξενοδοχείου θέλουμε να στέλνετε μήνυμα επιτυχίας. Αλλιώς, εάν κάτι δεν έχει γίνει σωστά, π.χ δεν καταχωρηθεί τίτλος, θα στέλνετε έναν error message στο frontend.

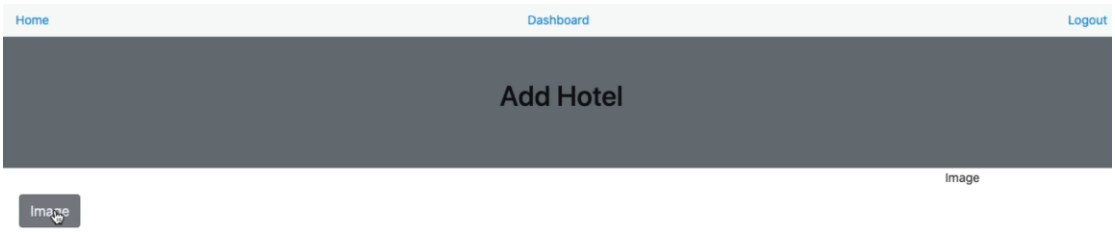
3.3.3ΕΜΦΑΝΙΣΗ ΞΕΝΟΔΟΧΕΙΩΝ

Το επόμενό μας βήμα είναι να δείξουμε όλα τα ξενοδοχεία στην αρχική μας σελίδα. Για να το κάνουμε αυτό πρέπει να πάμε στο backend και να φτιάξουμε ένα endpoint και έναν controller. Ένα API Endpoint είναι σαν μια γέφυρα μεταξύ του frontend και του backend μιας εφαρμογής web. Όλες οι σχετικές πληροφορίες διαβιβάζονται μεταξύ των συστημάτων χρησιμοποιώντας αυτήν τη μέθοδο. Το API σημαίνει διεπαφή προγράμματος εφαρμογής, όπως υποδηλώνει το όνομα είναι μια διεπαφή για την εφαρμογή Ιστού.

Παράδειγμα ενός endpoint

```
[
  {
    "image": {
      "contentType": "image/jpeg"
    },
    "_id": "600174e7c970c607284f4849",
    "title": "Bondi hotel",
    "content": "Bondi hotelBondi hotelBondi hotelBondi hotelBondi hotelBondi hotel",
    "location": "Bondi Junction , New South Wales, Australia",
    "price": 200,
    "from": "2021-01-15T00:00:00.000Z",
    "to": "2021-01-16T00:00:00.000Z",
    "bed": 2,
    "createdAt": "2021-01-15T10:56:39.792Z",
    "updatedAt": "2021-01-15T10:56:39.792Z",
    "__v": 0
  },
```


Αυτές τις πληροφορίες θα τις χρησιμοποιήσουμε στο πρόγραμμα μας με το axios για να έχουμε στην αρχική μας όλα τα ξενοδοχεία. Στη συνέχεια θα φτιάξουμε το styling και κάποια επιπρόσθετα components για να φαίνονται σωστά ταξινομημένα.



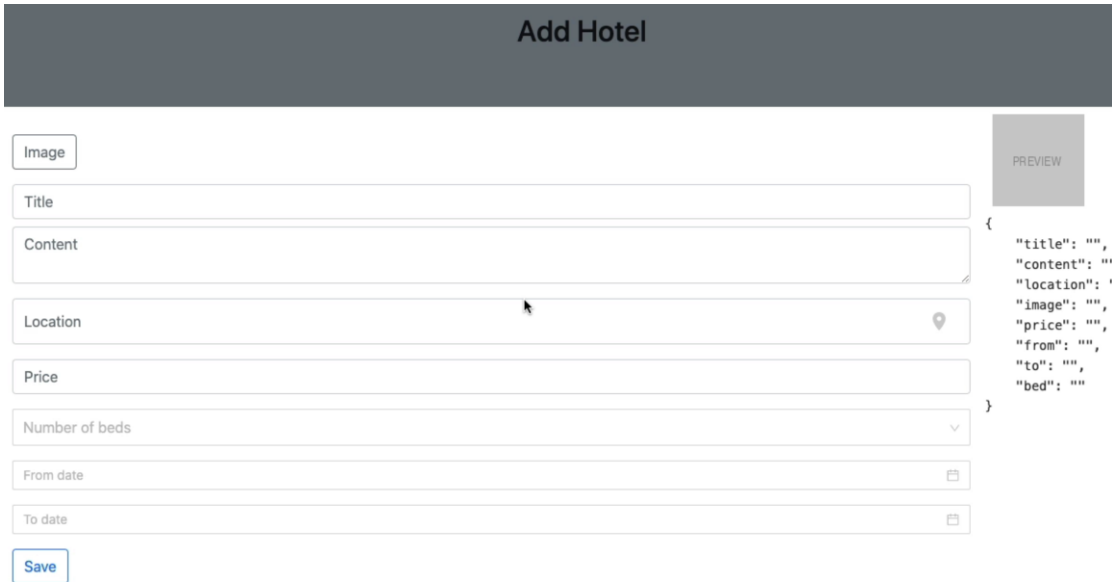
3.3.4ΕΜΦΑΝΙΣΗ / ΕΠΕΞΕΡΓΑΣΙΑ / ΔΙΑΓΡΑΦΗ ΞΕΝΟΔΟΧΕΙΩΝ

Τέλος θα χρειαστεί να δώσουμε το δικαίωμα στον admin της ανάρτησης να διαχειρίζεται όπως αυτός επιθυμεί το ξενοδοχείο του. Για να το κάνουμε αυτό θα χρησιμοποιήσουμε μια λογική ευρέως γνωστή στον προγραμματισμό με όνομα **CRUD.** Ας δούμε τι σημαίνει αυτό.

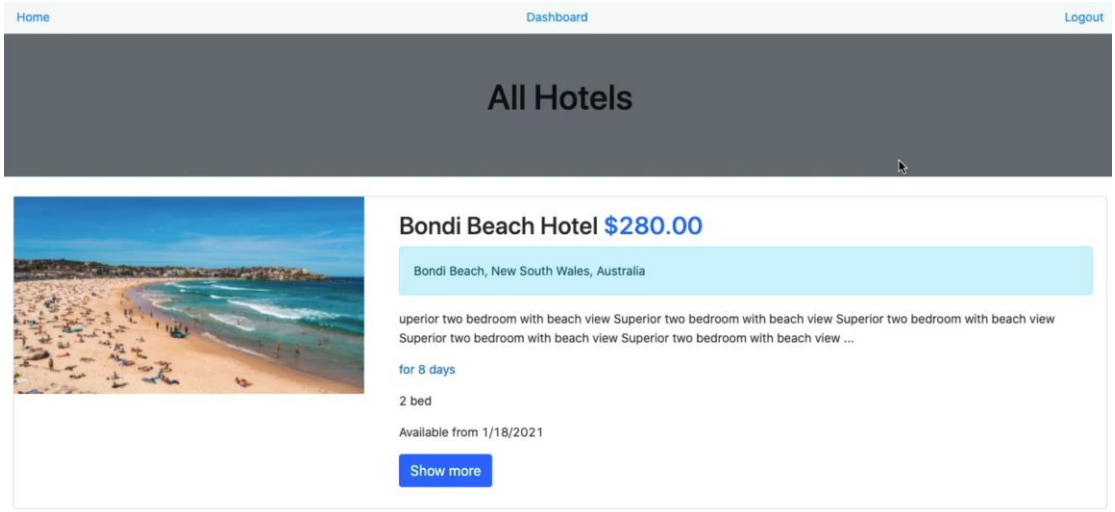
Στον προγραμματισμό υπολογιστών, η δημιουργία, η ανάγνωση, η ενημέρωση και η διαγραφή (CRUD) είναι οι τέσσερις βασικές λειτουργίες της μόνιμης αποθήκευσης. Το CRUD χρησιμοποιείται επίσης μερικές φορές για να περιγράψει συμβάσεις διεπαφής χρήστη που διευκολύνουν την προβολή, την αναζήτηση και την αλλαγή πληροφοριών

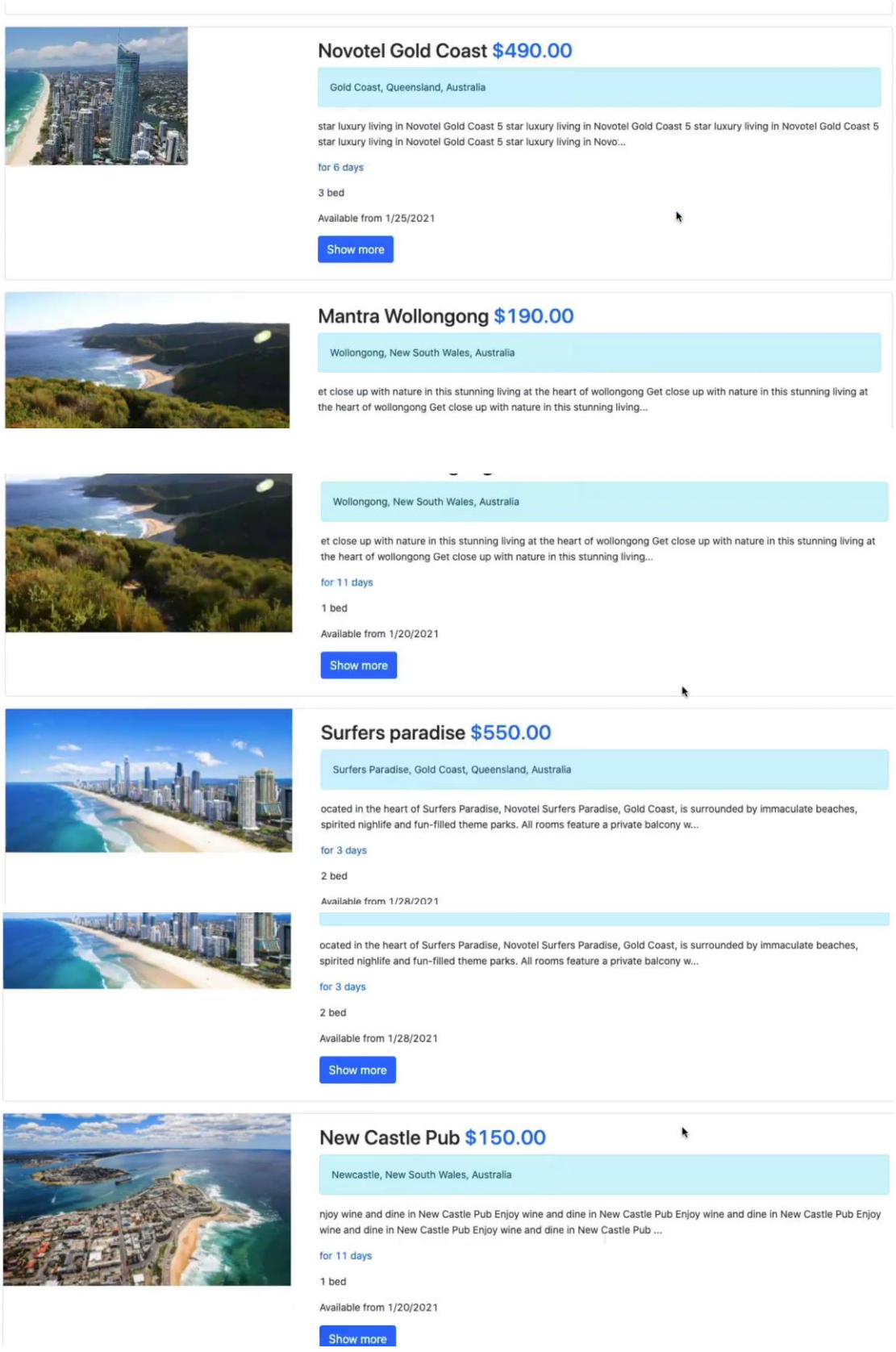
χρησιμοποιώντας φόρμες και αναφορές που βασίζονται σε υπολογιστή. Ο όρος πιθανότατα διαδόθηκε για πρώτη φορά από τον James Martin στο βιβλίο του το 1983 Managing the Data-base περιβάλλον.

Προσθήκη ενός ξενοδοχείου



Εμφάνιση ενός ξενοδοχείου





Για την διεπαφή του χρήστη το CRUD χρησιμοποιείται για την:

- Δημιουργία ή προσθήκη μιας νέας καταχώρησης.

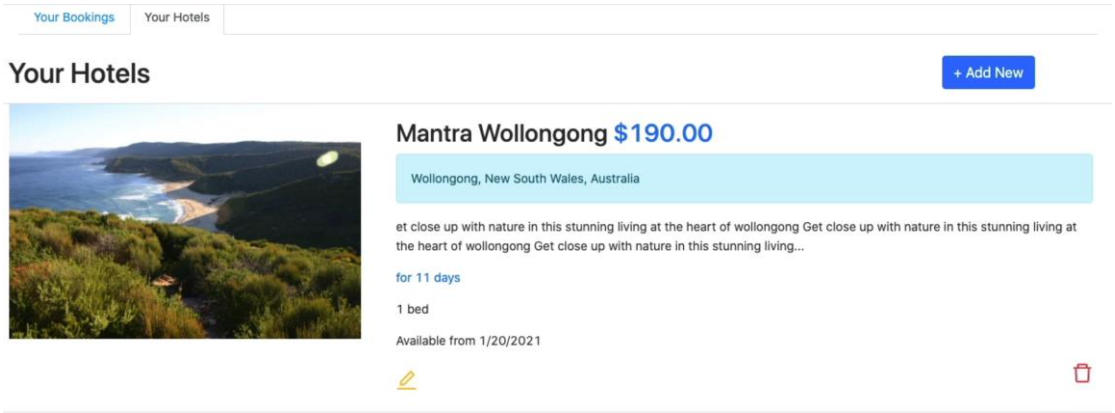
- Ανάγνωση, ανάκτηση, αναζήτηση ή προβολή υπάρχοντών εγγραφών.
- Ενημέρωση ή επεξεργασία της υπάρχουσας καταχώρησης.
- Διαγραφή, απενεργοποίηση ή αφαίρεση υπάρχουσας καταχώρησης.

Για τα RESTful APIs τα CRUD μεταφράζονται ως εξής:

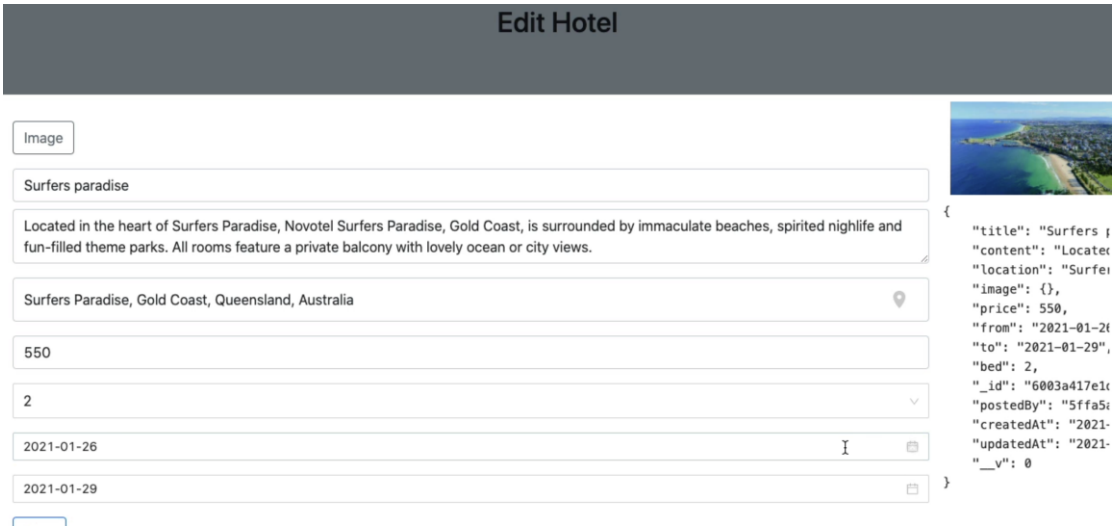
CREATE	READ	UPDATE	DELETE
POST, PUT	GET	PUT	DELETE

Στο HTTP, οι μέθοδοι GET (ανάγνωση), PUT (δημιουργία και ενημέρωση), POST (δημιουργία - αν δεν έχουμε `id` ή uuid`) και DELETE (διαγραφή) είναι λειτουργίες CRUD καθώς έχουν σημασιολογία διαχείρισης αποθήκευσης , που σημαίνει ότι επιτρέπουν στους πράκτορες χρήστη να χειρίζονται απευθείας τις καταστάσεις των πόρων-στόχων. Η μέθοδος POST, από την άλλη πλευρά, είναι μια λειτουργία διεργασίας που έχει σημασιολογία συγκεκριμένης πηγής στόχου η οποία συνήθως υπερβαίνει το εύρος των λειτουργιών CRUD.

Προσθήκη της δυνατότητας ‘επεξεργασίας’ και ‘διαγραφής’ ξενοδοχείων



Σελίδα επεξεργασίας ξενοδοχείου



4 ΑΠΟΤΕΛΕΣΜΑ

Φτάσαμε λοιπόν στο τέλος της εφαρμογής μας. Η πιο σημαντική επιδεξιότητα που μάθαμε είναι πως με σύγχρονες προγραμματιστικές τεχνολογίες, οι οποίες μας παρέχονται δωρεάν, μπορούμε να χτίσουμε ένα ισχυρό και γρήγορο πρόγραμμα που θα μας βοηθάει να ανεβάσουμε την επιχείρηση μας ένα σκαλοπάτι πιο πάνω. Η online προώθηση είναι ο πιο σημαντικός παράγοντας για την σωστή και επιτυχημένη λειτουργία ενός ξενοδοχείου. Με αυτό το πρόγραμμα επίσης, πέρα από την προώθηση ενός ξενοδοχείου, μπορούμε να το χρησιμοποιήσουμε και για διάφορες επιχειρήσεις όπως κάποιο συνεργείο αμαξιών, εστιατόριο κλπ.

4.1 ΣΤΟΧΟΙ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Ο αρχικός στόχος ήταν, η εφαρμογή να έχει μια ολοκληρωμένη ροή από την αρχή μέχρι και το τέλος της, συμφώνα με τις ανάγκες ενός διαχειριστή ξενοδοχείου και ενός χρήστη που επιθυμεί να προβάλει από μόνος του τα δωμάτια του ξενοδοχείου του.

Κάποιοι στόχοι που θα επιθυμούσα να θέσω είναι:

- **Αναζήτηση ξενοδοχείων**
  Στην αρχή ήταν εφικτό με το API της algolia αλλά στην πορεία κάποια αναβάθμιση έγινε και δεν υπήρχε πλέον δωρεάν τρόπος για να κάνεις οποιαδήποτε αναζήτηση. Η χρήση κάποιου άλλου API ίσως βοηθούσε, αλλά το algolia είναι το πιο γνωστό.
- **Χαρακτηριστικά όπως:**
  Αξιολόγηση ξενοδοχείων με αστέρια και ημερολόγιο που να εμφανίζει τις ημερομηνίες που δεν είναι διαθέσιμα τα δωμάτια.
- **Χρήση της stripe**
  Αυτός ίσως να είναι και ο κυριότερος στόχος. Μέσω stripe θα υπάρχει δυνατότητα της online κράτησης και πληρωμής.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Μέχρι τώρα η εφαρμογή καλύπτει ένα μεγάλο πλαίσιο στοιχείων για την προώθηση μιας τοπικής επιχείρησης. Παρόλα αυτά η εφαρμοφή δεν μπορεί να θεωρηθεί ένα ολοκληρωμένο έργο. Μπορεί όμως να είναι η αρχή για κάτι μεγαλύτερο. Έχει τις δυνατότητες να κάνει κάτι τέτοιο γιατί χρησιμοποιεί κάποιες από τις πιο σύγχρονες τεχνολογίες προγραμματισμού. Η React μπορεί να θεωρείται ένα δύσκολο framework για να μάθει κάποιος, αλλά είναι για πολλά χρόνια πρώτο στην λίστα με τα καλύτερα frameworks που κυκλοφορούν. Το πιο σημαντικό είναι επίσης πως είναι δωρεάν για όλους τους χρήστες. Υπάρχουν πολλές πληροφορίες στο διαδίκτυο που μπορούν να βοηθήσουν κάποιον να δουλέψει πάνω στην React. Η εργασία αυτή βοήθησε και εμένα στο να κατανοήσω από τις πιο απλές λειτουργίες όπως π.χ. πως πραγματικά λειτουργεί το διαδίκτυο, μέχρι και τις πιο σύγχρονες λειτουργίες του MERN Stack. Ελπίζω, αυτή η πτυχιακή να γίνει φάρος στην επίτευξη του στόχου μου και να φωτίσει τον δρόμο της επαγγελματικής μου πορείας.

ΑΝΑΦΟΡΕΣ

https://codersera.com/blog/npx-vs-npm-a-comparison/

https://www.pluralsight.com/guides/pros-and-cons-of-client-side-routing-with-react

https://www.youtube.com/watch?v=qt6gSW-uYKI

https://getbootstrap.com/

https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835/#how

https://www.geeksforgeeks.org/what-are-the-advantages-of-using-mongoose-module/

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

https://kaloraat.com/

https://www.geeksforgeeks.org/reactjs-components/

https://www.mulesoft.com/resources/api/what-is-an-api

https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

Και tutorials από το YouTube.