ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός υλοποίηση εφαρμογής σύγκρισης και αξιολόγησης δυνατοτήτων αυτοκινήτων με αυτοκίνητων με JavaScript

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

των

ΧΑΚΑΝ ΓΙΟΥΣΟΥΦ (2400)

ΣΤΡΟΦΥΛΑΣ ΝΙΚΟΛΑΟΣ (2404)

Επιβλέπων : ΔΟΣΗΣ ΜΙΧΑΗΛ

ΚΑΘΗΓΗΤΗΣ

Καστοριά Μάρτιος, 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός υλοποίηση εφαρμογής σύγκρισης και αξιολόγησης δυνατοτήτων αυτοκινήτων με αυτοκίνητων με JavaScript

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

των

ΧΑΚΑΝ ΓΙΟΥΣΟΥΦ (2400)

ΣΤΡΟΦΥΛΑΣ ΝΙΚΟΛΑΟΣ (2404)

Επιβλέπων : Μιχαήλ Δόσης

Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5 Απριλίου 2023

.....	.....	.....
Μιχαήλ Δόσης	Ιωάννης Βαρδάκας	Δημήτριος Ι. Βέργαδος
Καθηγητής	Αναπληρωτής Καθηγητής	Επίκουρος Καθηγητής

Καστοριά Μάρτιος - 2023

Περίληψη

Τα τελευταία χρόνια η Πληροφορική (IT) έχει διεισδύσει σε πολλές βιομηχανίες και μία από αυτές είναι οι διαδικτυακές εφαρμογές. Η ανάλυση των δυνατοτήτων δύο ή περισσότερων αυτοκινήτων είναι μια τέτοια εφαρμογή. Η κύρια δυνατότητα αυτής της εφαρμογής είναι ότι ένας χρήστης μπορεί να αποκτήσει γνώσεις που μπορούν να τον βοηθήσουν σε μια μελλοντική αγορά αυτοκινήτου, με τη χρήση μόνο ενός κουμπιού. Αυτό που διαφέρει σε αυτή τη μελέτη είναι η δημιουργία ενός συστήματος διαχείρισης χρηστών, με τη βοήθεια μιας βάσης δεδομένων, προκειμένου κάθε χρήστης να μπορεί να συνδεθεί με τα προσωπικά του στοιχεία. Έχει παρατηρηθεί ότι οι άνθρωποι εμπιστεύονται όλο και περισσότερο τέτοιες εφαρμογές καθώς οι χρήστες έχουν άμεση επαφή με την πληροφορία. Συμπερασματικά, με τον ιλιγγιώδη ρυθμό που εξελίσσονται οι εφαρμογές και τον υπάρχοντα ανταγωνισμό, είναι επιτακτική η ανάγκη να συνεχίσουμε να δημιουργούμε καινοτόμα και χρήσιμα εργαλεία.

Λέξεις Κλειδιά: Διαδικτυακή εφαρμογή, Βάσεις Δεδομένων, JavaScript, Node.Js, Express.Js, MongoDB, mongoose, React.Js, Redux, html, CSS, JSON Web Token (JWT), Postman

Abstract

In recent years Information Technology (IT) has penetrated many industries and one of them is web applications. Analysing the capabilities of two or more cars is such an application. Main capability of this application is that a user can gain knowledge that can help them in a future car purchase, with just the use of a single button. What is different in this study is the creation of a user management system, with the help of a database, in order each user to be able to connect to their personal information. It has been observed that people increasingly trust such applications as users have direct contact with information. Concluding, with the dizzying pace that applications are evolving and the existent competition it's more imperative than ever to keep creating innovative and useful tools.

Key Words: Web application, Data Bases, JavaScript, Node.Js, Express.Js, MongoDB, mongoose, React.Js, Redux, html, CSS, JSON Web Token (JWT), Postman

Πίνακας Περιεχομένων

Λίστα Εικόνων

Εισαγωγή

Στόχος της εργασίας είναι ο σχεδιασμός, η ανάπτυξη και η δημιουργία μίας JavaScript εφαρμογής για την σύγκριση δύο ή περισσότερων αυτοκινήτων. Απαραίτητη προϋπόθεση για την λειτουργία μιας τέτοιας εφαρμογής είναι η δημιουργία βάσης δεδομένων ώστε να μπορούν να διαχειριστούν και να αποθηκεύονται τα δεδομένα είτε των αυτοκινήτων είτε των χρηστών για να μπορούμε να τα αξιοποιήσουμε κατάλληλα. Στην περίπτωση μας χρησιμοποιήσαμε μία βάση δεδομένων MongoDB και πιο συγκεκριμένα μία βιβλιοθήκη της, την mongoose. Για την υλοποίηση της εφαρμογής χρησιμοποιείται επίσης nodejs, μία τεχνολογία JavaScript που βοηθάει στην δημιουργία του backend. Πιο συγκεκριμένα η εργασία αυτή εφαρμόστηκε με το framework της nodejs, το express js που είναι ένα εργαλείο που μας βοηθάει να δημιουργήσουμε τα APIs μας με πολύ γρήγορο τρόπο. Στην πορεία για την ασφάλεια της εφαρμογής χρησιμοποιήθηκε το JWT (json web token), μία τεχνολογία που βοηθάει τους χρήστες να ταυτοποιηθούν με μεγαλύτερη ασφάλεια.

Επίσης, για την δημιουργία του frontend επιλέχθηκε η χρήση της React js, μίας βιβλιοθήκης της JavaScript που βοηθάει στην ένωση του backend με το frontend μέσω των APIs που δημιουργήθηκαν στο backend της εφαρμογής. Τέλος, για να είναι πιο ευπαρουσίαστη η εφαρμογή τα απαραίτητα εργαλεία είναι η html και η CSS, το πρώτο για τη δημιουργία των containers και το δεύτερο για να τη διαχείριση του design. Στο πρώτο κεφάλαιο, παρουσιάζονται όλες οι τεχνολογίες, που χρειάστηκαν για την υλοποίηση της εργασίας. Στο δεύτερο κεφάλαιο, αναφέρεται η διαδικασία υλοποίησης του πρακτικού μέρους της εργασίας. Τέλος, στο τρίτο κεφάλαιο, παρουσιάζεται η διαδικασία εγκατάστασης με την οποία ο χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή.

Κεφάλαιο 1ο: Τεχνολογίες του project

Ο στόχος του πρώτου κεφαλαίου της παρούσας εργασίας είναι η παρουσίαση των βασικών τεχνολογιών που θα χρησιμοποιηθούν στο πλαίσιο της εν λόγω εργασίας. Πιο συγκεκριμένα, θα αναλυθούν λεπτομερώς τα εργαλεία, οι τεχνολογίες και οι τεχνοτροπίες που αποτελούν τη βάση αυτής της εργασίας, όπως είναι ο HTML, ο CSS, ο ReactJS, ο Redux, ο Nodjes, ο Express, ο MongoDB, η βιβλιοθήκη Mongoose, το MongoDB Compass, το Postman, το JSON Web Token, το VScode, το Canva, ο Responsive CSS, οι Ux&Ui, το CanIUse και ο Compressor. Τέλος, θα γίνει μια ανασκόπηση του τρόπου με τον οποίο οι παραπάνω τεχνολογίες μπορούν να συνδυαστούν για να δημιουργήσουν μια εφαρμογή με εξαιρετικές δυνατότητες.

1.1 Τι είναι το MongoDB;

Εικόνα 1 . Logo MongoDB

Το MongoDB είναι μια δωρεάν βάση δεδομένων ανοιχτού κώδικα προσανατολισμένη σε έγγραφα που μπορεί να χωρέσει σημαντικό όγκο δεδομένων, επιτρέποντάς σας παράλληλα να το αντιμετωπίσετε γρήγορα. Αποθηκεύει και ανακτά δεδομένα με τη μορφή εγγράφων και όχι πινάκων, γι 'αυτό ονομάζεται βάση δεδομένων NoSQL (Daragh Ó Tuama, 2022).

Η MongoDB χρησιμοποιείται ευρέως για τη διαχείριση μεγάλου όγκου δεδομένων και ταχεία ανάκτηση δεδομένων. Επίσης, η MongoDB είναι πολύ ευέλικτη και επιτρέπει στους προγραμματιστές να αλλάζουν εύκολα τη μορφή των δεδομένων χωρίς να χρειάζεται να αλλάξουν τη δομή της βάσης δεδομένων. Η MongoDB χρησιμοποιείται ευρέως σε διάφορες εφαρμογές, όπως web και mobile εφαρμογές, παιχνίδια, και συστήματα ανάλυσης δεδομένων.

Έπειτα, μπορούν να γίνουν αναζητήσεις στη βάση δεδομένων για να βρεθούν όλα τα άρθρα που περιέχουν συγκεκριμένη λέξη-κλειδί ή όλα τα άρθρα που έχουν δημοσιευτεί από έναν συγκεκριμένο συγγραφέα. Αυτό επιτυγχάνεται μέσω της χρήσης της γλώσσας ερωτημάτων της MongoDB (MongoDB Query Language) ή μέσω ενός ORM (Object-

Relational Mapping) όπως το MongooseJS για NodeJS. Έχει σχεδιαστεί για την ευκολία ανάπτυξης και κλιμάκωσης εφαρμογών.

Παράδειγμα :

```
{
    "id": 8,
    "name": "Nikos Strofyllas",
    "age": 26
}
{
    "id": 9,
    "name": "Xakan Giousouf",
    "age": 30
}
```

Μερικά από τα πλεονεκτήματα της MongoDB είναι:

1. Ευκολία χρήσης: Η MongoDB είναι εύκολη στη χρήση και δεν απαιτεί πολλές γνώσεις SQL.

2. Ευελιξία: Η MongoDB μπορεί να χρησιμοποιηθεί για να αποθηκευτούν δεδομένα σε διάφορες μορφές, όπως αντικείμενα JSON ή έγγραφα.

3. Υψηλή απόδοση: Η MongoDB είναι σχεδιασμένη για να παρέχει υψηλή απόδοση σε μεγάλους όγκους δεδομένων.

4. Συμβατότητα με τα περισσότερα συστήματα: Η MongoDB μπορεί να τρέξει σε περισσότερα συστήματα από τις παραδοσιακές σχεσιακές βάσεις δεδομένων.

5. Ανοιχτοί κώδικες: Η MongoDB είναι μια βάση δεδομένων ανοιχτού κώδικα, που σημαίνει ότι οι προγραμματιστές μπορούν να προσαρμόσουν το λογισμικό στις ανάγκες τους.

Μερικά από τα μειονεκτήματα της MongoDB είναι τα εξής:

1. Μη συμβατότητα με την SQL: Η MongoDB δεν χρησιμοποιεί τη γλώσσα SQL, που μπορεί να καθιστά δυσκολότερη τη μετάβαση από παραδοσιακές σχεσιακές βάσεις δεδομένων σε MongoDB.

2. Αναποτελεσματικότητα στην αναζήτηση: Στη MongoDB, η αναζήτηση βασίζεται σε κατακερματισμένα δεδομένα, και αυτό μπορεί να οδηγήσει σε αναποτελεσματική αναζήτηση, ειδικά σε περιπτώσεις μεγάλου όγκου δεδομένων.

3. Απώλεια δεδομένων: Η MongoDB μπορεί να χάσει δεδομένα σε περιπτώσεις αποτυχίας του διακομιστή, εάν δεν έχει διαμορφωθεί σωστά για τη διαχείριση αντιγράφων ασφαλείας.

1.2 Τι είναι το Mongoose;

Εικόνα 2. Logo Mongoose js

Το Mongoose είναι μια βιβλιοθήκη JavaScript που χρησιμοποιείται συνήθως μαζί με τη MongoDB για την ανάπτυξη εφαρμογών στο πλαίσιο του Node.js. Το Mongoose παρέχει ένα αντικειμενοστραφές προγραμματιστικό μοντέλο για τη MongoDB, το οποίο καθιστά ευκολότερη την ανάπτυξη εφαρμογών με τη χρήση της MongoDB.

Πολλοί που μαθαίνουν MongoDB εισάγονται σε αυτό μέσω της πολύ δημοφιλούς βιβλιοθήκης, Mongoose. Η μαγκούστα περιγράφεται ως "κομψή μοντελοποίηση αντικειμένων MongoDB για node.js " (Jesse Hall, 2022).

Η βιβλιοθήκη Mongoose προσφέρει μια σειρά από δυνατότητες, όπως ορισμός σχήματος δεδομένων, επικύρωση δεδομένων, middleware, τροποποιητές και άλλα, για να διευκολύνει την ανάπτυξη των εφαρμογών που χρησιμοποιούν MongoDB.

Συνολικά, το Mongoose βοηθά στην επιτάχυνση της ανάπτυξης των εφαρμογών που χρησιμοποιούν τη MongoDB και βελτιώνει την εμπειρία ανάπτυξης των προγραμματιστών. Το Mongoose είναι μια βιβλιοθήκη της MongoDB, που βοηθάει τους χρήστες να δομήσουν και να αποκτήσουν πρόσβαση στα δεδομένα τους με ευκολία.

Η Mongoose επιβάλλει ένα ημι-άκαμπτο σχήμα από την αρχή. Με το Mongoose, οι προγραμματιστές πρέπει να ορίσουν σχήμα και μοντέλο.

Τι είναι το σχήμα;

Ένα σχήμα καθορίζει τη δομή των εγγράφων της συλλογής σας. Ένα σχήμα Μαγκούστας αντιστοιχεί απευθείας σε μια συλλογή MongoDB.

Τι είναι το μοντέλο;

Τα μοντέλα παίρνουν το σχήμα σας και το εφαρμόζουν σε κάθε έγγραφο της συλλογής του.

Τα μοντέλα είναι υπεύθυνα για όλες τις αλληλεπιδράσεις εγγράφων, όπως η δημιουργία, η ανάγνωση, η ενημέρωση και η διαγραφή (CRUD).

Μια σημαντική σημείωση: το πρώτο όρισμα που μεταβιβάζεται στο μοντέλο πρέπει να είναι η μοναδική μορφή του ονόματος της συλλογής σας. Το Mongoose το αλλάζει αυτόματα στον πληθυντικό, το μετατρέπει σε πεζά και το χρησιμοποιεί για το όνομα της συλλογής βάσης δεδομένων.

Μερικά από τα θετικά της βιβλιοθήκης Mongoose είναι:

1. Επιτρέπει στους προγραμματιστές να ορίσουν σαφώς τη δομή των δεδομένων στο MongoDB και να επιβάλλουν κανόνες για τα δεδομένα που εισάγονται στη βάση δεδομένων.
2. Παρέχει πλήθος επιλογών για τον έλεγχο και την επεξεργασία των δεδομένων, όπως δυνατότητα εύρεσης εγγραφών, ενημέρωσης, διαγραφής και προβολής των δεδομένων.
3. Παρέχει εύκολη δυνατότητα σύνδεσης με τη βάση δεδομένων MongoDB, καθώς παρέχει πολλά εργαλεία και μεθόδους για τη διαχείριση των συνδέσεων και των δεδομένων.
4. Υποστηρίζει τη δημιουργία πολύπλοκων ερωτημάτων στη βάση δεδομένων, χρησιμοποιώντας τη γλώσσα ερωτημάτων MongoDB, η οποία είναι παρόμοια με τη γλώσσα SQL.

Μερικά από τα αρνητικά του Mongoose είναι τα εξής:

1. Πολυπλοκότητα: Η Mongoose είναι μια πολύπλοκη βιβλιοθήκη με πολλές λειτουργίες και επιλογές, οπότε μπορεί να απαιτηθεί αρκετός χρόνος και προσπάθεια για να μάθει κάποιος πώς να τη χρησιμοποιήσει σωστά.
2. Εξάρτηση από τη MongoDB: Η Mongoose είναι σχεδιασμένη να λειτουργεί με τη MongoDB και δεν υποστηρίζει άλλους τύπους βάσεων δεδομένων. Αυτό σημαίνει ότι οι προγραμματιστές πρέπει να χρησιμοποιούν τη MongoDB αν θέλουν να χρησιμοποιήσουν τη Mongoose.
3. Αλλαγή στη δομή της βάσης δεδομένων: Αν οι προγραμματιστές αλλάξουν τη δομή της βάσης δεδομένων, ίσως πρέπει να αλλάξουν και το σχήμα τους στη Mongoose. Αυτό μπορεί να είναι χρονοβόρο και δύσκολο, ειδικά σε μεγάλες εφαρμογές.

1.3 Τι είναι το MongoDB Compass

Το MongoDB επιτρέπει στους χρήστες να αλληλεπιδρούν με τον διακομιστή MongoDB μέσω δύο διεπαφών. Το πρώτο είναι το κέλυφος Mongo, το οποίο είναι η απλή διεπαφή γραμμής εντολών του διακομιστή MongoDB.

Το MongoDB Compass είναι ανοιχτού κώδικα και είναι διαθέσιμο για λειτουργικά συστήματα όπως Linux, Windows, macOS κ.λπ. Η πυξίδα MongoDB είναι μια τέλεια εναλλακτική λύση για το κέλυφος MongoDB, καθώς δεν είναι εφικτό να

πραγματοποιήσετε σύνθετα ερωτήματα με το κέλυφος MongoDB (Manjiri Gaikwad, 2022).

Το Compass είναι μία εφαρμογή που για να την χειριστούν οι χρήστες πρέπει να την κατεβάσουν στον υπολογιστή τους. Είναι ένα γραφικό, δια δραστικό περιβάλλον που βοηθάει να ελέγχει ο χρήστης τις βάσεις του, τους πίνακες του επίσης μπορεί να θέσει υποβολή ερωτημάτων, τη βελτιστοποίηση και την ανάλυση των δεδομένων. Με αυτόν τον τρόπο μπορεί να έχει μία βάση δεδομένων στον τοπικό του δίσκο και να την διαχειριστεί. Ταυτόχρονα όμως μπορεί να συνδεθεί με το Compass και με τον online λογαριασμό μας με πολύ απλό τρόπο. Δημιουργεί έναν λογαριασμό στο MongoDB Atlas | MongoDB και έπειτα πρέπει να συνδέσει την βάση του με την εφαρμογή Compass.

Με το MongoDB Compass, οι χρήστες μπορούν να δημιουργήσουν, να ενημερώσουν και να διαγράψουν δεδομένα, να δημιουργήσουν ευέλικτα ερωτήματα, να εξερευνήσουν τις σχέσεις μεταξύ των δεδομένων και να παρακολουθήσουν την απόδοση της βάσης δεδομένων. Το MongoDB Compass είναι επίσης εξοπλισμένο με δυνατότητες ασφαλείας που επιτρέπουν τον έλεγχο της πρόσβασης στις βάσεις δεδομένων και τη διαχείριση των ρυθμίσεων ασφαλείας. Επιπλέον, μπορεί να ενσωματωθεί με άλλα εργαλεία ανάλυσης δεδομένων, όπως το Tableau, το Excel και το Power BI.

Μερικά θετικά του MongoDB Compass είναι:

1. Εύκολο στη χρήση: Το MongoDB Compass προσφέρει μια εύχρηστη διεπαφή χρήστη για τη διαχείριση της MongoDB.
2. Οπτικοποίηση δεδομένων: Μπορεί να χρησιμοποιηθεί για την εμφάνιση και την ανάλυση των δεδομένων MongoDB με διάφορους τρόπους οπτικοποίησης.
3. Ενσωμάτωση με τα άλλα εργαλεία MongoDB: Μπορεί να συνδεθεί με άλλα εργαλεία MongoDB, όπως το MongoDB Atlas, για την ευκολότερη διαχείριση των MongoDB.
4. Δυνατότητα επεξεργασίας δεδομένων: Μπορεί να χρησιμοποιηθεί για την επεξεργασία δεδομένων MongoDB με μια εύχρηστη διεπαφή.
5. Ασφαλής σύνδεση: Προσφέρει ασφαλή σύνδεση με τα MongoDB servers, χρησιμοποιώντας κρυπτογράφηση SSL/TLS.

Μερικά από τα αρνητικά του MongoDB Compass είναι τα εξής:

1. Υψηλή κατανάλωση πόρων: Ο MongoDB Compass χρειάζεται αρκετούς πόρους για να λειτουργήσει ομαλά, καθώς παρέχει πολλές λειτουργίες και επιλογές.
2. Περιορισμένη δυνατότητα προσαρμογής: Ορισμένοι χρήστες μπορεί να βρουν το MongoDB Compass λιγότερο ευέλικτο σε σχέση με άλλα εργαλεία MongoDB, καθώς περιορίζεται στις επιλογές που παρέχει.
3. Μη δυνατότητα αποθήκευσης παραμέτρων σύνδεσης: Δεν υπάρχει επιλογή για αποθήκευση των παραμέτρων σύνδεσης σε διαφορετικά προφίλ, που θα μπορούσε να είναι χρήσιμο για χρήστες με πολλαπλές βάσεις δεδομένων.
4. Περιορισμένη υποστήριξη για παλαιότερες εκδόσεις MongoDB: Η τρέχουσα έκδοση του MongoDB Compass υποστηρίζει μόνο MongoDB 3.6 και

μεταγενέστερες εκδόσεις, ενώ παλαιότερες εκδόσεις της βάσης δεδομένων μπορεί να μην υποστηρίζονται πλήρως ή καθόλου.

1.4 Τι είναι το Nodejs

Εικόνα 3. Logo Nodejs

Node.js είναι ένα περιβάλλον της JavaScript που βασίζεται στη μηχανή JavaScript V8 του Chrome. Επιτρέπει στους προγραμματιστές να δημιουργούν γρήγορες, κλιμακούμενες εφαρμογές JavaScript. Η ασύγχρονη και καθοδηγούμενη από συμβάντα φύση του Node.js το καθιστά κατάλληλο για τη δημιουργία εφαρμογών που μπορούν να χειριστούν πολλές συνδέσεις ταυτόχρονα, μια κοινή εργασία για διακομιστές ιστού και άλλες εφαρμογές back-end (Brian Boucheron, 2021).

Αξιοποιώντας το JavaScript τόσο στο frontend όσο και στο backend, η ανάπτυξη μπορεί να είναι πιο συνεπής και οι εφαρμογές web πλήρους στοίβας μπορούν να σχεδιαστούν μέσα στο ίδιο περιβάλλον ανάπτυξης.

Το Node.js είναι μια πλατφόρμα ανοιχτού κώδικα για την εκτέλεση κώδικα JavaScript στην πλευρά του server. Βασίζεται στον JavaScript, μία διαδεδομένη γλώσσα προγραμματισμού στο web, και επιτρέπει στους προγραμματιστές να γράφουν εντολές στη γλώσσα αυτή για τη δημιουργία server-side εφαρμογών.

Με το Node.js, οι προγραμματιστές μπορούν να δημιουργήσουν server-side εφαρμογές με JavaScript και να χρησιμοποιήσουν μια μεγάλη ποικιλία από πακέτα και βιβλιοθήκες που υποστηρίζονται από την κοινότητα του Node.js. Επίσης, το Node.js υποστηρίζει τη χρήση του non-blocking I/O model, που του επιτρέπει να ανταποκρίνεται σε αιτήσεις από πολλούς χρήστες ταυτόχρονα χωρίς να κλειδώνει την εκτέλεση των υπολοίπων αιτημάτων. Η τεχνολογία αυτή χρησιμοποιείται αποκλείστηκα για backend εφαρμογές. Η Node.js φημίζεται για την ταχύτητα της, την εξοικονόμηση πόρων που επιτρέπει να πετύχουν οι χρήστες με την χρήση της. Ως ασύγχρονος χρόνος εκτέλεσης JavaScript που βασίζεται σε συμβάντα, το Node.js έχει σχεδιαστεί για τη δημιουργία εφαρμογών δικτύου με δυνατότητα κλιμάκωσης. Ο Ασύγχρονος χρόνος εκτέλεσης είναι ένα από τα πλεονεκτήματα της Node.js διότι με αυτόν τον τρόπο δεν περιμένει μια αίτηση από τον χρήστη και μόλις την τελειώσει να προχωρήσει στην επόμενη, αλλά δέχεται πολλές αιτήσεις ταυτόχρονα.

Τα θετικά του Node.js περιλαμβάνουν:

1. Υψηλή απόδοση: Το Node.js είναι γραμμένο σε C++ και χρησιμοποιεί μια μη-αποκλειστική μοντέλο εκτέλεσης επεξεργασίας, το οποίο του επιτρέπει να χειρίζεται μεγάλα όγκους δεδομένων με μεγάλη ταχύτητα.
2. Εύκολη διαμόρφωση και ανάπτυξη: Η πλατφόρμα προσφέρει μια εύκολη στη χρήση σειρά διαδικασιών ανάπτυξης, συμπεριλαμβανομένου του npm (Node Package Manager) για τη διαχείριση εξαρτήσεων και του πλούσιου οικοσυστήματος πακέτων για την ανάπτυξη εφαρμογών.
3. Μεγάλη κοινότητα προγραμματιστών: Το Node.js έχει μια μεγάλη και ενεργή κοινότητα προγραμματιστών που παρέχουν βοήθεια και υποστήριξη, αναπτύσσουν νέα πακέτα, δίνοντας λύση στα εκάστοτε προβλήματα.
4. Ισχυρή δυνατότητα δικτύωσης: Η δυνατότητα του Node.js να χειρίζεται ασύγχρονες λειτουργίες επιτρέπει στους προγραμματιστές να δημιουργούν αποτελεσματικές δικτυακές εφαρμογές και υπηρεσίες.

Μερικά από τα αρνητικά του Node.js είναι:

1. Μνήμη: Η χρήση μνήμης στο Node.js είναι υψηλή. Αυτό μπορεί να οδηγήσει σε προβλήματα απόδοσης, εάν η εφαρμογή σας δεν είναι σωστά σχεδιασμένη και υλοποιημένη.
2. Δυσκολία στο debugging: Λόγω της ασύγχρονης φύσης του Node.js, το debugging μπορεί να είναι πιο δύσκολο από ό, τι σε άλλες πλατφόρμες. Αυτό οφείλεται στο γεγονός ότι οι συναρτήσεις εκτελούνται ασύγχρονα και οι λεπτομέρειες του προγράμματος είναι δυσκολότερο να εντοπιστούν.

1.5 Τι είναι το Express.js

Εικόνα 4. Logo Express.js

Το ExpressJS είναι ένα πλαίσιο εφαρμογών ιστού για το NodeJS. Αυτό είναι που κάνει κυρίως τη διαφορά μεταξύ Express JS και Node JS. Το πρώτο παρέχει διάφορες δυνατότητες που κάνουν την ανάπτυξη εφαρμογών ιστού γρήγορη και εύκολη, η οποία διαφορετικά απαιτεί περισσότερο χρόνο χρησιμοποιώντας μόνο τη δεύτερη. Παρέχει ένα ισχυρό σύνολο δυνατοτήτων για εφαρμογές ιστού και κινητών συσκευών (Oleg Kopachovets, 2022).

Το ExpressJS μπορεί να χρησιμοποιηθεί για τη δημιουργία υπηρεσιών σε πραγματικό χρόνο, όπως εφαρμογές συνομιλίας. Υποστηρίζει WebSockets, τα οποία σας επιτρέπουν να μεταδίδετε δεδομένα από τον διακομιστή σας στον πελάτη σας σε πραγματικό χρόνο χρησιμοποιώντας μόνιμες συνδέσεις.

Το Express.js είναι ένα πλαίσιο εφαρμογών του Node.js που χρησιμοποιείται για τη δημιουργία διακομιστών (servers) και εφαρμογών στο web. Αποτελεί ένα από τα πιο δημοφιλή πλαίσια εφαρμογών για τον χειρισμό αιτημάτων (requests) και αποστολή αποκρίσεων (responses) στο web, καθώς παρέχει μια απλή και ευέλικτη δομή για τη δημιουργία εφαρμογών.

Το Express.js παρέχει πολλά χαρακτηριστικά όπως routing, middleware, αντιμετώπιση σφαλμάτων, και πακετάρισμα (packaging) της εφαρμογής. Επιπλέον, υποστηρίζει πολλά plugins και middleware που μπορούν να βελτιώσουν τη λειτουργικότητα της εφαρμογής.

Το Express.js είναι επίσης ελαφρύ και αποτελεσματικό στην απόδοση, επιτρέποντας στους προγραμματιστές να δημιουργήσουν γρήγορα και εύκολα εφαρμογές που μπορούν να ανταποκριθούν σε μεγάλες ποσότητες αιτημάτων.

Επιπλέον, το Express.js έχει μια μεγάλη κοινότητα ανοιχτού κώδικα, που σημαίνει ότι υπάρχουν πολλοί plugins και βοηθητικά εργαλεία διαθέσιμα για την επέκταση της λειτουργικότητας της εφαρμογής.

Το Express είναι ένα ευέλικτο πλαίσιο εφαρμογών ιστού Node.js που παρέχει ένα ισχυρό σύνολο χαρακτηριστικών για εφαρμογές ιστού και κινητών συσκευών.

Με μία πληθώρα μεθόδων χρησιμότητας HTTP και ενδιάμεσου λογισμικού (middleware), η δημιουργία ενός REST API (τα οποία είναι ένας από τους πιο συνηθισμένους τρόπους ανταλλαγής δεδομένων από διακομιστές με πελάτες) είναι γρήγορη και εύκολη. Το Express επίσης παρέχει ένα λεπτό επίπεδο θεμελιωδών λειτουργιών εφαρμογών ιστού, χωρίς να αποκρύπτει τις δυνατότητες του Node.js. Το Express είναι μια εξαιρετική λύση για εφαρμογές μικρού έως μεσαίου μεγέθους επειδή είναι ελαφρύ και πολύ εύκολο στην εφαρμογή. Το Express μπορεί επίσης να χρησιμοποιηθεί για την εξυπηρέτηση στατικών αρχείων και έχει χρησιμοποιηθεί σε περιβάλλοντα παραγωγής για τη διαχείριση εκατοντάδων χιλιάδων ταυτόχρονων συνδέσεων.

Κάποια από τα θετικά του Express.js είναι τα εξής:

1. Εύκολη μάθηση και χρήση: Οι περισσότεροι προγραμματιστές μπορούν να μάθουν το Express.js σε σύντομο χρονικό διάστημα και να το χρησιμοποιήσουν στα project τους.
2. Επεκτασιμότητα: Το Express.js παρέχει μια σειρά από middleware που δίνουν τη δυνατότητα επέκτασης της λειτουργικότητάς του. Τα middleware μπορούν να προστεθούν ή να αφαιρεθούν εύκολα ανάλογα με τις ανάγκες του project.
3. Ενιαία αρχιτεκτονική: Το Express.js ακολουθεί το μοντέλο MVC (Model-View-Controller) για την ανάπτυξη εφαρμογών. Αυτό κάνει ευκολότερη τη διαχείριση του κώδικα και την ανάπτυξη πολύπλοκων εφαρμογών.
4. Απλό API: Το Express.js παρέχει ένα απλό API για τη διαχείριση HTTP αιτημάτων και απαντήσεων. Αυτό καθιστά ευκολότερη την ανάπτυξη RESTful εφαρμογών.

Μερικά από τα αρνητικά του Express.js είναι:

1. Δεν παρέχει πλήρη λειτουργικότητα σαν framework - Το Express.js είναι ένα ελαφρύ framework και δεν παρέχει όλες τις λειτουργίες ενός πλήρους framework, όπως το Laravel ή το Ruby on Rails.
2. Απαιτεί ικανότητες προγραμματισμού - Η χρήση του Express.js απαιτεί ικανότητες προγραμματισμού στη JavaScript για να το χρησιμοποιήσετε αποτελεσματικά. Αυτό μπορεί να καθιστά την εκμάθηση του πλαισίου δυσκολότερη για αρχάριους προγραμματιστές.
3. Δεν είναι καλό για μεγάλα και πολύπλοκα συστήματα - Ενώ το Express.js είναι κατάλληλο για μικρότερα και πιο απλά συστήματα, δεν είναι το καλύτερο πλαίσιο για μεγάλα και πολύπλοκα συστήματα, όπου η ανάγκη για δομημένο κώδικα και οργάνωση είναι πολύ σημαντική.
4. Μπορεί να είναι επιρρεπές σε ασφάλεια - Το Express.js μπορεί να είναι επιρρεπές σε επιθέσεις ασφαλείας αν δεν χρησιμοποιηθεί σωστά και δεν προστατευθεί κατάλληλα. Είναι σημαντικό να λαμβάνονται υπόψη τα μέτρα ασφαλείας κατά τη χρήση του πλαισίου.

1.6 Τι είναι το Postman;

Εικόνα 5. Logo Postman

Το Postman είναι μια πλατφόρμα API για τη δημιουργία και τη χρήση API. Ο Postman απλοποιεί κάθε βήμα του κύκλου ζωής του API και βελτιστοποιεί τη συνεργασία, ώστε να μπορεί ο χρήστης να δημιουργεί καλύτερα API, πιο γρήγορα (postman, -).

Με το Postman υπάρχει δυνατότητα δημιουργίας αιτήσεων HTTP, αποστολή σε έναν εξυπηρετητή (server) και να έλεγχος των απαντήσεων που λαμβάνονται πίσω. Υπάρχει η δυνατότητα επιλογής του τύπου των αιτήσεων (π.χ. GET, POST, PUT, DELETE κλπ.), προσθήκη παραμέτρων, headers και body στις αιτήσεις, και εξερεύνηση της απόκρισης του εξυπηρετητή σε μια φιλική προς τον χρήστη διεπαφή. Επίσης, η δυνατότητα δημιουργίας και να εκτέλεση συλλογών από αιτήσεις, αυτοματοποίηση των διαδικασίων ελέγχου και η δημιουργία ένος περιβάλλον για διαφορετικές ρυθμίσεις εξυπηρετητών.

Η χρήση του Postman είναι ευρέως διαδεδομένη στη βιομηχανία λογισμικού και την ανάπτυξη APIs, καθώς βοηθά τους προγραμματιστές να επιβεβαιώσουν τη σωστή λειτουργία των APIs που δημιουργούν ή χρησιμοποιούν.

Μερικά από τα θετικά του Postman είναι:

1. Δυνατότητα δοκιμής και ανάπτυξης API: Ο Postman επιτρέπει στους προγραμματιστές να δοκιμάζουν, να αναπτύσσουν και να τεκμηριώνουν τα API τους με μια εύχρηστη διασύνδεση χρήστη.
2. Εξοικονόμηση χρόνου: Με το Postman, οι προγραμματιστές μπορούν να εξοικονομήσουν χρόνο, καθώς δεν χρειάζεται να χτίζουν ένα δικό τους εργαλείο δοκιμής.
3. Επαναχρησιμοποίηση δεδομένων δοκιμής: Οι προγραμματιστές μπορούν να αποθηκεύσουν τα δεδομένα δοκιμής τους στο Postman και να τα χρησιμοποιούν ξανά σε διαφορετικά αιτήματα.
4. Αναφορές και παρακολούθηση: Ο Postman παρέχει εκτεταμένες αναφορές και παρακολούθηση για να βοηθήσει τους προγραμματιστές να εντοπίζουν σφάλματα και να βελτιώνουν την απόδοση τους.

Μερικά από τα αρνητικά του Postman μπορεί να είναι:

1. Πολύπλοκη διεπαφή χρήστη: Το Postman έχει μια πολύπλοκη διεπαφή χρήστη που μπορεί να είναι δύσκολο να κατανοηθεί από αρχάριους χρήστες.
2. Υπερβολική εξάρτηση από το cloud: Η συνδρομή στο Postman Pro απαιτεί την εγγραφή σε ένα λογαριασμό cloud και την αποθήκευση των δεδομένων σε αυτόν τον λογαριασμό. Αυτό μπορεί να επηρεάσει την ασφάλεια των δεδομένων και να καθιστά δύσκολη την πρόσβαση, αν δεν έχετε πρόσβαση στο cloud.
3. Προβλήματα ασφάλειας: Το Postman μπορεί να επηρεάσει την ασφάλεια των δεδομένων των χρηστών αν δεν έχουν πρόσβαση στο cloud και μπορεί να προκαλέσει προβλήματα στον server τους αν χρησιμοποίησουν κακόβουλο λογισμικό ή κακόβουλα δεδομένα.
4. Περιορισμένοι συνδυασμοί εργασιών: Το Postman δεν παρέχει τη δυνατότητα να συνδυαστούν εργασίες μεταξύ τους, όπως η διαχείριση εργασιών σε μία σειρά ή η αυτοματοποίηση των διαδικασιών εργασίας.

1.7 Τι είναι το JSON Web Token (JWT)

Εικόνα 6. Logo JSON Web Token

Το JSON Web Token (JWT) είναι ένα ανοιχτό πρότυπο (RFC 7519) που καθορίζει έναν συμπαγή και αυτόνομο τρόπο για την ασφαλή μετάδοση πληροφοριών μεταξύ των μερών ως αντικείμενο JSON. Αυτές οι πληροφορίες μπορούν να επαληθευτούν και να θεωρηθούν αξιόπιστες επειδή είναι ψηφιακά υπογεγραμμένες. Τα JWTs μπορούν να υπογραφούν χρησιμοποιώντας ένα μυστικό (με αλγόριθμο HMAC) ή ένα ζεύγος δημόσιου/ιδιωτικού κλειδιού χρησιμοποιώντας RSA (Auth0, -).

Τα JWTs αποτελούνται από τρία μέρη: το header, το payload και η υπογραφή.

Το header περιέχει πληροφορίες σχετικά με τον τύπο του token και τον αλγόριθμο υπογραφής που χρησιμοποιείται. Το payload περιέχει τα δεδομένα που αφορούν τον χρήστη ή την εφαρμογή που τον χρησιμοποιεί. Η υπογραφή είναι ένα κρυπτογραφημένο κομμάτι δεδομένων που χρησιμοποιείται για να επιβεβαιώσει ότι τα δεδομένα που περιέχονται στο token δεν έχουν τροποποιηθεί από κανέναν τρίτο.

Τα JWTs χρησιμοποιούνται ευρέως για την αυθεντικοποίηση χρηστών και τον έλεγχο της πρόσβασης σε πόρους σε μια εφαρμογή. Επιπλέον, μπορούν να χρησιμοποιηθούν για τη μεταφορά δεδομένων ανάμεσα σε εφαρμογές ή για την επικοινωνία με ένα API.

Το JSON Web Token (JWT) είναι ένα ανοιχτό πρότυπο που καθορίζει έναν συμπαγή και αυτόνομο τρόπο για την ασφαλή μετάδοση πληροφοριών μεταξύ των μερών ως αντικείμενο JSON. Αυτές οι πληροφορίες μπορούν να επαληθευτούν και να θεωρηθούν αξιόπιστες επειδή είναι ψηφιακά υπογεγραμμένες. Ουσιαστικά είναι ένα σύστημα προστασίας δεδομένων. Μερικά συνηθισμένα σενάρια είναι η εξουσιοδότηση και η ανταλλαγή πληροφοριών. Η εξουσιοδότηση είναι το πιο συνηθισμένο σενάριο για τη χρήση του JWT. Μόλις συνδεθεί ο χρήστης, κάθε επόμενο αίτημα θα περιλαμβάνει το JWT, επιτρέποντας στον χρήστη να έχει πρόσβαση σε διαδρομές, υπηρεσίες και πόρους που επιτρέπονται με αυτό το διακριτικό.

Επίσης ένα σενάριο που χρησιμοποιείται είναι η ανταλλαγή πληροφοριών με τα JSON Web Token είναι ένας καλός τρόπος ασφαλούς μετάδοσης πληροφοριών μεταξύ των μερών. Επειδή τα JWT μπορούν να υπογραφούν - για παράδειγμα, χρησιμοποιώντας ζεύγη δημόσιων / ιδιωτικών κλειδιών - μπορείτε να είστε σίγουροι ότι οι αποστολείς είναι αυτοί που λένε ότι είναι. Επιπλέον, καθώς η υπογραφή υπολογίζεται χρησιμοποιώντας την κεφαλίδα και το ωφέλιμο φορτίο, μπορείτε επίσης να επαληθεύσετε ότι το περιεχόμενο δεν έχει αλλοιωθεί.

Κάποια από τα θετικά του JSON Web Token (JWT) είναι τα εξής:

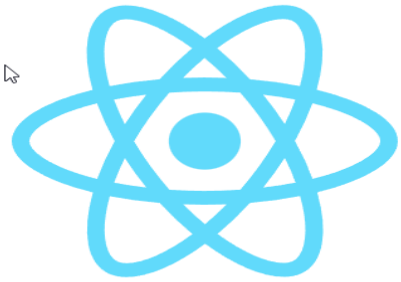
1. Ασφαλής επικύρωση: Τα JWTs είναι κρυπτογραφημένα και υπογεγραμμένα με ένα μυστικό κλειδί, επιτρέποντας στους παραλήπτες να επιβεβαιώνουν ότι οι πληροφορίες στο token είναι αυθεντικές και ασφαλείς.
2. Ευελιξία: Τα JWTs είναι πολύ ευέλικτα και μπορούν να προσαρμοστούν για πολλές διαφορετικές χρήσεις. Μπορούν να περιλαμβάνουν οποιαδήποτε πληροφορία θέλετε και μπορούν να χρησιμοποιηθούν σε πολλά σημεία στην εφαρμογή σας.

3. Αποθήκευση στον πελάτη: Τα JWTs αποθηκεύονται στον πελάτη, επομένως δεν χρειάζεται να αποθηκεύονται στον εξυπηρετητή, μειώνοντας έτσι την κίνηση δεδομένων στο δίκτυο.

Μερικά από τα αρνητικά του JWT είναι:

1. Μέγεθος: Τα JWT μπορεί να γίνουν αρκετά μεγάλα, καθώς περιλαμβάνουν πολλά δεδομένα, συμπεριλαμβανομένων των πληροφοριών χρήστη και μεταδεδομένων. Αυτό μπορεί να οδηγήσει σε αργούς χρόνους φόρτωσης και απόκρισης για τον server.
2. Περιορισμένη ασφάλεια: Τα JWT είναι επικυρωμένα μόνο από την υπογραφή τους, που μπορεί να παραβιαστεί αν οι κλειδικοί συνδυασμοί δεν είναι ισχυροί ή δεν έχουν υλοποιηθεί σωστά.
3. Αδύναμος έλεγχος εκπροσώπησης: Αν ένα JWT κλαπεί από έναν κακόβουλο χρήστη, αυτός ο χρήστης μπορεί να προσποιηθεί την ταυτότητα του αρχικού χρήστη και να αποκτήσει πρόσβαση σε ευαίσθητες πληροφορίες.
4. Προβλήματα αποστολής: Αν ένα JWT χαθεί κατά τη διάρκεια της μεταφοράς, τότε ο χρήστης θα πρέπει να συνδεθεί ξανά, καθώς δεν μπορεί να ανακτηθεί.

1.8 Τι είναι η React;



Εικόνα 7. Logo React

Η React είναι μια βιβλιοθήκη ανάπτυξης περιβάλλοντος εργασίας χρήστη που βασίζεται σε JavaScript. Αν και η React είναι μια βιβλιοθήκη και όχι μια γλώσσα, χρησιμοποιείται ευρέως στην ανάπτυξη ιστοσελίδων. Η βιβλιοθήκη εμφανίστηκε για πρώτη φορά τον Μάιο του 2013 και είναι πλέον μία από τις πιο συχνά χρησιμοποιούμενες βιβλιοθήκες frontend για την ανάπτυξη ιστοσελίδων.

Η React προσφέρει διάφορες επεκτάσεις για αρχιτεκτονική υποστήριξη ολόκληρης της εφαρμογής, όπως το Flux και το React Native, πέρα από το απλό περιβάλλον εργασίας χρήστη (Chinmayee Deshpande, 2023).

Η React είναι μια open-source JavaScript βιβλιοθήκη που χρησιμοποιείται για τη δημιουργία διεπαφών χρήστη (user interfaces ή UIs). Αναπτύχθηκε από την Facebook και αποτελεί μια από τις πιο δημοφιλείς επιλογές για τη δημιουργία εφαρμογών web και mobile. Η React βασίζεται σε ένα μοντέλο αντιδραστικής ανάπτυξης, όπου η διεπαφή αλληλοεπιδρά με τον χρήστη και ανανεώνεται δυναμικά όταν υπάρχουν αλλαγές στα δεδομένα.

Η React επιτρέπει την κατασκευή επαναχρησιμοποιήσιμων στοιχείων (components) με τη χρήση JavaScript και JSX, μια επέκταση της JavaScript που επιτρέπει την ανάμιξη κώδικα HTML και CSS στην JavaScript. Αυτό διευκολύνει την ανάπτυξη και συντήρηση του κώδικα, καθώς και τη βελτίωση της απόδοσης της εφαρμογής.

Η React είναι μία βιβλιοθήκη JavaScript για τη δημιουργία διεπαφών χρήστη. Οι δηλωτικές προβολές καθιστούν τον κώδικα σας πιο προβλέψιμο και πιο εύκολο στον εντοπισμό σφαλμάτων. Με την React μπορείτε να δημιουργήστε ενθυλακωμένα στοιχεία που διαχειρίζονται τη δική τους κατάσταση και, στη συνέχεια, συνθέστε τα για να δημιουργήσετε σύνθετα UI. Δεδομένου ότι η λογική στοιχείων είναι γραμμένη σε JavaScript αντί για πρότυπα, μπορείτε εύκολα να μεταφέρετε εμπλουτισμένα δεδομένα μέσω της εφαρμογής σας και να διατηρήσετε την κατάσταση εκτός του DOM. Δεν κάνουμε υποθέσεις για την υπόλοιπη τεχνολογική στοίβα σας, επομένως μπορείτε να αναπτύξετε νέες δυνατότητες στο React χωρίς να ξαναγράψετε τον υπάρχοντα κώδικα. Η React μπορεί επίσης να αποδώσει στο διακομιστή χρησιμοποιώντας εφαρμογές Node και power για κινητά χρησιμοποιώντας το React Native. Επίσης βοηθάει στην σύνδεση του backend και του frontend σε εφαρμογές με την χρήση APIs.

Η React έχει πολλά θετικά στοιχεία, μερικά από αυτά είναι τα εξής:

1. Υψηλή απόδοση: Η React είναι σχεδιασμένη να είναι γρήγορη και αποτελεσματική, επιτρέποντας την ανανέωση των στοιχείων σε πραγματικό χρόνο.
2. Επαναχρησιμοποίηση κώδικα: Η React επιτρέπει στους προγραμματιστές να επαναχρησιμοποιούν κώδικα σε πολλά μέρη της εφαρμογής τους, μειώνοντας έτσι τον όγκο του κώδικα που πρέπει να γραφτεί.
3. Ευελιξία: Η React επιτρέπει στους προγραμματιστές να επιλέξουν την πλατφόρμα και την τεχνολογία που θέλουν να χρησιμοποιήσουν στο πλαίσιο της εφαρμογής τους.
4. Απλότητα: Η React έχει απλή σύνταξη και είναι εύκολη στην κατανόηση και στη χρήση.
5. Αναβαθμισμένη ανάπτυξη εφαρμογών: Η React επιτρέπει στους προγραμματιστές να αναπτύξουν επαναχρησιμοποιήσιμα και συναρμολογούμενα στοιχεία, επιτρέποντας την εύκολη και αποδοτική ανάπτυξη εφαρμογών.

Μερικά αρνητικά στοιχεία του React είναι τα εξής:

1. Μάθηση: Μπορεί να απαιτηθεί περισσότερος χρόνος για να μάθει κάποιος η React σε σχέση με άλλα πλαίσια εργασίας, όπως το jQuery ή το AngularJS.

2. Πολυπλοκότητα: Η πολυπλοκότητα της React μπορεί να είναι αποθαρρυντική για αρχάριους προγραμματιστές.
3. Αστάθεια: Καθώς η React είναι μια σχετικά νέα τεχνολογία, υπάρχει η πιθανότητα να υπάρχουν ακόμα προβλήματα στην ανάπτυξη εφαρμογών.
4. Υποστήριξη: Επειδή η React είναι ανοιχτού κώδικα λογισμικό, η υποστήριξη εξαρτάται από την κοινότητα των προγραμματιστών. Αυτό μπορεί να σημαίνει ότι η υποστήριξη μπορεί να είναι περιορισμένη σε σύγκριση με τα εμπορικά πλαίσια εργασίας.

1.9 Τι είναι το Redux;

Εικόνα 8. Logo Redux

Το Redux είναι μία βιβλιοθήκη για εφαρμογές JavaScript. Βοηθά να συνταχθούν εφαρμογές που συμπεριφέρονται με συνέπεια, εκτελούνται σε διαφορετικά περιβάλλοντα (υπολογιστή-πελάτη, διακομιστή και εγγενή) και είναι εύκολο να δοκιμαστούν. Το Redux διαχειρίζεται την κατάσταση μιας εφαρμογής με ένα μόνο καθολικό αντικείμενο που ονομάζεται Store (Ravikiran A S, 2023).

Αναλυτικότερα, το Redux βασίζεται στην αρχή της μοναδικής πηγής της αλήθειας, όπου η κατάσταση της εφαρμογής αποθηκεύεται σε ένα αντικείμενο που δεν μπορεί να τροποποιηθεί από άλλες περιοχές του κώδικα. Αντίθετα, η κατάσταση μπορεί να ενημερωθεί μόνο μέσω ενεργειών (actions), τα οποία αναλύονται από έναν επεξεργαστή (reducer) και ενημερώνουν το αντικείμενο της κατάστασης.

Το Redux είναι ένα προβλέψιμο container κατάστασης για εφαρμογές JavaScript. Παρέχει μια εξαιρετική εμπειρία στον προγραμματιστή, καθώς με το Redux DevTools διευκολύνει τον εντοπισμό πότε, πού, γιατί και πώς άλλαξε η κατάσταση της εφαρμογής. Η αρχιτεκτονική του Redux επιτρέπει να καταγράφουν οι χρήστες αλλαγές, να χρησιμοποιούν "εντοπισμό σφαλμάτων ταξιδιού στο χρόνο" και ακόμη και να στέλνουν πλήρεις αναφορές σφαλμάτων σε έναν διακομιστή. Οι χρήστες μπορούν να χρησιμοποιήσουν το Redux μαζί με τη React ή με οποιαδήποτε άλλη βιβλιοθήκη προβολής. Το Redux λειτουργεί με οποιοδήποτε επίπεδο διεπαφής χρήστη και διαθέτει ένα μεγάλο οικοσύστημα πρόσθετων που ταιριάζουν στις ανάγκες των χρηστών.

Τα θετικά του Redux είναι τα εξής:

1. Καθιερωμένη δομή δεδομένων: Με το Redux, η κατάσταση της εφαρμογής αποθηκεύεται σε ένα δέντρο αντικειμένων σε έναν καθολικό καταχωρητή κατάστασης, που είναι εύκολο να αναπαρασταθεί και να κατανοηθεί.
2. Εύκολη διαχείριση κατάστασης: Το Redux καθιστά εύκολο την αλληλεπίδραση με την κατάσταση της εφαρμογής, είτε με την ενημέρωση της κατάστασης είτε με την ανάκτηση της κατάστασης.
3. Ευκολία συντήρησης: Η δομή του Redux καθιστά εύκολη τη συντήρηση της εφαρμογής. Η κατάσταση της εφαρμογής αποθηκεύεται σε ένα μόνο σημείο, κάνοντας την εφαρμογή πιο αξιόπιστη και πιο εύκολη στη διαχείριση.
4. Επαναχρησιμότητα κώδικα: Το Redux καθιστά εύκολο να επαναχρησιμοποιήσει ο κώδικας σε διάφορα σημεία της εφαρμογής, καθιστώντας τον κώδικα πιο αποτελεσματικό.

Κάποια από τα αρνητικά του Redux είναι τα εξής:

1. Υπάρχει μια εκμάθηση που απαιτείται για να κατανοηθεί πλήρως ο τρόπο λειτουργίας του Redux και πώς να το εφαρμοστεί σε ένα project.
2. Αν δεν υπάρχει μια καλή αρχιτεκτονική, μπορεί να καταλήξει σε περιττή πολυπλοκότητα στον κώδικα, καθώς και σε μεγαλύτερη ποσότητα κώδικα που πρέπει να γραφτεί.
3. Η εισαγωγή του Redux σε ένα μικρό project μπορεί να είναι υπερβολική, καθώς η πολυπλοκότητα του Redux μπορεί να είναι περιττή σε ένα απλό project.
4. Επίσης, μπορεί να προκύψει μια μικρή καθυστέρηση στην ενημέρωση του state στο Redux σε σχέση με άλλα state management libraries, όπως το MobX.

1.10 Τι είναι CSS: Cascading Style Sheets;

Εικόνα 9. Logo CSS

Το Cascading Style Sheets (CSS) είναι μια γλώσσα φύλλου στυλ που χρησιμοποιείται για τον έλεγχο της παρουσίασης των ιστότοπων.

Το CSS παρουσιάστηκε για πρώτη φορά από τον Håkon Wium Lie το 1994 ενώ εργαζόταν στον Ευρωπαϊκό Οργανισμό Πυρηνικών Ερευνών (CERN) μαζί με τον Tim Berners-Lee, τον εφευρέτη του Παγκόσμιου Ιστού. Εκείνη την εποχή, οι ιστοσελίδες δημιουργούνταν συνήθως αποκλειστικά με HTML, τη γλώσσα σήμανσης υπερκειμένου που είχε αναπτύξει ο Berners-Lee τη δεκαετία του 1990.

Ωστόσο, η HTML είχε αναπτυχθεί για να περιγράψει τη σημασιολογία των στοιχείων ενός εγγράφου web (όπως οι επικεφαλίδες και οι παράγραφοί του) αντί να παρέχει οδηγίες στυλ. Καθώς η αυξανόμενη χρήση της HTML για το στυλ ιστοσελίδων έγινε όλο και πιο δυσκίνητη, το CSS εισήχθη για να παρέχει μια πιο αποτελεσματική μέθοδο για το στυλ ιστότοπων σε συνδυασμό με HTML. Σήμερα, μαζί με την HTML και την JavaScript, η CSS είναι μία από τις βασικές τεχνολογίες που υποστηρίζουν τον Παγκόσμιο Ιστό (Erin Glass, 2020).

Η CSS είναι μια γλώσσα φύλλου στυλ που χρησιμοποιείται για να περιγράψει την παρουσίαση ενός εγγράφου γραμμένου σε HTML ή XML (συμπεριλαμβανομένων διαλέκτων XML όπως SVG, MathML ή XHTML). Το CSS περιγράφει τον τρόπο με τον οποίο τα στοιχεία πρέπει να αποδίδονται στην οθόνη, σε χαρτί, σε ομιλία ή σε άλλα μέσα. Το CSS είναι από τις βασικές γλώσσες του ανοιχτού ιστού και τυποποιείται σε όλα τα προγράμματα περιήγησης στο Web σύμφωνα με τις προδιαγραφές του W3C. Προηγουμένως, η ανάπτυξη διαφόρων τμημάτων της προδιαγραφής CSS έγινε ταυτόχρονα, γεγονός που επέτρεψε την έκδοση των πιο πρόσφατων συστάσεων.

Οι απαιτήσεις των φυλλομετρητών βοήθησε την CSS στην απότομη βελτίωση των κανόνων της με τέτοιο τρόπο που οι προγραμματιστές να είναι σε θέση να δημιουργήσουν περίπλοκα σχέδια για τις εφαρμογές. Όπως και η HTML, η CSS δεν είναι γλώσσα προγραμματισμού. Δεν είναι ούτε γλώσσα σήμανσης. Η CSS είναι μια γλώσσα φύλλου στυλ. Το CSS είναι αυτό που χρησιμοποιείτε για να διαμορφώσετε επιλεκτικά στοιχεία HTML. Για παράδειγμα, αυτό το CSS επιλέγει κείμενο παραγράφου, ορίζοντας το χρώμα σε μαύρο και με κόκκινο περίγραμμα:

```
p {
  color: black;
  border: 1px solid black;
}
```

Παράδειγμα 2ο (Στο σώμα του html κώδικα στο body θα γίνει background μαύρο και θα οι διαστάσεις των λεκτικών στα 16px) :

```
body{
   background: #000;
   font-size: 16px;
```

}

Μερικά θετικά του CSS είναι τα εξής:

1. Χωρισμός του περιεχομένου από τη μορφή: Με το CSS, μπορείτε να χωρίσετε το περιεχόμενο της ιστοσελίδας σας από την εμφάνιση της. Αυτό σημαίνει ότι μπορείτε να αλλάξετε την εμφάνιση της ιστοσελίδας σας χωρίς να αλλάξετε τη δομή του περιεχομένου.
2. Διαχείριση στυλ: Με το CSS, μπορείτε εύκολα να διαχειριστείτε τα στυλ στην ιστοσελίδα σας, όπως το χρώμα, τα περιθώρια, το μέγεθος της γραμματοσειράς και πολλά άλλα. Αυτό σας επιτρέπει να δημιουργείτε μια ομοιόμορφη και συνεπή εμφάνιση σε όλη την ιστοσελίδα.
3. Εξοικονόμηση χρόνου: Το CSS μπορεί να σας βοηθήσει να εξοικονομήσετε χρόνο στον σχεδιασμό της ιστοσελίδας σας. Μπορείτε να χρησιμοποιήσετε κλάσεις και id για να εφαρμόζετε τα ίδια στυλ σε διαφορετικά τμήματα της ιστοσελίδας σας, αντί να επαναλαμβάνετε τον κώδικα στυλ για κάθε στοιχείο ξεχωριστά.

Μερικά από τα αρνητικά του CSS είναι:

1. Δυσκολία στη διαχείριση: Καθώς ένα έγγραφο HTML μπορεί να περιλαμβάνει πολλά αρχεία CSS, η διαχείριση και η συντήρηση μπορεί να γίνει δύσκολη, ειδικά για μεγάλα έγγραφα.
2. Πρόβλημα συμβατότητας: Υπάρχουν διάφοροι φυλλομετρητές που διαφέρουν στην υποστήριξή τους για το CSS, οπότε η διασφάλιση της συμβατότητας μεταξύ διαφορετικών φυλλομετρητών μπορεί να είναι πρόβλημα.
3. Περιορισμένη δυνατότητα επεξεργασίας: Ενώ το CSS μπορεί να είναι πολύ δυνατό, δεν μπορεί να αντικαταστήσει τις γλώσσες προγραμματισμού όταν πρόκειται για πιο προηγμένες λειτουργίες, όπως η αλληλεπίδραση του χρήστη και η διαχείριση δεδομένων.
4. Πολυπλοκότητα: Καθώς το CSS μπορεί να χρησιμοποιηθεί για να σχεδιαστεί σχεδόν τα πάντα σε μια ιστοσελίδα, μπορεί να γίνει πολύπλοκο και δύσκολο να κατανοηθεί και να συντηρηθεί από τους προγραμματιστές και σχεδιαστές.

1.11 Τι είναι το Responsive CSS;

Το Responsive web design (RWD) είναι μια προσέγγιση σχεδιασμού ιστοσελίδων για να κάνει τις ιστοσελίδες να αποδίδουν καλά σε όλα τα μεγέθη και τις αναλύσεις οθόνης, εξασφαλίζοντας παράλληλα καλή χρηστικότητα. Είναι ο τρόπος σχεδιασμού για έναν ιστό πολλαπλών συσκευών (MDN contributors, 2023).

Η HTML είναι βασικά ανταποκρινόμενη ή ρευστή. Εάν δημιουργήσετε μια ιστοσελίδα που περιέχει μόνο HTML, χωρίς CSS, και αλλάξετε το μέγεθος του παραθύρου, το πρόγραμμα περιήγησης θα αναδιαμορφώσει αυτόματα το κείμενο ώστε να ταιριάζει στη θύρα προβολής.

Ενώ η προεπιλεγμένη συμπεριφορά απόκρισης μπορεί να ακούγεται σαν να μην απαιτείται λύση, οι μεγάλες γραμμές κειμένου που εμφανίζονται σε πλήρη οθόνη σε μια ευρεία οθόνη μπορεί να είναι δυσανάγνωστες. Εάν μειωθεί το μεγάλο μήκος γραμμής οθόνης με CSS, όπως με τη δημιουργία στηλών ή την προσθήκη σημαντικής αναπλήρωσης, ο ιστότοπος μπορεί να φαίνεται στριμωγμένος για τον χρήστη που περιορίζει το παράθυρο του προγράμματος περιήγησής του ή ανοίγει τον ιστότοπο σε κινητή συσκευή.

Το Responsive CSS είναι μια τεχνική σχεδίασης ιστοσελίδων που βασίζεται στη χρήση CSS (Cascading Style Sheets) και επιτρέπει στην ιστοσελίδα να προσαρμόζεται αυτόματα σε διαφορετικές συσκευές και μεγέθη οθονών, όπως κινητά τηλέφωνα, tablet ή υπολογιστές.

Με το responsive CSS, οι σχεδιαστές μπορούν να δημιουργήσουν μια ιστοσελίδα που αναδιαμορφώνεται αυτόματα ώστε να φαίνεται καλά και να είναι λειτουργική σε όλες τις συσκευές. Η τεχνική αυτή συνήθως χρησιμοποιείται μέσω media queries, που ανιχνεύουν το μέγεθος της οθόνης της συσκευής και παρέχουν κατάλληλα στυλ CSS για την καθεμία από τις διαφορετικές κατηγορίες μεγέθους οθονών.

Με το Responsive CSS, οι ιστοσελίδες είναι πιο ευέλικτες και εξυπηρετούν καλύτερα τους χρήστες που χρησιμοποιούν διαφορετικούς τύπους συσκευών για να περιηγηθούν στο διαδίκτυο.

Η τεχνική αυτή χρησιμοποιεί το CSS (Cascading Style Sheets) για να ελέγχει την εμφάνιση της ιστοσελίδας σε διαφορετικά μεγέθη οθονών και να προσαρμόζει δυναμικά το περιεχόμενο και τη διάταξη της ιστοσελίδας ανάλογα με το μέγεθος και την ανάλυση της συσκευής που χρησιμοποιείται.

Οι τεχνικές του responsive CSS συνήθως περιλαμβάνουν τη χρήση media queries, που επιτρέπουν στον προγραμματιστή να καθορίσει διαφορετικούς κανόνες CSS για διαφορετικά μεγέθη οθονών και τη χρήση σχετικών μονάδων μέτρησης, όπως το ποσοστό ή το em, αντί για απόλυτα μεγέθη, ώστε οι διαστάσεις και οι αναλογίες των στοιχείων της ιστοσελίδας να προσαρμόζονται ανάλογα με το μέγεθος της οθόνης.

Ένα απλό παράδειγμα από το responsive CSS είναι η προσαρμογή της διάταξης και του μεγέθους του κειμένου σε μια ιστοσελίδα ανάλογα με το μέγεθος της οθόνης.

Παρακάτω παρουσιάζεται ένα παράδειγμα κώδικα HTML και CSS που χρησιμοποιεί την τεχνική του responsive CSS για να προσαρμόσει το μέγεθος του κειμένου σε μια ιστοσελίδα:

html

Copy code

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Responsive CSS Example</title>
    <style>
        body {
            font-size: 16px;
            line-height: 1.5;
        }
        @media (max-width: 768px) {
            body {
                font-size: 14px;
            }
        }
        @media (max-width: 480px) {
            body {
                font-size: 12px;
            }
        }
    </style>
</head>
<body>
    < div class=”container”>
        <h1>Κείμενο στην ιστοσελίδα</h1>
        <p>Αυτό είναι ένα παράδειγμα κειμένου στην ιστοσελίδα.</p>
</div>
```

</body>

</html>

Στον παραπάνω κώδικα, ορίζεται το μέγεθος της γραμματοσειράς του κειμένου στην κλάση .container της ιστοσελίδας σε 16 pixels. Στη συνέχεια, χρησιμοποιείται media queries για να ορίστούν διαφορετικά μεγέθη γραμματοσειράς για διαφορετικά μεγέθη οθονών. Στην περίπτωση που το πλάτος της οθόνης είναι μικρότερο από 768 pixels, ορίζουμε το μέγεθος της γραμματοσειράς σε 14 pixels και στην περίπτωση που το πλάτος της σελίδας είναι 480 pixels το μέγεθος της γραμματοσείρας σε 12 pixels

Τα θετικά του responsive CSS είναι πολλά και σημαντικά. Ανάμεσα σε αυτά είναι τα εξής:

1. Βελτιωμένη εμπειρία χρήστη: Οι ιστοσελίδες που χρησιμοποιούν responsive CSS προσαρμόζονται στο μέγεθος και τις δυνατότητες της συσκευής που χρησιμοποιεί ο χρήστης, προσφέροντας μια βελτιωμένη εμπειρία χρήσης.
2. Μείωση του χρόνου φόρτωσης: Με το responsive CSS, η ιστοσελίδα φορτώνεται μόνο με το απαραίτητο περιεχόμενο ανάλογα με τη συσκευή του χρήστη, μειώνοντας τον χρόνο φόρτωσης και εξοικονομώντας δεδομένα.

3. Απλοποίηση της συντήρησης: Αντί να αναπτύσσουμε διαφορετικές ιστοσελίδες για διαφορετικές συσκευές, με το responsive CSS δημιουργούμε μια ιστοσελίδα που προσαρμόζεται αυτόματα σε διάφορες συσκευές, απλοποιώντας έτσι τη συντήρηση της ιστοσελίδας.
4. Βελτιωμένος SEO: Οι ιστοσελίδες που χρησιμοποιούν responsive CSS έχουν μια μόνο URL και μια μόνο HTML

Αν και το responsive CSS έχει πολλά θετικά, υπάρχουν και κάποια αρνητικά που πρέπει να ληφθούν υπόψη:

1. Μεγαλύτερο μέγεθος αρχείων: Οι ιστοσελίδες που χρησιμοποιούν responsive CSS συνήθως έχουν μεγαλύτερα αρχεία CSS, καθώς πρέπει να υποστηρίξουν διαφορετικά μεγέθη οθονών και συσκευών. Αυτό μπορεί να οδηγήσει σε αργότερους χρόνους φόρτωσης.
2. Πιο περίπλοκος κώδικας: Ο σχεδιασμός μιας ιστοσελίδας με responsive CSS μπορεί να απαιτεί πιο περίπλοκο κώδικα, καθώς πρέπει να ληφθούν υπόψη διάφοροι παράγοντες όπως το μέγεθος της οθόνης και ο περιορισμένος χώρος στις μικρότερες οθόνες.
3. Δυσκολία στον σχεδιασμό: Στον σχεδιασμό μιας ιστοσελίδας με responsive CSS, πρέπει να ληφθούν υπόψη διάφορες παράμετροι, όπως η διάταξη των στοιχείων και ο τρόπος προβολής τους σε διάφορες συσκευές, και αυτό μπορεί να καθιστά τον σχεδιασμό πιο δύσκολο και περίπλοκο.

1.12 Τι είναι Html;

Εικόνα 10. Logo Html (Anon., χ.χ.)

Το HTML είναι το πιο βασικό δομικό στοιχείο του Ιστού. Ορίζει την έννοια και τη δομή του περιεχομένου ιστού. Άλλες τεχνολογίες εκτός από την HTML χρησιμοποιούνται γενικά για την περιγραφή της εμφάνισης / παρουσίασης μιας ιστοσελίδας (CSS) ή της λειτουργικότητας / συμπεριφοράς (JavaScript) (MDN contributors, 2023).

Το HTML (Hypertext Markup Language) είναι μια γλώσσα σήμανσης που χρησιμοποιείται για τη δημιουργία και τον σχεδιασμό ιστοσελίδων στον παγκόσμιο ιστό (World Wide Web). Το HTML δίνει τη δυνατότητα στους δημιουργούς ιστοσελίδων να δομήσουν την πληροφορία και τα περιεχόμενα της ιστοσελίδας, καθώς και να τοποθετήσουν πλαίσια, εικόνες, κείμενο, φόρμες και άλλα στοιχεία στη σελίδα.

Μερικά από τα θετικά του HTML είναι:

1. Είναι εύκολο στην εκμάθηση και στη χρήση, καθώς η σύνταξη της είναι απλή και κατανοητή.
2. Επιτρέπει στους προγραμματιστές να δημιουργήσουν διάφορα είδη περιεχομένου, όπως κείμενο, εικόνες, βίντεο και ήχο.
3. Είναι ευέλικτο, επιτρέποντας στους προγραμματιστές να προσαρμόζουν την εμφάνιση της σε διάφορες συσκευές και πλατφόρμες.
4. Επιτρέπει τη δημιουργία διαδραστικών στοιχείων στη σελίδα, όπως φόρμες και κουμπιά.
5. Είναι συμβατό με διάφορους τύπους προγραμματιστικών γλωσσών και επιτρέπει την ενσωμάτωση διαφόρων εφέ και δυνατοτήτων στη σελίδα.

   Αυτά τα θετικά καθιστούν την HTML ένα απαραίτητο εργαλείο για τη δημιουργία ιστοσελίδων και την παρουσίαση περιεχομένου στο διαδίκτυο.

Κάποια από τα αρνητικά του HTML είναι:

1. Περιορισμένη ευελιξία: Το HTML είναι σχεδιασμένο να δημιουργεί στατικά περιεχόμενα και δεν είναι πολύ εύκολο να δημιουργηθούν δυναμικά περιεχόμενα μόνο με το HTML.

2. Περιορισμένες δυνατότητες διάταξης: Το HTML παρέχει μόνο βασικές εντολές για τη διάταξη στοιχείων στη σελίδα, όπως το στοίχισμα και η ευθυγράμμιση, και όχι πολύπλοκες λειτουργίες διάταξης.
3. Περιορισμένα χαρακτηριστικά ασφάλειας: Το HTML δεν παρέχει πολλά χαρακτηριστικά ασφάλειας και μπορεί να είναι επιρρεπές σε επιθέσεις όπως το cross-site scripting (XSS).
4. Απαιτείται συχνή ενημέρωση: Λόγω της συνεχούς εξέλιξής του, το HTML απαιτεί συνεχή ενημέρωση για να παραμένει αποτελεσματικό και ασφαλές.

1.13 Τι είναι η Canva;

Εικόνα 11. Logo Canva

Η Canva είναι μια διαδικτυακή πλατφόρμα σχεδίασης γραφικών που παρέχει μια εύχρηστη διεπαφή για τη δημιουργία επαγγελματικών σχεδίων, ακόμα και αν δεν υπάρχει προηγούμενη εμπειρία στον σχεδιασμό.

Η Canva διαθέτει χιλιάδες δωρεάν, επαγγελματικά σχεδιασμένα πρότυπα που μπορούν να προσαρμοστούν με λίγα μόνο κλικ. Απλώς ο χρήστης μπορεί να ανεβάσει τις φωτογραφίες του στο Canva, να επιλέξει ένα πρότυπο της επιλογής του και να το αποθηκεύσει το αρχείο στον υπολογιστή του (businessinsider, 2020).

Η Canva είναι επίσης γνωστή για τη δυνατότητά της να επεξεργάζεται φωτογραφίες. Οι χρήστες μπορούν να επεξεργαστούν, να προσθέσουν φίλτρα και εφέ, να αλλάξουν το μέγεθος και την κατεύθυνση των εικόνων, να δημιουργήσουν κολάζ, και να κάνουν πολλά άλλα.

Η Canva έχει πολλά θετικά χαρακτηριστικά και πλεονεκτήματα, μερικά από αυτά είναι:

5. Εύκολη χρήση: Η Canva έχει μια εύχρηστη εφαρμογή που επιτρέπει στους χρήστες να δημιουργούν επαγγελματικά σχέδια χωρίς προηγούμενη εμπειρία στον σχεδιασμό.
6. Μεγάλη συλλογή γραφικών: Η Canva παρέχει μια μεγάλη συλλογή από δωρεάν και πληρωμένα γραφικά, εικόνες, εικονίδια, κείμενα και φοντίδες που μπορούν να χρησιμοποιηθούν στα σχέδια.

7. Δυνατότητα επεξεργασίας φωτογραφιών: Οι χρήστες μπορούν να επεξεργαστούν και να προσαρμόσουν εικόνες με διάφορους τρόπους, όπως προσθήκη φίλτρων, αλλαγή μεγέθους και κατεύθυνσης, δημιουργία κολάζ κτλπ.
8. Προσαρμογή σχεδίων: Οι χρήστες μπορούν να προσαρμόσουν τα σχέδιά τους με τη χρήση διάφορων επιλογών στυλ, χρωμάτων, γραμματοσειρών και εικονιδίων.

Μερικά από τα αρνητικά της Canva είναι:

9. Περιορισμένη προσαρμοστικότητα: Αν και η Canva παρέχει μια μεγάλη ποικιλία από στυλ, γραμματοσειρές και γραφικά, η προσαρμοστικότητά της είναι περιορισμένη σε σύγκριση με προηγμένα εργαλεία σχεδίασης.
10. Περιορισμένη επεξεργασία εικόνων: Αν και η Canva παρέχει κάποια βασικά εργαλεία επεξεργασίας εικόνων, δεν είναι τόσο προηγμένη όσο άλλα προηγμένα εργαλεία επεξεργασίας εικόνων.
11. Υψηλό κόστος για προηγμένες λειτουργίες: Οι προηγμένες λειτουργίες της Canva είναι διαθέσιμες μόνο με τη συνδρομή στο Canva Pro, η οποία είναι αρκετά ακριβή συγκριτικά με άλλες επιλογές εργαλείων σχεδίασης.
12. Εξάρτηση από σύνδεση στο διαδίκτυο: Η Canva είναι μια πλατφόρμα βασισμένη στον ιστό, οπότε οι χρήστες χρειάζονται πρόσβαση στο διαδίκτυο για να τη χρησιμοποιήσουν, κάτι που μπορεί να είναι ανασταλτικό για ορισμένους χρήστες.

1.14 Τι είναι το Visual Studio code (VScode);

Εικόνα 12. Logo Visual Studio code

Το Visual Studio Code (ή απλώς VS Code) είναι ένας δωρεάν επεξεργαστής κειμένου, που αναπτύσσεται από την Microsoft και χρησιμοποιείται κυρίως για την ανάπτυξη λογισμικού. Παρέχει πλούσια λειτουργικότητα και δυνατότητες για τη γρήγορη και εύκολη ανάπτυξη κώδικα, όπως σύνταξη και υπογράμμιση κώδικα, αυτόματη συμπλήρωση κώδικα, αναζήτηση και αντικατάσταση, ενσωμάτωση με λειτουργίες απομακρυσμένης ανάπτυξης.

Το VS Code είναι επίσης εξαιρετικά προσαρμόσιμο, καθώς παρέχει πολλά πακέτα επέκτασης και πρόσθετα που επιτρέπουν στους χρήστες να προσαρμόσουν το περιβάλλον εργασίας τους στις ανάγκες τους και να προσθέσουν λειτουργικότητα που δεν είναι διαθέσιμη στο προεπιλεγμένο πακέτο. Είναι επίσης διαθέσιμο σε πολλά λειτουργικά συστήματα, όπως Windows, Linux και macOS.

Εκτός από την όλη ιδέα του να είναι ελαφρύ και να ξεκινά γρήγορα, το Visual Studio Code έχει συμπλήρωση κώδικα IntelliSense για μεταβλητές, μεθόδους και εισαγόμενες ενότητες. Γραφικός εντοπισμός σφαλμάτων. Linting, επεξεργασία πολλαπλών δρομέα, υποδείξεις παραμέτρων και άλλες ισχυρές δυνατότητες επεξεργασίας. Κομψή πλοήγηση κώδικα και επανασχεδιασμός. Τέλος, και ενσωματωμένο έλεγχο πηγαίου κώδικα, συμπεριλαμβανομένης της υποστήριξης Git. Πολλά από αυτά προσαρμόστηκαν από την τεχνολογία Visual Studio (Martin Heller, 2022).

Το Visual Studio Code (VS Code) έχει πολλά χαρακτηριστικά που καθιστούν τη χρήση του πολύ γρήγορη και ευκολότερη από άλλα εργαλεία ανάπτυξης λογισμικού. Μερικά παραδείγματα είναι:

1. Αυτόματη συμπλήρωση κώδικα: Το VS Code παρέχει ένα ισχυρό σύστημα αυτόματης συμπλήρωσης κώδικα, που μπορεί να επιταχύνει την πληκτρολόγηση και να μειώσει τον χρόνο ανάπτυξης κώδικα.
2. Στιγμιότυπα αρχείων: Το VS Code μπορεί να δημιουργήσει αυτόματα στιγμιότυπα αρχείων κατά τη διάρκεια της ανάπτυξης λογισμικού, που μπορούν να επιταχύνουν την αναζήτηση και την πλοήγηση στα αρχεία.
3. Ενσωμάτωση με Git: Το VS Code έχει ενσωματωμένη υποστήριξη για το Git, το οποίο μπορεί να επιταχύνει τη διαδικασία της ανάπτυξης λογισμικού σε ένα συνεργατικό περιβάλλον.

Μερικά θετικά στοιχεία του Visual Studio Code (VSCode) είναι τα εξής:

1. Είναι δωρεάν και ανοιχτού κώδικα, διαθέσιμο για όλους τους χρήστες.
2. Παρέχει ένα ευρύ φάσμα επεκτάσεων και πακέτων για τη βελτίωση της απόδοσης και της λειτουργικότητας του.
3. Διαθέτει πλούσιο σύστημα αυτόματης συμπλήρωσης κώδικα, αντικατοπτρίζοντας τις βέλτιστες πρακτικές στον τομέα του προγραμματισμού.
4. Παρέχει τη δυνατότητα σύγκρισης διαφορετικών εκδόσεων κώδικα και συγχώνευσης των αλλαγών μεταξύ τους.
5. Είναι διαθέσιμο σε πολλά λειτουργικά συστήματα, συμπεριλαμβανομένων των Windows, macOS και Linux.
6. Παρέχει τη δυνατότητα αποσφαλμάτωσης κώδικα σε πολλές γλώσσες προγραμματισμού.
7. Παρέχει τη δυνατότητα ενσωμάτωσης της εφαρμογής στις υπηρεσίες της Microsoft, όπως ο Azure και η Visual Studio Team Services.
8. Έχει μια μεγάλη και ενεργή κοινότητα χρηστών, με πλούσιο υλικό υποστήριξης και επίλυσης προβλημάτων.

Μερικά αρνητικά στοιχεία του Visual Studio Code είναι:

1. Κατανάλωση πόρων: Το Visual Studio Code μπορεί να καταναλώνει περισσότερους πόρους από άλλα ελαφριά περιβάλλοντα ανάπτυξης κώδικα.
2. Αργή εκκίνηση: Το Visual Studio Code μπορεί να χρειαστεί λίγο περισσότερο χρόνο για να εκκινήσει από ό, τι άλλα ελαφριά περιβάλλοντα ανάπτυξης κώδικα.
3. Πολυπλοκότητα: Ενώ οι επιλογές που παρέχει το Visual Studio Code είναι χρήσιμες, η πληθώρα τους μπορεί να καθιστά το πρόγραμμα λίγο πιο περίπλοκο στη χρήση.
4. Περιορισμένη υποστήριξη για μεγάλα έργα: Το Visual Studio Code δεν είναι τόσο καλό στη διαχείριση μεγάλων έργων όπως άλλα περιβάλλοντα ανάπτυξης κώδικα.
5. Ανεπαρκής υποστήριξη για το debugging: Το Visual Studio Code δεν παρέχει την ίδια λειτουργικότητα debugging με άλλα περιβάλλοντα ανάπτυξης κώδικα, όπως το Visual Studio της Microsoft.

1.15 Τι είναι τα APIs;

Τα APIs (Application Programming Interfaces) είναι ένα σύνολο κανόνων, προδιαγραφών και διαδικασιών που επιτρέπουν σε δύο ή περισσότερες εφαρμογές να επικοινωνούν μεταξύ τους και να ανταλλάσσουν δεδομένα. Συνήθως χρησιμοποιούνται για την επικοινωνία μεταξύ του front-end (ο χρήστης) και του back-end (ο server) μιας εφαρμογής, αλλά μπορούν να χρησιμοποιηθούν και για την επικοινωνία μεταξύ δύο διαφορετικών εφαρμογών.

Ένα API είναι μια διεπαφή που χρησιμοποιούν οι προγραμματιστές λογισμικού για να αλληλεπιδρούν μέσω προγραμματισμού με στοιχεία λογισμικού ή πόρους εκτός του δικού τους κώδικα. Ένας ακόμη απλούστερος ορισμός είναι ότι ένα API είναι το μέρος ενός στοιχείου λογισμικού που είναι προσβάσιμο σε άλλα στοιχεία (Matthew Tyson, 2022).

Οι προγραμματιστές μπορούν να χρησιμοποιήσουν τα APIs για να αναζητήσουν ή να ανακτήσουν δεδομένα από μια άλλη εφαρμογή ή για να καλέσουν μια συγκεκριμένη λειτουργία που παρέχεται από μια άλλη εφαρμογή. Τα APIs μπορούν να επιστρέψουν δεδομένα σε διαφορετικά μορφότυπα, όπως JSON, XML, κείμενο κ.λπ., ανάλογα με την απαίτηση της εφαρμογής που τα χρησιμοποιεί.

Οι APIs (Application Programming Interfaces) έχουν πολλά θετικά στοιχεία και είναι απαραίτητα για τη σύγχρονη ανάπτυξη λογισμικού. Ανάμεσα στα κύρια θετικά στοιχεία των APIs περιλαμβάνονται:

1. Διευκόλυνση της επικοινωνίας μεταξύ διαφορετικών συστημάτων: Τα APIs επιτρέπουν σε διαφορετικά συστήματα και εφαρμογές να επικοινωνούν μεταξύ τους και να ανταλλάσσουν δεδομένα, χωρίς να χρειάζεται να γνωρίζουν τις λεπτομέρειες υλοποίησης του κάθε συστήματος.
2. Εξοικονόμηση χρόνου και κόστους στην ανάπτυξη λογισμικού: Χρησιμοποιώντας ένα API μπορείτε να αποφύγετε την ανάγκη να αναπτύξετε μια ολόκληρη

λειτουργία από την αρχή. Αντί αυτού, μπορείτε να χρησιμοποιήσετε ένα ήδη υπάρχον API που παρέχει τη λειτουργία που χρειάζεστε.
3. Ευκολία χρήσης και πρόσβασης: Οι APIs παρέχουν μια καλά τεκμηριωμένη διεπαφή χρήστη (user interface) που διευκολύνει τη χρήση τους από άλλα συστήματα ή εφαρμογές.

Παρά τα πολλά οφέλη που προσφέρουν τα APIs, υπάρχουν και κάποια αρνητικά στοιχεία που πρέπει να ληφθούν υπόψη:

1. Εξάρτηση από τρίτους: Καθώς οι APIs παρέχονται από τρίτους, οι εφαρμογές που τα χρησιμοποιούν είναι εξαρτημένες από τη λειτουργικότητα και τη διαθεσιμότητα των παρόχων των APIs. Αυτό μπορεί να οδηγήσει σε προβλήματα εάν ο πάροχος του API δεν είναι διαθέσιμος ή αν ο πάροχος αλλάξει τη λειτουργικότητα ή την πολιτική του χωρίς προειδοποίηση.
2. Ασφάλεια: Εάν δεν υλοποιηθούν σωστά, τα APIs μπορεί να επιτρέψουν σε εξωτερικούς χρήστες να έχουν πρόσβαση σε ευαίσθητα δεδομένα ή να εκτελέσουν επιθέσεις. Οι παρόχοι των APIs πρέπει να λαμβάνουν κατάλληλα μέτρα ασφαλείας, όπως την χρήση του πρωτοκόλλου HTTPS, την ταυτοποίηση και τον έλεγχο πρόσβασης.

1.16 Τι είναι το Compressor;

Το Compressor είναι ένα online εργαλείο συμπίεσης εικόνων. Με το Compressor, μπορείτε να μειώσετε το μέγεθος των εικόνων σας χωρίς να χάσετε ποιότητα. Αυτό είναι χρήσιμο για τη βελτίωση της ταχύτητας φόρτωσης της ιστοσελίδας σας, καθώς μικρότερο μέγεθος εικόνων σημαίνει μικρότερο χρόνο φόρτωσης.

Εκτός από το JPEG, λειτουργεί επίσης με αρχεία PNG, GIF και SVG. Υπάρχει επίσης ένα τακτοποιημένο παράθυρο προεπισκόπησης στο οποίο υπάρχει η δυνατότητα να σύρει ο χρήστης εμπρός και πίσω για να δει το αρχικό και το νέο αρχείο, μόνο για να ελέγξει την απώλεια ποιότητας (Mihir Patkar, 2014).

Το Compressor είναι δωρεάν και δεν απαιτεί καταχώρηση ή λήψη εγκατάστασης εφαρμογής, καθιστώντας το εύκολο στη χρήση από οποιονδήποτε υπολογιστή με σύνδεση στο διαδίκτυο.

Οι θετικές πτυχές της συμπίεσης των φωτογραφιών για μια ιστοσελίδα είναι οι εξής:

1. Βελτίωση της ταχύτητας φόρτωσης της ιστοσελίδας: Η συμπίεση των φωτογραφιών μειώνει το μέγεθος τους και έτσι η ιστοσελίδα φορτώνεται πιο γρήγορα, βελτιώνοντας την εμπειρία του χρήστη.
2. Μείωση των χρεώσεων φιλοξενίας: Όσο μεγαλύτερο είναι το μέγεθος των φωτογραφιών, τόσο περισσότερο χώρο απαιτείται για την αποθήκευσή τους στον διακομιστή. Με τη συμπίεση των φωτογραφιών, μειώνεται ο χώρος αποθήκευσης που απαιτείται, με αποτέλεσμα τη μείωση των χρεώσεων φιλοξενίας της ιστοσελίδας.

1.17 Τι είναι η βιβλιοθηκη Axios;

Το Axios είναι μια βιβλιοθήκη JavaScript που χρησιμοποιείται κυρίως για να κάνει αιτήσεις δικτύου σε έναν διακομιστή και να λαμβάνει απαντήσεις. Στο πλαίσιο της ReactJS, το Axios χρησιμοποιείται κυρίως για να αλληλοεπιδρά με API και να φορτώνει δεδομένα από αυτά.

Μία από τις σημαντικές δυνατότητες του Axios είναι η ισόμορφη φύση του, πράγμα που σημαίνει ότι μπορεί να τρέξει στο πρόγραμμα περιήγησης καθώς και σε εφαρμογές Node.js διακομιστή με την ίδια βάση κώδικα (Pratik Das, 2022).

Το Axios χρησιμοποιείται για να κάνει HTTP αιτήσεις, όπως GET, POST, PUT, DELETE κλπ., σε μια εύκολη και απλή σύνταξη. Μπορεί επίσης να διαχειριστεί τις αποκρίσεις HTTP, να διαχειριστεί σφάλματα και να παρέχει μια ποικιλία από χρήσιμες λειτουργίες όπως αυτόματη μετατροπή δεδομένων σε JSON.

Με το Axios, οι προγραμματιστές μπορούν να γράψουν κώδικα που εκτελεί αιτήσεις σε μια API και να λαμβάνει απαντήσεις χωρίς να χρειάζεται να γράψουν έναν ολόκληρο διακομιστή. Αυτό καθιστά την ανάπτυξη εφαρμογών πιο γρήγορη και αποτελεσματική.

Τα θετικά του Axios περιλαμβάνουν:

1. Ευκολία χρήσης: Το Axios είναι πολύ εύκολο στη χρήση και στην ενσωμάτωση σε ένα έργο React. Μπορεί να χρησιμοποιηθεί σε όλα τα επίπεδα της εφαρμογής σας, όπως στην ανάγνωση και την εγγραφή δεδομένων από και προς τον διακομιστή.
2. Συνοχή: Το Axios παρέχει μια συνεπή συμπεριφορά σε όλα τα περιβάλλοντα που χρησιμοποιούνται.

Μερικά αρνητικά χαρακτηριστικά του Axios είναι:

1. Μέγεθος: Το μέγεθος της βιβλιοθήκης Axios μπορεί να είναι μεγαλύτερο σε σχέση με άλλες βιβλιοθήκες HTTP όπως το Fetch API του προγραμματιστικού περιβάλλοντος του προγραμματιστή (Browser). Αυτό μπορεί να οδηγήσει σε αυξημένο χρόνο φόρτωσης στην αρχική φόρτωση της ιστοσελίδας ή της εφαρμογής.
2. Συμβατότητα: Ενώ η Axios λειτουργεί σε πολλά προγραμματιστικά περιβάλλοντα, δεν είναι πάντα συμβατή με παλαιότερα προγραμματιστικά περιβάλλοντα. Αυτό μπορεί να επηρεάσει τη συμβατότητα με παλαιότερες εκδόσεις προγραμματιστικού περιβάλλοντος, και να απαιτεί επιπλέον εργασία για να διασφαλιστεί η συμβατότητα με όλα τα περιβάλλοντα.

1.18 Τι είναι το Can I Use;

Το "Can I Use" παρέχει ενημερωμένους πίνακες υποστήριξης προγράμματος περιήγησης για την υποστήριξη τεχνολογιών ιστού front-end σε προγράμματα περιήγησης ιστού για επιτραπέζιους υπολογιστές και κινητά (Alexis Deveria, -).

Το εργαλείο παρουσιάζει μια λίστα με τα χαρακτηριστικά, τις εκδόσεις των browsers και το ποσοστό των χρηστών που υποστηρίζεται από το κάθε χαρακτηριστικό. Αυτό βοηθά τους developers και τους σχεδιαστές να λάβουν αποφάσεις σχετικά με τη χρήση των χαρακτηριστικών και την υποστήριξή τους από διαφορετικούς browsers.

Κάποια θετικά του να χρησιμοποιεί κανείς το CanIUse είναι:

1. Δυνατότητα να ελέγξει τη συμβατότητα χαρακτηριστικών μεταξύ διαφορετικών browsers: Το CanIUse παρέχει μια ενημερωμένη λίστα με τα χαρακτηριστικά που υποστηρίζονται από τους διάφορους browsers, που μπορεί να βοηθήσει τους developers και τους σχεδιαστές να ελέγξουν τη συμβατότητα χαρακτηριστικών σε διαφορετικά περιβάλλοντα.
2. Βελτιστοποίηση του κώδικα: Χρησιμοποιώντας το CanIUse, οι developers μπορούν να βελτιστοποιήσουν τον κώδικα τους, διασφαλίζοντας ότι τα χαρακτηριστικά που χρησιμοποιούν υποστηρίζονται από τους περισσότερους browsers και δεν θα προκαλέσουν προβλήματα στην εμπειρία των χρηστών.
3. Εξοικονόμηση χρόνου: Το CanIUse μπορεί να βοηθήσει τους developers και τους σχεδιαστές να εξοικονομήσουν χρόνο και πόρους, καθώς δεν χρειάζεται να εκτελούν δοκιμές σε διάφορους browsers για να ελέγξουν τη συμβατότητα των χαρακτηριστικών.

1.19 Πως μπορούν να συνδυαστούν όλες αυτές οι τεχνολογίες;

Όλες οι παραπάνω τεχνολογίες μπορούν να συνδυαστούν για τη δημιουργία μιας πλήρους web εφαρμογής. Για παράδειγμα, μπορεί να χρησιμοποιηθούν το Node.js και το Express.js για να δημιουργηθεί ο server και να συνδεθούν στη βάση δεδομένων MongoDB χρησιμοποιώντας το Mongoose.

Με τη χρήση της React και του Redux, μπορεί να δημιουργηθεί το front-end της εφαρμογής και να διαχειριστεί ο χρήστης την κατάσταση της εφαρμογής του μέσω της διαχείρισης του state του με το Redux. Η CSS μπορεί να χρησιμοποιηθεί για τον σχεδιασμό και την εμφάνιση της ιστοσελίδας του.

Αν θέλει να δοκιμάσει μια API με το Postman, μπορεί να εκτελέσει αιτήσεις HTTP στον server του και να λάβει αποτελέσματα σε μορφή JSON Web Token (JWT) για να αυθεντοκοποιήσει τους χρήστες του.

Όλες αυτές οι τεχνολογίες μπορούν να συνδυαστούν για να δημιουργήσουν μια πλήρως λειτουργική και εύχρηστη web εφαρμογή.

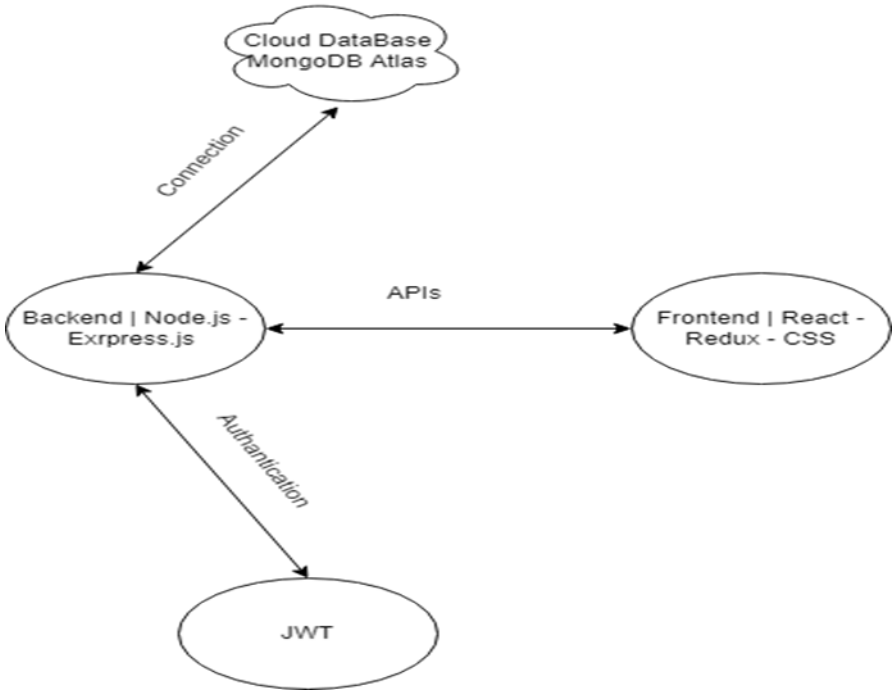
Κεφάλαιο 2ο: Δομή και ανάλυση έργου

Σκοπός του 2ου κεφαλαίου είναι η αναλυτική δομή και οργάνωση της υπάρχουσας πτυχιακής εργασίας.

Η Δομή της εργασίας είναι πολύ σημαντική για το πως θα εξελιχθεί στην πορεία Όποτε η το πρώτο πράγμα που έγινε ήταν η οργάνωση του έργου, τι τεχνολογίες θα χρησιμοποιηθούν πως και γιατί.

Αρχικά το σχεδιάγραμμα του έργου με τον τρόπο που διδάχθηκε στο μάθημα Τεχνολογίες Πολυμέσων.

Στο διάγραμμα θα αποτυπώνεται σχηματικά η δομή, οι λειτουργίες και η αρχιτεκτονική της εφαρμογής. Αυτή η διαδικασία συνήθως πραγματοποιείται πριν την φάση της υλοποίησης. Γιατί με αυτόν τον τρόπο στην πορεία του έργου θα εξοικονομηθεί πολύτιμος χρόνος.



Εικόνα 13. Σχεδιάγραμμα έργου

Όπως φαίνεται και στην εικόνα η δομή είναι η εξής. Το έργο αποτελείται από μία βάση δεδομένων ποιο συγκεκριμένα μία ασύγχρονη βάση την MongoDB. Η βάση αποτελείται από τους πίνακες, όπου εκεί συλλέγονται όλα τα δεδομένα της σε JSON μορφή.

Στην συνέχεια, φαίνεται η εφαρμογή (backend) που χτίστηκε με nodejs και το framework της το express js που βοηθάει για την δημιουργία των APIs και για την σύνδεση με την το frontend. Παρατηρείτε την εφαρμογή να συνδέεται με την βάση δεδομένων που από εκεί θα τραβήξει τα απαραίτητα δεδομένα που χρειάζεται κάθε στιγμή ώστε να τα

σερβίρει. Στην συνέχεια το JWT μία τεχνολογία που βοηθάει στην αυθεντικοποίηση των χρηστών την ώρα της σύνδεσης. Ουσιαστικά η εφαρμογή δέχεται τις πληροφορίες των χρηστών από την βάση δεδομένων και κάνει την αυθεντικοποίηση των χρηστών με την τεχνολογία JWT και εφόσον ο χρήστης δώσει τα σωστά στοιχεία θα μπορεί να συνδεθεί στην εφαρμογή.

Τέλος δημιουργήθηκε ένα γραφικό περιβάλλον για τον χρήστη με την τεχνολογία React μία βιβλιοθήκη της javascript που βοηθάει να συνδεθεί η εφαρμογή με όμορφο τρόπο με το backend με την βοήθεια των APIs και ταυτόχρονα με αρκετά οργανωμένο κώδικα. Γιατί η αρχιτεκτονική MVC βοηθάει να διαχωριστεί η εφαρμογή (backend) από αυτό που φαίνεται στον χρήστη (frontend). Επίσης η χρήση CSS ήταν αναπόσπαστο κομμάτι της σχεδίασης της εφαρμογής γιατί με την χρήση της δημιουργήθηκε μία σελίδα εύκολα διαχειρίσιμη από τον μέσο χρήστη. Ακόμη, έγιναν τα απαραίτητα προθέματα στον κώδικα του css έτσι ώστε να μπορεί να είναι λειτουργική η σχεδίαση της σελίδας σε πολλούς browsers. Τέλος, για την καλύτερη λειτουργία του τις εφαρμογής χρησιμοποιήθηκαν τεχνικές όπως η συμπίεση φωτογραφιών που είναι πολύ χρήσιμο για μία σελίδα.

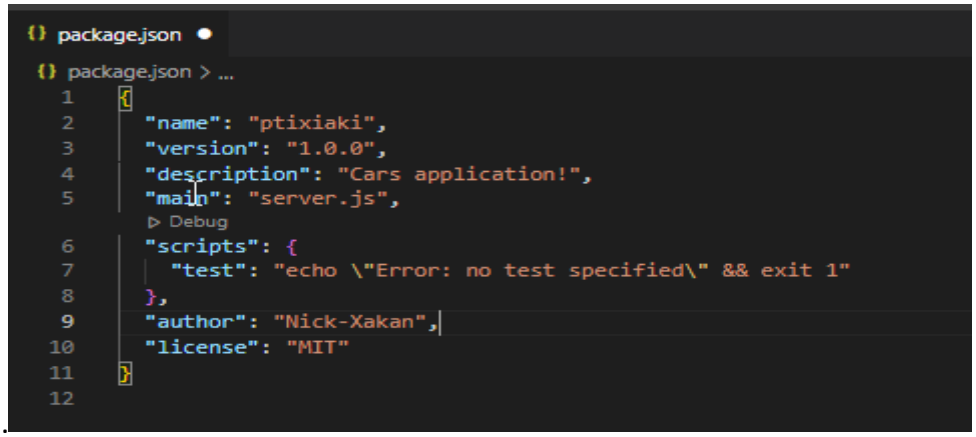
2.1 backend

Δημιουργήθηκε ένας φάκελος με το όνομα backend και μέσα στον φάκελο δημιουργήθηκε να αρχείο με το όνομα server.js. Έπειτα σε έναν τερματικό με την εντολή npm init θα εμφανίσει κάποιες επιλογές όπως την παρακάτω εικόνα. Σε αυτό το σημείο θα πρέπει να συμπληρωθούν τα στοιχεία της εφαρμογής.

```
PS C:\Users\nikos\Desktop\test-pty> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (test-pty)
version: (1.0.0)
description: Cars application!
entry point: (index.js) server.js
test command:
git repository:
keywords:
author: Nick-Xakan
license: (ISC) MIT
About to write to C:\Users\nikos\Desktop\test-pty\package.json:

{
  "name": "test-pty",
  "version": "1.0.0",
  "description": "Cars application!",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Nick-Xakan",
  "license": "MIT"
}


Is this OK? (yes)
```

Εικόνα 14 . στοιχεία έργου 1

Και με αυτόν τον τρόπο θα δημιουργηθεί ένα αρχείο με το όνομα package.json και μέσα θα έχει τα στοιχεία που δόθηκαν για την εφαρμογή

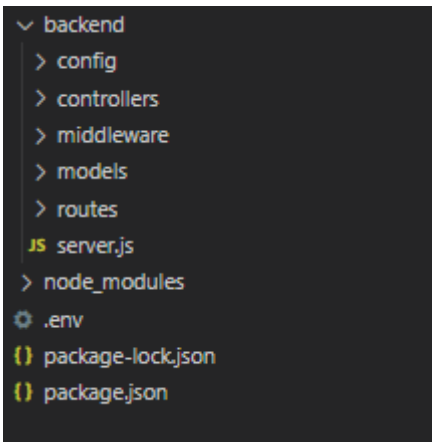


Εικόνα 15 . στοιχεία έργου 2

Στην πορεία μέσα στον φάκελο backend θα δημιουργηθούν ακόμα πέντε φάκελοι με τα εξής ονόματα.

- Routes
- Models
- Middleware
- Controllers
- Config

Ο κάθε ένας έχει τον δικό του ρόλο που θα αναλυθεί στην πορεία. Τα αρχεία και οι φάκελοι θα φαίνονται όπως στην εικόνα παρακάτω.



Εικόνα 16 . φάκελοι backend και αρχεία

Στην συνέχεια εγκατασταθούν όλες τις τεχνολογίες που θα χρησιμοποιηθούν στον φάκελο backend που στην παρούσα περίπτωση ο συγκεκριμένος φάκελος είναι αυτός που θα διαχειρίζεται την εφαρμογή που θα συνδέεται με την βάση και γενικότερα θα αναλαμβάνει την λειτουργικότητα του έργου. Οι τεχνολογίες θα γραφτούν στον τερματικό με τον εξής τρόπο.

npm i express dotenv mongoose colors cors express-async-handler bcryptjs jsonwebtoken

npm i -D nodemon

npm i -D concurrently

Παρακάτω θα γίνει μία επιγραμματική αναφορά των τεχνολογιών και ποια θα είναι η χρήση τους στην εφαρμογή.

- Express : Framework της nodejs που θα βοηθήσει στην δημιουργία των APIs μας.

- Mongoose : Βιβλιοθήκη της mongoBD που θα βοηθήσει για την δημιουργία των σχημάτων και των μοντέλων.
- Colors : Με αυτήν την προσθήκη θα μπορεί ο χρήστης να έχει χρώματα στον τερματικό έτσι ώστε να διευκολύνει στην ανάγνωση του τερματικού.

- Dotenv : Είναι μία λειτουργική μονάδα μηδενικής εξάρτησης που φορτώνει μεταβλητές περιβάλλοντος από ένα αρχείο στο αρχείο .env με αυτόν τον τρόπο μπορέσουν να διαφυλαχτούν σημαντικές πληροφορίες για την εφαρμογή.

- Cors : Η κοινή χρήση πόρων μεταξύ προελεύσεων (CORS) είναι ένας μηχανισμός που βασίζεται σε κεφαλίδα HTTP και επιτρέπει σε ένα 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 οι φυλλομετρητές υποβάλλουν ένα αίτημα "προκαταρκτικού ελέγχου" στον διακομιστή που φιλοξενεί τον πόρο διασταυρούμενης προέλευσης, προκειμένου να ελέγξουν ότι ο διακομιστής θα επιτρέψει το πραγματικό αίτημα. Σε αυτόν τον προκαταρκτικό έλεγχο, το πρόγραμμα περιήγησης στέλνει κεφαλίδες που υποδεικνύουν τη μέθοδο HTTP και κεφαλίδες που θα χρησιμοποιηθούν στην πραγματική αίτηση. Όποτε με την χρήση του cors στον τερματικό μπορούν να δοθούν συγκεκριμένες περιπτώσεις πρόσβασης.

- D nodemon : Είναι ένα εργαλείο που βοηθά στην ανάπτυξη εφαρμογών που βασίζονται στο node.js κάνοντας αυτόματη επανεκκίνηση την εφαρμογή όταν εντοπίζονται αλλαγές αρχείων.

- Express-async-handler : Απλό ενδιάμεσο λογισμικό για τον χειρισμό εξαιρέσεων εντός των διαδρομών async express και τη μετάδοση τους στους ρητούς χειριστές σφαλμάτων σας.

- Bcryptjs : Βελτιστοποιημένο bcrypt σε JavaScript με μηδενικές εξαρτήσεις. Συμβατό με το C++ bcrypt στο node.js και επίσης λειτουργεί στο πρόγραμμα περιήγησης. Και με αυτόν τον τρόπο θα μας βοηθήσει να υπάρξει ασφάλεια από επιθέσεις.

- Jsonwebtoken : Βοηθάει επίσης στην ασφάλεια της εφαρμογή μας λόγο του ότι Το JSON Web Token (JWT) είναι ένα ανοιχτό πρότυπο (RFC 7519) που καθορίζει έναν συμπαγή και αυτόνομο τρόπο για την ασφαλή μετάδοση πληροφοριών μεταξύ των μερών ως αντικείμενο JSON. Αυτές οι πληροφορίες μπορούν να επαληθευτούν και να θεωρηθούν αξιόπιστες επειδή είναι ψηφιακά υπογεγραμμένες. Τα JWTs μπορούν να υπογραφούν χρησιμοποιώντας ένα μυστικό (με τον αλγόριθμο HMAC) ή ένα ζεύγος δημόσιου/ιδιωτικού κλειδιού χρησιμοποιώντας RSA ή ECDSA.

- -D concurrently : Βοηθάει την εφαρμογή με μία εντολή στον τερματικό να τρέξει όλο το πρόγραμμα(δηλαδή να τρέξει ταυτόχρονα και ο φάκελος που εμπεριέχει την εφαρμογή και τον φάκελο που εμπεριέχει το μπροστά μέρος που βλέπει ο χρήστης με μία εντολή.)

Μετά την εγκατάσταση των τεχνολογιών που θα χρησιμοποιηθούν θα δημιουργηθούν τοπικά στον υπολογιστή ο φάκελος node_modules και package-lock.json που εμπεριέχουν ότι χρειάζεται για να αρχίσει να δουλεύει επάνω στην εφαρμογή. Έπειτα στο αρχείο package.json έχουν προσθέσει όλες τις τεχνολογίες όπως παρατηρείται και στην παρακάτω εικόνα.

Εικόνα 17 . τεχνολογίες έργου backend και στοιχεία

Έπειτα στο αρχείο package.json και στο σημείο

“scripts”:{

}

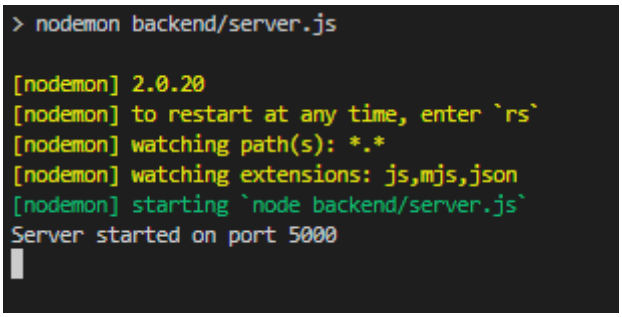
Θα προστεθούν τα εξής.

"start": "node backend/server.js",
"server": "nodemon backend/server.js"

Με αυτόν τον τρόπο στον τερματικό θα μπορούν να τρέξουν την εφαρμογή γράφοντας απλά npm run server και εφόσον προσθέσει το nodemon κάθε φορά που πραγματοποιείτε αποθήκευση του προγράμματος να γίνεται ταυτόχρονη ανανέωση και χρήση των αλλαγών.

Στην συνέχεια γίνεται έλεγχος για το αν ο server θα τρέξει στην πόρτα που θα του έχει οριστεί.

Και εφόσον λειτουργεί ορθά στον τερματικό μόλις γραφτεί npm run server θα πρέπει να εμφανίσει το παρακάτω μήνυμα.



Εικόνα 18. Τρέχει ο Server στην πόρτα 5000

Στην πορεία θα δημιουργηθεί ένας φάκελο .env ώστε εκεί να οριστούν οι μεταβλητές περιβάλλοντος που θα χρησιμοποιηθούν.

Ουσιαστικά το αρχείο server.js θα είναι το αρχείο που θα καταλήγουν όλα έτσι ώστε να τρέχει η εφαρμογή. Εκεί θα καταλήγει η βάση, τα APIs, εκεί γίνεται η σύνδεση με τον φάκελο frontend που θα αναλυθεί στην πορεία και τέλος εκεί τρέχει ο server και σε ποια πόρτα τρέχει.

Έπειτα ο φάκελος routes που έχει δημιουργηθεί μέσα στον φάκελο backend θα εμπεριέχει δύο αρχεία

- carRoutes.js
- userRoutes.js

Αυτά τα αρχεία θα είναι υπεύθυνα για το σύστημα CRUD. Δηλαδή την εμφάνιση, την δημιουργία, την επεξεργασία και την διαγραφή ενός αυτοκίνητου ή ενός χρήστη.

Η ανάλυση τον GET, POST, PUT, DELETE πραγματοποιείται στον φάκελο controllers που εκεί εμπεριέχονται δύο αρχεία

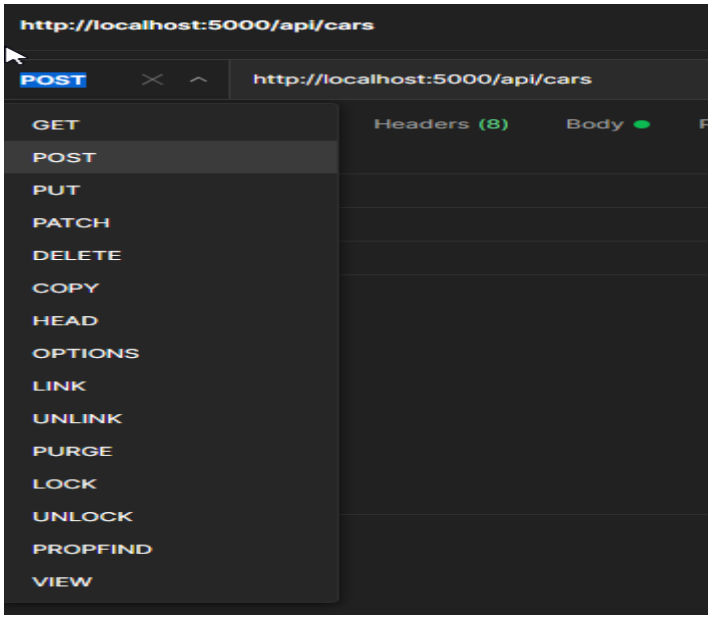
- carController.js
- userController.js

Σε αυτά τα δύο αρχεία γίνεται η ανάλυση των διαδικασιών π.χ. Πότε πρέπει να εμφανίζει σφάλμα (error) η εφαρμογή και κάτω από ποιες συνθήκες να λειτουργεί η κάθε περίπτωση.

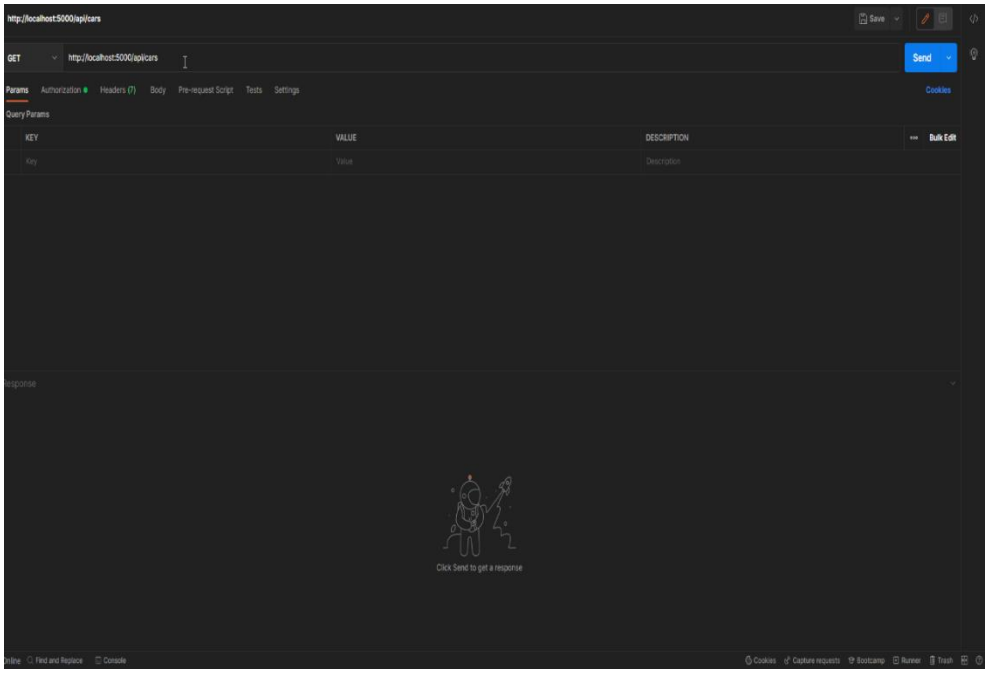
Και η ίδια περίπτωση χρησιμοποιείται και στο αρχείο userControllers.js με την μόνη διαφορά πως σε αυτή την περίπτωση δεν δημιουργείτε η διαδικασία DELETE και η διαδικασία PUT και επιπλέον γίνεται χρήση του bcryptjs και του json web token για την ασφάλεια των δεδομένων των χρηστών που πρόκειται να εγγραφούν στην εφαρμογή.

Ένα εργαλείο που χρειάστηκε για τον έλεγχο των APIs και αν λειτουργούν σωστά ήταν το Postman. Οπότε αρχικά γίνεται εγκατάσταση του postman από το Link Download Postman | Get Started for Free . Στην συνέχεια πραγματοποιείτε εγγραφή στο Postman - Sign Up (getpostman.com) έτσι ώστε να έχει ο χρήστης έναν λογαριασμό. Τέλος εφόσον ολοκληρωθεί η εγκατάσταση της εφαρμογής στον υπολογιστή μπορεί ο χρήστης να συνδεθεί με τον προσωπικό του λογαριασμό και να αρχίσει να εργάζεται επάνω στο postman.

Ανοίγοντας την εφαρμογή στο επάνω μέρος γίνεται ο απαραιτήτως έλεγχος για ποια περίπτωση θέλει να πραγματοποιηθεί.



Εικόνα 19 . Postman POST

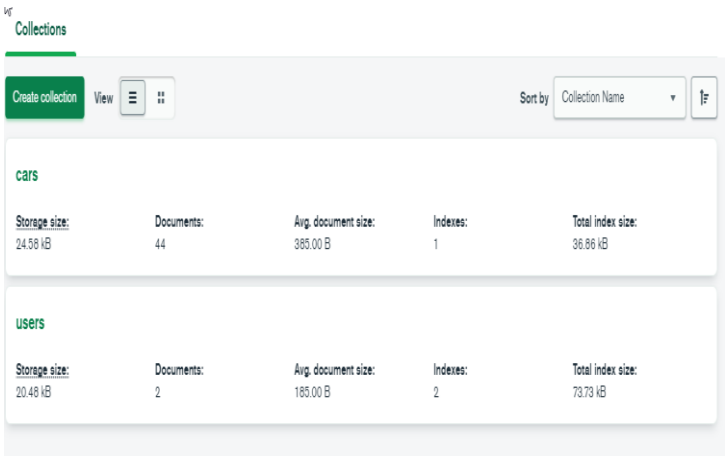


Εικόνα 20 . περιβάλλον Postman

Και μέσω της συγκεκριμένης εφαρμογής μπορούν να πραγματοποιηθούν οι απαραίτητοι έλεγχοι που απαιτούνται για την εύρυθμη λειτουργία των APIs της εφαρμογής.

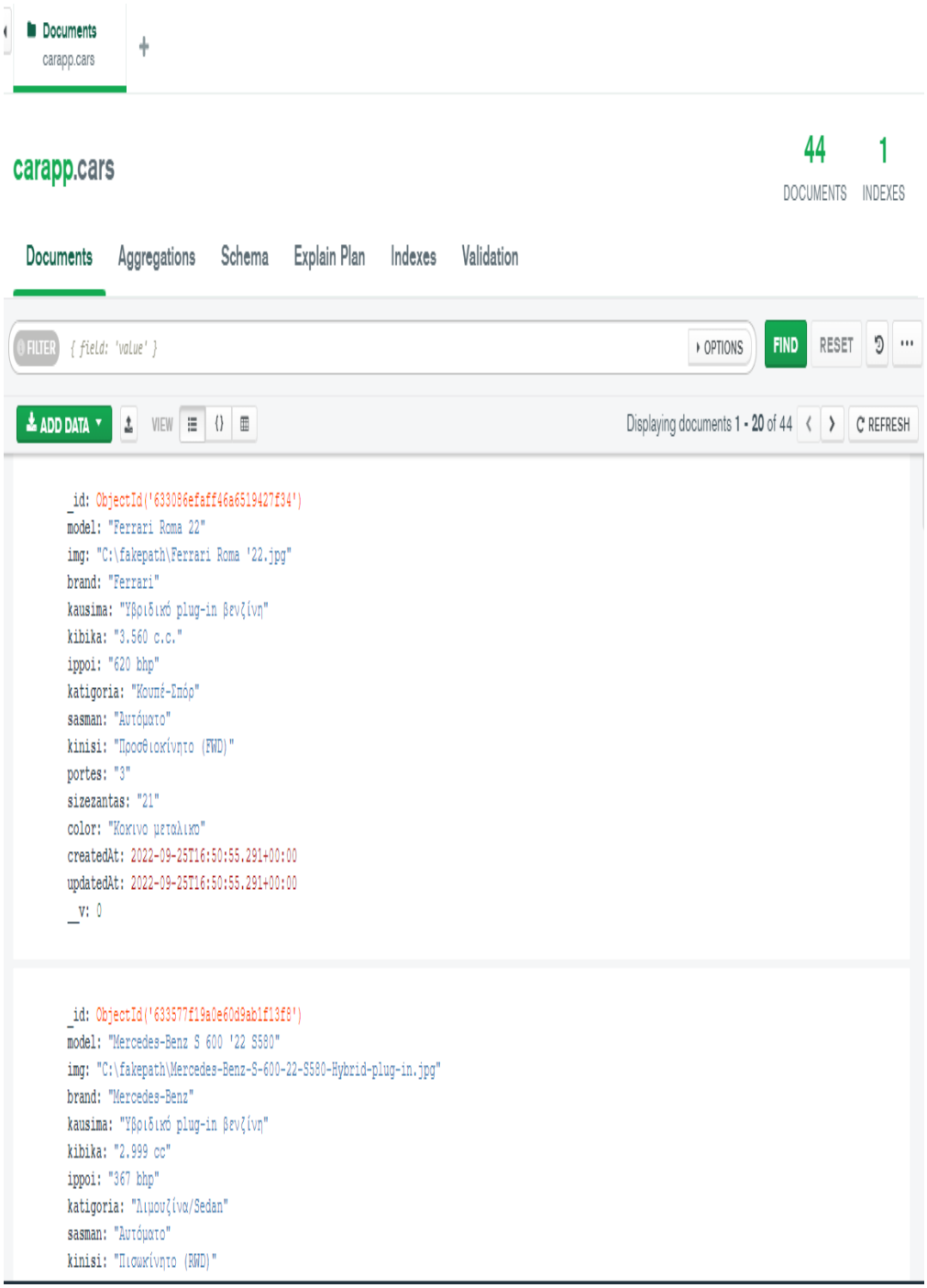
Στην συνέχεια για την εύκολη διαχείριση της βάσης δεδομένων θα χρειαστεί να εγκατασταθεί στον υπολογιστή η εφαρμογή MongoDB Compass από τον σύνδεσμο MongoDB Compass Download | MongoDB .

Και εφόσον γίνει η σύνδεση στην εφαρμογή θα παρατηρήσει πως υπάρχει η βάση με το όνομα carapp και εφόσον μπεί μέσα τους δύο πίνακες με τα ονόματα cars και users.



Εικόνα 21 . Βάσεις Δεδομένων στο MongoDB Compass

Και εφόσον μπει στον κάθε πίνακα θα μπορέσει να τον διαχειριστεί π.χ. (να προσθέσει στον πίνακα cars αυτοκίνητα ή να διαγράψει ή και να επεξεργαστεί).



Εικόνα 22 . Βάση Δεδομένων με τις κατηγορίες αυτοκινήτων

Επιπλέον μέσα στον φάκελο backend υπάρχει ο φάκελος config που μέσα υπάρχει το αρχείο

- db.js

Που εκεί γίνεται σύνδεση της μεταβλητή περιβάλλοντος από το αρχείο .env και με την βάση μας.

Παρακάτω αναλύεται ο φάκελος models που εμπεριέχει δύο αρχεία

- carModel.js
- userModel.js

Που σε αυτά τα δύο αρχεία ορίζεται ποια πεδία θα δέχεται ο κάθε πίνακας και μέσω των σχημάτων που θα δημιουργηθούν θα μπορέσει ο χρήστης να καλέσει τα συγκεκριμένα στοιχεία που χρειάζεται κάθε στιγμή.

Τέλος στον φάκελο backend υπάρχει ένας ακόμα φάκελος με το όνομα middleware που εκεί υπάρχουν δύο αρχεία με τα ονόματα

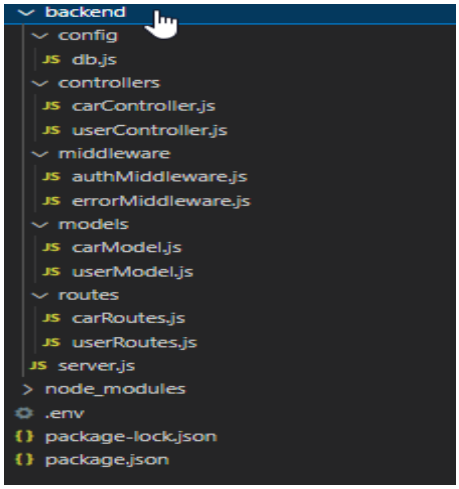
- authMiddleware.js
- errorMiddleware.js

Το Express.js είναι ένα πλαίσιο δρομολόγησης και middleware για το χειρισμό της διαφορετικής δρομολόγησης της ιστοσελίδας και λειτουργεί μεταξύ του κύκλου αίτησης και απόκρισης. Το middleware εκτελείται αφού ο διακομιστής λάβει την αίτηση και πριν από τις ενέργειες του ελεγκτή να στείλουν την απόκριση. Το middleware έχει την πρόσβαση στο αντικείμενο αίτησης, στο αντικείμενο αποκρίσεων και, στη συνέχεια, μπορεί να επεξεργαστεί την αίτηση πριν ο διακομιστής στείλει μια απάντηση. Μια εφαρμογή που βασίζεται σε Express είναι μια σειρά κλήσεων λειτουργίας ενδιάμεσου λογισμικού.

Στο αρχείο authMiddleware.js εκτελείτε μία διαδικασία που βοηθάει στο να γίνει η ταυτοποίηση των χρηστών με την βοήθεια του json web token.

Και στο αρχείο errorMiddleware.js ελέγχεται τι σφάλματα θα εμφανίσει σε περίπτωση που εφαρμογή είναι σε περιβάλλον product και τι σφάλματα θα εμφανίσει αν είναι σε περιβάλλον development.

Τα αρχεία θα φαίνονται με τον εξής τρόπο για το backend.



Εικόνα 23 . φάκελοι και αρχεία backend

2.2 Frontend

Όπως χρησιμοποιήθηκαν κάποιες τεχνολογίες στο backend έτσι θα χρειαστεί και πάλι στον τερματικό και να γίνουν κάποιες εγκαταστάσεις κάποιων τεχνολογιών.

npx i create-react-app@latest frontend - -template redux

- Create-react-app frontend : Με αυτόν τον τρόπο θα δημιουργηθεί μία εφαρμογή react μέσα σε έναν φάκελο με όνομα frontend

- - - template redux : Είναι ένα πλαίσιο redux που θα βοηθήσει στο στήσιμο της εφαρμογής.
- Έπειτα μπαίνοντας μέσα στον φάκελο frontend μέσω του τερματικού με την εντολή
- Cd frontend για να μπορεί να προστεθεί μέσα σε αυτόν τον φάκελο ότι έξτρα χρειάζεται όπως
- npm i react-rooter-dom axios react-icons react-toastify
- React-rooter-dom : Χρησιμοποιείτε το React Router για δρομολόγηση σε σελίδες με βάση τη διεύθυνση URL

- Axios : Axios Component for React with child back call function. Αυτό προορίζεται να επιτρέψει την απόδοση ασύγχρονων αιτημάτων.

- React-icons : Συμπεριλαμβάνονται δημοφιλή εικονίδια εύκολα με react-icons , το οποίο χρησιμοποιεί εισαγωγές ES6 που επιτρέπει να συμπεριληφθούν μόνο τα εικονίδια που χρησιμοποιεί το υπάρχον έργο.

- React-toastify : Το React-Toastify επιτρέπει να προστεθούν ειδοποιήσεις στην εφαρμογή με ευκολία.

Τώρα όπως και προηγουμένως στο αρχείο package.json που βρίσκεται έξω από τον φάκελο frontend θα προστεθεί στο σημείο scripts{} το client ώστε με αυτόν τον τρόπο να ανοίξει η εφαρμογή πληκτρολογώντας στον τερματικό την εντολή npm start

```
"scripts": {
  "start": "node backend/server.js",
  "server": "nodemon backend/server.js",
  "client": "npm start --prefix frontend"
},
```

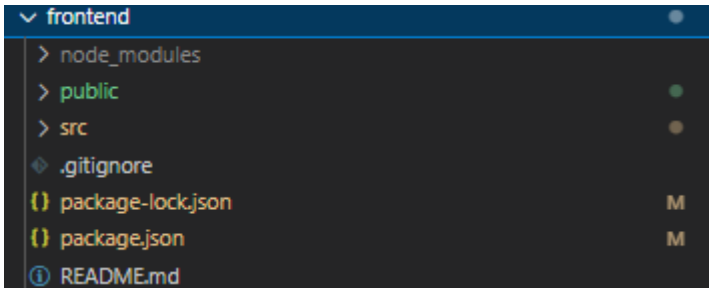
Εικόνα 24 . τρόπος για να τρέξει το frontend

Επιπλέον διότι προηγούμενός προστέθηκε μέσω τερματικού το npm i -D conurrently. Στο ίδιο σημείο που τοποθετήθηκε το client ώστε να ανοίγει το frontend. Πλέον με την εντολή npm run dev θα μπορεί να τρέξει και το frontend και το backend ταυτόχρονα.

```
"scripts": {
  "start": "node backend/server.js",
  "server": "nodemon backend/server.js",
  "client": "npm start --prefix frontend",
  "dev": "concurrently \"npm run server\" \"npm run client\""
},
```

Εικόνα 25 . τρόπος για να τρέξει το backend μαζί με το frontend

Ο φάκελος frontend μέσα θα έχει την εξής μορφή.



Εικόνα 26 . φάκελοι και αρχεία frontend

Στον Φάκελο node_modules βρίσκονται όλα όσα χρειάζεστε από τις τεχνολογίες που χρησιμοποιήθηκαν στην εφαρμογή.

Στον φάκελο public βρίσκονται κάποιες φωτογραφίες που χρησιμοποιήθηκαν, το favicon.ico και το αρχείο html ώστε να το διαβάσει ο φυλλομετρητής και να εμφανίζεται στον χρήστη.

Στην συνέχεια υπάρχει το αρχείο src που εκεί εμπεριέχει τα αρχεία που θα διαχειριστούμε. Αρχικά θα δημιουργηθούν κάποιοι φάκελοι.

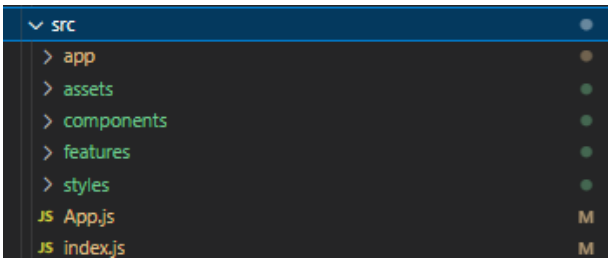
- App
- Assets

- Components
- Features
- Styles

Και υπάρχουν και δύο αρχεία

- App.js
- Index.js

Όποτε η κατανομή θα είναι όπως την παρακάτω εικόνα.



Εικόνα 27 . φάκελοι και αρχεία src

Το αρχείο index.js έχει να κάνει με την εφαρμογή. Δηλαδή σε αυτό το σημείο χρησιμοποιείτε ο τρόπος με τον οποίο γίνεται η σύνδεση της εφαρμογής με το αρχείο index.html. Επίσης σε αυτό το αρχείο αρχικοποιούνται κάποια αρχεία .css για να οριστούν καθολικά σε όλη την εφαρμογή.

Έπειτα το αρχείο app.js χρησιμοποιείται για να γίνει η διαχείρισή των components για να οριστούν σε ποια URLs θα χρησιμοποιηθούν και ποιο component θα ανοίγει στο κατάλληλο URL και όλα αυτά με την βοήθεια του react rooter dom.

Έπειτα παρατηρείτε τον φάκελο assets που σε αυτόν τον φάκελο βρίσκονται οι φωτογραφίες και τα βίντεο που εμφανίζονται στην ιστοσελίδα.

Στην συνέχεια υπάρχει ο φάκελος app που δημιουργείτε μόλις γίνει η εγκατάσταση του redux - -template απλός αφαιρέθηκε ότι υπήρχε μέσα στον φάκελο και δημιουργήθηκε ένα αρχείο με το όνομα store.js.

Στον φάκελο features υπάρχει ο φάκελος auth που σε αυτό τον φάκελο υπάρχουν δύο αρχεία με τα ονόματα

- authService.js
- authSlice.js

Το authService.js βοηθάει στο να τραβήξει τα δεδομένα από το API http://localhost:5000/api/users/ με την βοήθεια του axios.

Ώστε να μπορεί ο χρήστης να κάνει εγγραφή, σύνδεση και αποσύνδεση.

Στο αρχείο authSlice.js ορίζονται όλα τα μηνύματα που θα εμφανιστούν στον χρήστη κατά την προσπάθεια σύνδεσης είτε αυτή είναι επιτυχημένη είτε όχι.

Τέλος ο φάκελος components που εμπεριέχει αρκετά αρχεία όπως

- AboutUs.js
- AddCar.js
- Car.js
- CarList.js
- Cars.js
- CompareCars.js
- Copyright.js
- EditCar.js
- Footer.js
- Header.js
- HomePage.js
- Login.js
- Register.js
- ScrollToTop.js

Τα αρχεία Login.js και Register.js είναι το ένα για την σύνδεση του χρήστη και το άλλο για την εγγραφή του χρήστη στην βάση.

Register.js είναι το αρχείο για την εγγραφή του χρήστη στην βάση δεδομένων και ελέγχει αν ο χρήστης βάζει στοιχεία που δεν έχουν χρησιμοποιηθεί από άλλον χρήστη. Επίσης, ελέγχει αν ο χρήστης έχει γράψει σωστά τους δύο κωδικούς, γιατί σε περίπτωση που έχει γράψει διαφορετικούς θα εμφανιστεί το κατάλληλο μήνυμα λάθους. Και τέλος εμφανίζει το πως θα εμφανίζεται το συγκεκριμένο αρχείο στον φυλομετρητη (browser).

```
import { useState, useEffect } from 'react'

import { useSelector, useDispatch } from 'react-redux'

import { useNavigate } from 'react-router-dom'

import { toast } from 'react-toastify'

import { FaUser } from 'react-icons/fa'

import { register, reset } from '../features/auth/authSlice'

function Register() {

  const [formData, setFormData] = useState({

    name: '',

    email: '',
```

```
  password: '',
  password2: '',
})

const { name, email, password, password2 } = formData
const navigate = useNavigate()
const dispatch = useDispatch()
const { user,
  isError, isSuccess, message } = useSelector(
  (state) => state.auth
)
useEffect(() => {
  if (isError) {
    toast.error(message)
  }
  if (isSuccess || user) {
    navigate('/')
  }
  dispatch(reset())
}, [user, isError, isSuccess, message, navigate, dispatch])
const onChange = (e) => {
  setFormData((prevState) => ({
    ...prevState,
    [e.target.name]: e.target.value,
  }))
}
const onSubmit = (e) => {
```

```
  e.preventDefault()

  if (password !== password2) {
    toast.error('Passwords do not match')
  } else {
    const userData = {
      name,
      email,
      password,
    }
    dispatch(register(userData))
  }
}
return (
  <>
    <div className='heading'>
      <h1>
        <FaUser /> Register
      </h1>
      <p>CAR FOR YOU</p>
    </div>

    <div className="columns">
    <div className='form'>
      <form onSubmit={onSubmit}>
      <br/>
        <div className="field">
```

```
<div className='control'>
  <input
    type='text'
    className='input'
    id='name'
    name='name'
    value={name}
    placeholder='Enter your name'
    onChange={onChange}
  />
</div>
</div>

  <div className="field" >
  <div className='control'>
  <input
    type='email'
    className='input'
    id='email'
    name='email'
    value={email}
    placeholder='Enter your email'
    onChange={onChange}
  />
</div>
  </div>
  <div className="field" >
```

```
  <div className='control'>
  <input
    type='password'
    className='input'
    id='password'
    name='password'
    value={password}
    placeholder='Enter password'
    onChange={onChange}
  />
</div>
  </div>
  <div className="field" >
  <div className='control'>
  <input
    type='password'
    className='input'
    id='password2'
    name='password2'
    value={password2}
    placeholder='Confirm password'
    onChange={onChange}
  />
</div>
  </div>
  <div className="field" >
  <div className='control'>
```

```
            <button type='submit' className='btn btn-cars'>
              Register
            </button>
          </div>
            </div>
        </form>
      </div>
        </div>
    </>
  )
}
export default Register
```

Στον Κώδικα παρατειρείτε πως με την χρήση των απαραίτειτων ενεργείων μπορεί ο χρήστης να ολοκληρώση την εγραφή του σε μία φόρμα επικοινωνίας.

Login.js είναι το αρχείο που ο χρήστης κάνει σύνδεση στην εφαρμογή τακτοποιώντας τα στοιχεία που λέει με τα αντίστοιχα στοιχεία αν υπάρχουν στην βάση. Και τέλος εμφανίζει το πως θα εμφανίζεται το συγκεκριμένο αρχείο στον φυλομετρητη (browser).

```
import { useState, useEffect } from 'react'
import { FaSignInAlt } from 'react-icons/fa'
import { useSelector, useDispatch } from 'react-redux'
import { useNavigate } from 'react-router-dom'
import { toast } from 'react-toastify'
import { login, reset } from '../features/auth/authSlice'
function Login() {
  const [formData, setFormData] = useState({
    email: '',
    password: '',
  })
```

```
const { email, password } = formData

const navigate = useNavigate()
const dispatch = useDispatch()
const { user,
  isError, isSuccess, message } = useSelector(
  (state) => state.auth
)
useEffect(() => {
  if (isError) {
    toast.error(message)
  }
  if (isSuccess || user) {
    navigate('/')
  }
  dispatch(reset())
}, [user, isError, isSuccess, message, navigate, dispatch])
const onChange = (e) => {
  setFormData((prevState) => ({
    ...prevState,
    [e.target.name]: e.target.value,
  }))
}

const onSubmit = (e) => {
  e.preventDefault()
  const userData = {
```

```
    email,
    password,
  }
  dispatch(login(userData))
}
return (
  <>
    <div className='heading'>
      <h1>
        <FaSignInAlt /> Login
      </h1>
      <p>CAR FOR YOU</p>
    </div>
    <div className="columns">
    <div className='form'>
      <form onSubmit={onSubmit}>
        <br/>
        <div className="field">
        <div className='control'>
          <input
            type='email'
            className='input'
            id='email'
            name='email'
            value={email}
            placeholder='Enter your email'
            onChange={onChange}
```

```
        />
      </div>
       </div>
       <div className="field">
       <div className='control'>
       <input
         type='password'
         className='input'
         id='password'
         name='password'
         value={password}
         placeholder='Enter password'
         onChange={onChange}
       />
      </div>
       </div>
       <div className="field">
       <div className='control'>
       <button type='submit' className='btn btn-cars'>
         Login
       </button>
      </div>
       </div>
    </form>
  </div>
  </div>
</>
```

```
 )

}

export default Login
```

Στον Κώδικα παρατειρείτε πως με την χρήση των απαραίτειτων ενεργείων μπορεί να γίνει ο έλεγχος έτσι ώστε να ελεγθεί αν ο χρήστης υπαρχει στην βάση δεδομένων και αν τα στοιχεία που πληκτρολογισέ είναι σωστά.

Στην συνέχεια βλέπουμε ότι υπάρχει το αρχείο με το όνομα CarList.js. Σε αυτό το αρχείο παρατηρείτε πως τραβάει όλα τα αυτοκίνητα που είναι αποθηκευμένα στην βάση και τα σερβίρει μπροστά στον χρήστη. Επίσης δίνει την δυνατότητα στον χρήστη να επεξεργαστεί, να διαγράψει ένα αυτοκίνητο και τέλος να προσθέσει ένα καινούριο.

Πατώντας ο χρήστης το κουμπί επεξεργασία ή το κουμπί προσθήκη ενός αυτοκινήτου μεταβαίνει στην αντίστοιχη σελίδα για να επεξεργαστεί η να προσθέσει ένα αυτοκίνητο.

Στην συνέχεια υπάρχει το αρχείο CompareCars.js που στο συγκεκριμένο αρχείο βρίσκεται η σελίδα που γίνεται η σύγκριση των αυτοκινήτων μεταξύ αυτών που έχει επιλέξει ο χρήστης.

Τέλος υπάρχουν τα αρχεία

- Header.js
- Footer.js
- Copyright.js
- Homepage.js
- Cars.js
- Car.js
- AboutUs.js
- ScrollToTop.js

Το Header.js, Footer.js, Copyright.js είναι τα αχρεία που βοηθάνε τον χρήστη να περιηγηθεί μέσα στην εφαρμογή.

Το HomePage.js είναι η κεντρική σελίδα τις εφαρμογής. Είναι η πρώτη σελίδα που θα δει ο χρήστης την ώρα που θα συνδεθεί.

```
import React from 'react'

import { Link } from 'react-router-dom';

import "../styles/homepage.css";

// image import

import image from "../assets/car-4.png"
```

```
// video import
import carForYou from "../assets/video-main.mp4"
// New
import { useNavigate } from 'react-router-dom'
import { useSelector} from 'react-redux'
import { useEffect } from 'react'
function HomePage() {
  const navigate = useNavigate()
  const { user } = useSelector((state) => state.auth)
  useEffect(() => {
    if (!user) {
      navigate('/login')
    }
  }, [user, navigate])
  return (
    <main>
      <>
            <video >
              <source src={carForYou} type="video/mp4" />
            </video>
           <div className='container'>
            <div className='section'>
              <div className='info-home'>
                <div className='image-wrapper'>
                  <img src={image} alt='vintage' />
                </div>
                <div>
```

<h2>#Carsforyou</h2>

Τα τελευταία χρόνια η πληροφορική έχει εισχωρήσει σε πολλούς κλάδους και ένας από αυτούς είναι οι διαδικτυακές εφαρμογές.

Η ανάλυση των δυνατοτήτων δύο ή περισσότερων αυτοκινήτων είναι μία από αυτές τις εφαρμογές. Γενικότερα ο χρήστης με ένα μόνο κουμπί

μπορεί να συγκρίνει αυτοκίνητα, είτε για να λάβει πληροφορίες είτε για να αποκτήσει μία γνώση που μπορεί να τον βοηθήσει για το

μελλοντικό του αυτοκίνητο. Επίσης έχουμε δημιουργήσει ένα σύστημα διαχείρισής των χρηστών έτσι ώστε ο κάθε χρήστης να συνδέεται με

τα προσωπικά του στοιχεία και όλο αυτό με την βοήθεια μίας βάσης δεδομένων. Έχει παρατηρηθεί πως ο κόσμος ολοένα και εμπιστεύεται τέτοιου

είδους εφαρμογές γιατί κάνουν τους χρήστες να έχουν μία άμεση επαφή με την πληροφορία.

Οι εφαρμογές εξελίσσονται με ιλιγγιώδης ρυθμούς και οι ο ανταγωνισμός που υπάρχει βοηθάει στο να έχουμε καθημερινά πολύ

δημιουργικά και χρήσιμα εργαλεία.

</>


```
    </main>
  )
}
export default HomePage
```

Ο κώδικας παραπάνω παρουσιάζει το κύριο περιεχόμενο της σελίδας που θα δει ο χρήστης από την στιγμή που θα έχει πραγματοποιήσει εγγραφή και σύνδεση στην εφαρμογή.

Το Cars.js

Είναι η σελίδα που τραβάει όλα τα αυτοκίνητα από την βάση δεδομένων και τα εμφανίζει μπροστά στον χρήστη. Επίσης μπορεί ο χρήστης να επιλέξει όποιο αυτοκίνητο επιθυμεί ώστε να το προσθέσει στην σελίδα συγκρίσεις αυτοκινήτων (CompareCars.js) για να μπορεί να κάνει σύγκριση αυτοκινήτων.

Το Car.js είναι το αρχείο που τραβάει από την βάση δεδομένων και εμφανίζει το κάθε αυτοκίνητο ξεχωριστά με βάση το μοναδικό id που έχει το κάθε αυτοκίνητο.

Τέλος, το αρχείο AboutUs.js είναι το αρχείο που αναγράφονται λίγα λόγια για την εφαρμογή.

Ο τελευταίος φάκελος είναι ο φάκελος styles που εμπεριέχει όλα τα αρχεία για την κάθε σελίδα ξεχωριστά. Όλα τα αρχεία έχουν την κατάληξη .ccs που αυτό σημαίνει πώς ότι αναγράφεται μέσα στο κάθε αρχείο είναι για την εμφάνιση (design) της κάθε σελίδας στον φυλλομετριτή (browser).

Τα αρχεία είναι

- about-us.css
- back-to-top.css
- basic.css
- car.css
- cars.css
- compare.css
- footer.css
- global.css
- header.css

σε αυτό το σημείο ολοκληρώθηκε το πώς θα φαίνεται η εφαρμογή στον χρήστη.

```
.top-to-btm {
```

```
position: relative;
}
.icon-position {
 position: fixed;
 bottom: 40px;
 right: 25px;
 z-index: 20;
}

.icon-style {
background-color: var(--mainColor);
border: 2px solid #fff;
border-radius: 50%;
height: 50px;
 width: 50px;
 color: #fff;
cursor: pointer;
animation: movebtn 3s ease-in-out infinite;
transition: all .5s ease-in-out;
}
.icon-style:hover {
-webkit-animation: none;
animation: none;
background: #fff;
color: var(--mainColor);
border: 2px solid var(--mainColor);
}
```

```
@-webkit-keyframes movebtn {
   0% { transform: translateY(0px); }
   25% { transform: translateY(20px); }
   50% { transform: translateY(0px); }
   75% { transform: translateY(-20px); }
   100% { transform: translateY(0px); }
}
@keyframes movebtn {
   0% { transform: translateY(0px); }
   25% { transform: translateY(20px); }
   50% { transform: translateY(0px); }
   75% { transform: translateY(-20px); }
   100% { transform: translateY(0px); }
}
```

Παραπάνω είναι ένα μέρος του κώδικα css για το scroll to top δηλαδή πως φαίνεται το κουμπί που βρίσκεται στο κάτω μέρος της οθόνης μας .

Τέλος ο κώδικας css έχει περαστεί από ένα online πρόγραμμα για να μπορέσουμε να έχουμε τον βέλτιστο κώδικα css.

Δηλαδή αν υπήρχε στον κώδικα:

```
.example {
   display: grid;
   transition: all .5s;
   user-select: none;
   background: linear-gradient(to bottom, white, black);
}
```

Το πρόγραμμα θα επέστρεφε:

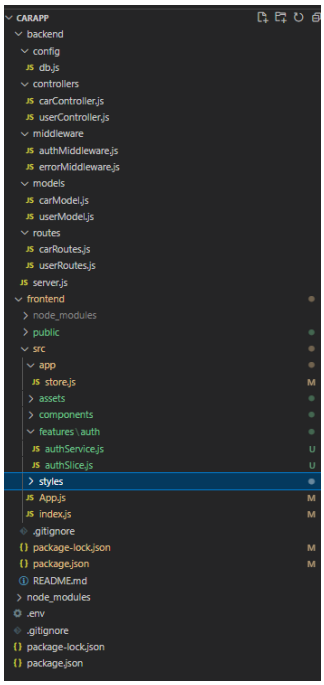
```
.example {
   display: -ms-grid;
```

```
display: grid;
-webkit-transition: all .5s;
-o-transition: all .5s;
transition: all .5s;
-webkit-user-select: none;
   -moz-user-select: none;
    -ms-user-select: none;
        user-select: none;
background: -webkit-gradient(linear, left top, left bottom, from(white), to(black));
background: -o-linear-gradient(top, white, black);
background: linear-gradient(to bottom, white, black);
}
```

Με αυτόν τον τρόπο φαίνεται πως έγιναν κάποιες προσθήκες στον κώδικα, έτσι ώστε να μπορεί το περιεχόμενο να λειτουργήσει σε περισσότερους browsers.

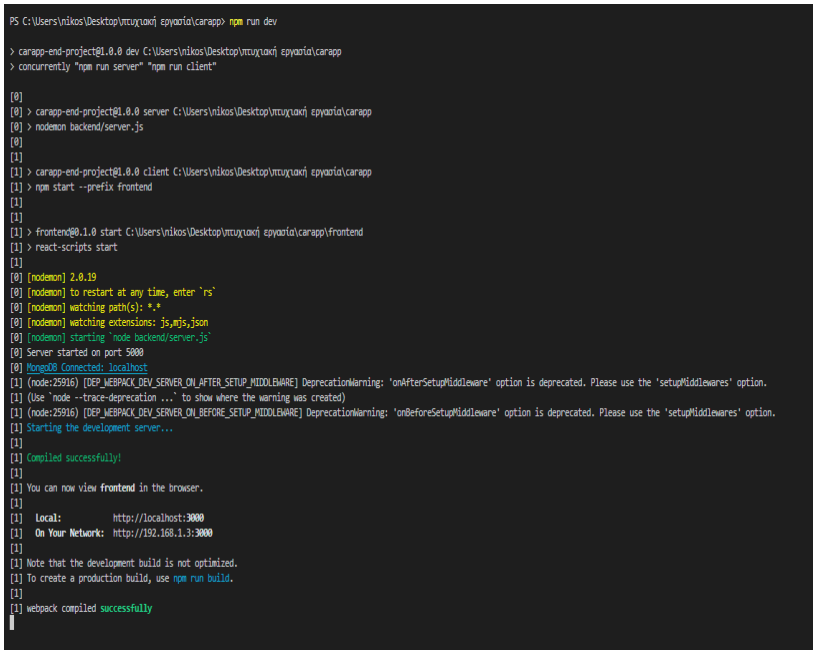
Παρουσίαση εφαρμογής

Παρακάτω φαίνεται η τελικά δομή της εφαρμογής.



Εικόνα 28 . αρχεία και φάκελοι front-end

Στην συνέχεια, στον τερματικό αναγράφοντας την παρακάτω εντολή για να τρέξει ταυτόχρονα και ο φάκελος backend που βρίσκεται το αρχείο server.js αλλά και ο φάκελος frontend, που βρίσκονται όλα τα αρχεία για το πως θα εμφανίζεται η εφαρμογή στον χρήστη.



Εικόνα 29 . τερματικός κατά την ώρα εκτέλεσης

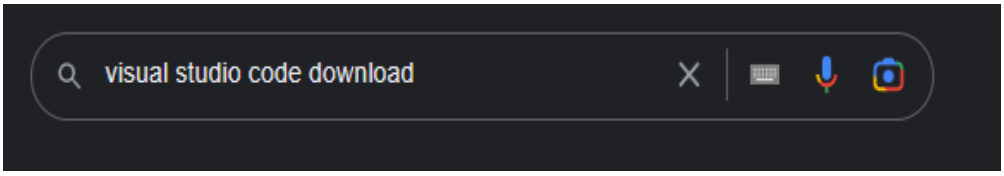
Κεφάλαιο 3ο: Ανάλυση εγκατάστασής και παρουσίαση έργου

3.1 Ανάλυση βημάτων για την υλοποίηση του έργου

Ο στόχος του τρίτου κεφαλαίου της παρούσας εργασίας είναι ανάλυση βήμα προς βήμα εγκατάστασης όλων των τεχνολογιών, πως όλες αυτές μπορούν να εργαστούν μεταξύ τους και τέλος θα γίνει παρουσίαση της εφαρμογής και των δυνατοτήτων της.

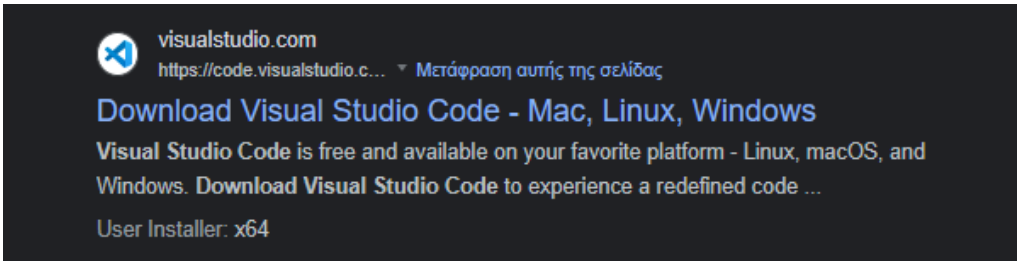
Βήμα 1ο: Αρχικά γίνεται εγκατάσταση του Visual Studio Code. Ο χρήστης μεταβαίνει σε έναν οποιονδήποτε browser π.χ.(Firefox, Google Chrome, Microsoft Edge κ.λ.π.). Στην συνέχεια ο χρήστης πηγαίνει σε μία μηχανή αναζήτησης π.χ.(google, windows bing κ.λ.π.). Για τη συγκεκριμένη εργασία χρησιμοποιήθηκαν σαν browser ο Edge και για μηχανή αναζήτησης η www.google.com. Εφόσον ο χρήστης επιλέξει να κάνει την εγκατάσταση με www.google.com ο τρόπος είναι ο εξής.

- Αρχικά ο χρήστης θα γράψει στην μηχανή αναζήτησης visual studio code download.



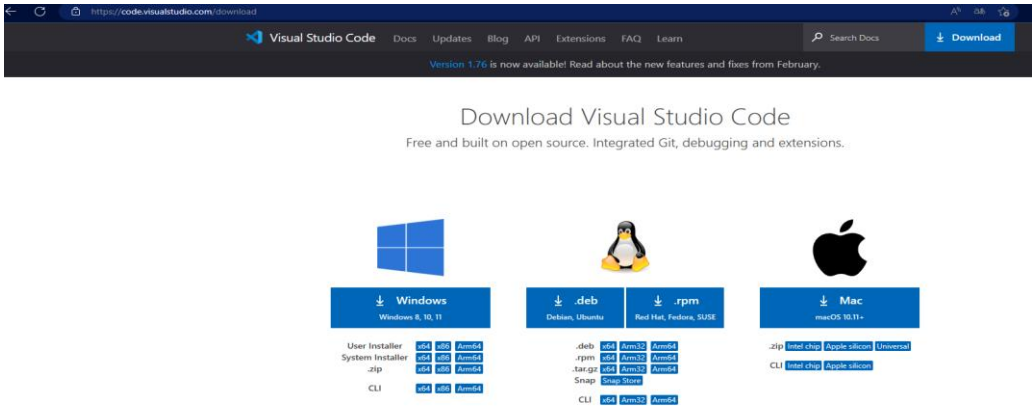
Εικόνα 30. Visual Studio Code

- Στην συνέχεια θα του εμφανίσει κάποια αποτελέσματα εκ των οποίων θα επιλέξει το παρακάτω.



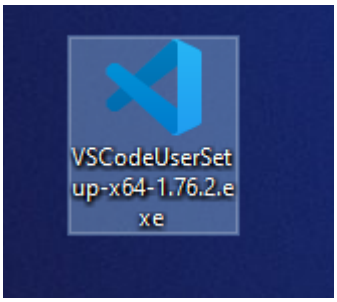
Εικόνα 31. Visual Studio Code search

- Εφόσον ο χρήστης έχει μεταφερθεί στην ιστοσελίδα https://code.visualstudio.com/Download μπορεί να επιλέξει τον τρόπο εγκατάστασης που επιθυμεί (Για την συγκεκριμένη εργασία έχει χρησιμοποιηθεί η γενική εγκατάσταση για windows).



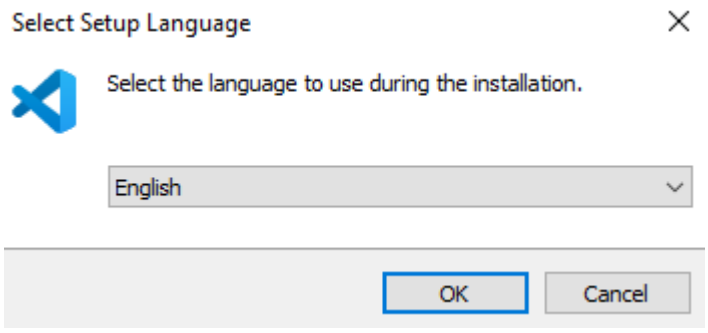
Εικόνα 32. Download Visual Studio Code

- Εφόσον ο Χρήστης έχει επιλέξει την εγκατάσταση του visual studio code με windows στον υπολογιστή του θα εμφανιστεί ένα εκτελέσιμο αρχείο για να μπορέσει να κάνει την εγκατάσταση του προγράμματος.



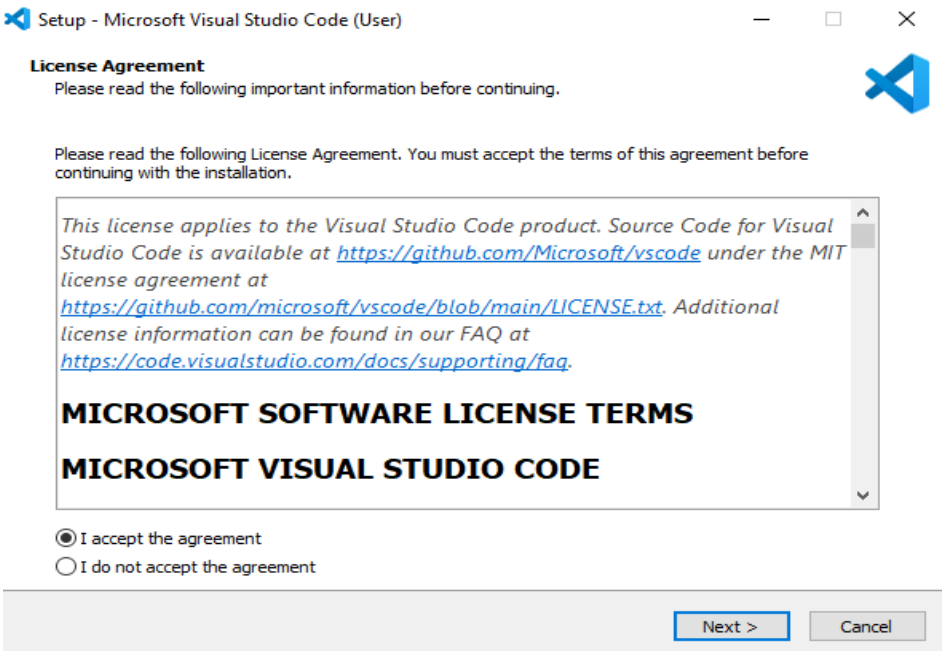
Εικόνα 33. Εκτελέσιμο εικονίδιο Vs code

- Στην συνέχεια τρέχει το εκτελέσιμο αρχείο και επιλεγεί την γλώσσα που επιθυμεί.



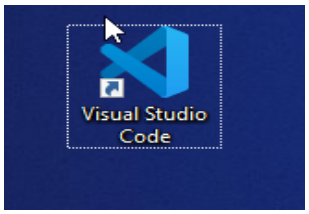
Εικόνα 34. Επιλογή γλώσσας

- Τέλος κάνει εγκατάσταση του visual studio code.



Εικόνα 35. Εγκατάσταση Visual Code

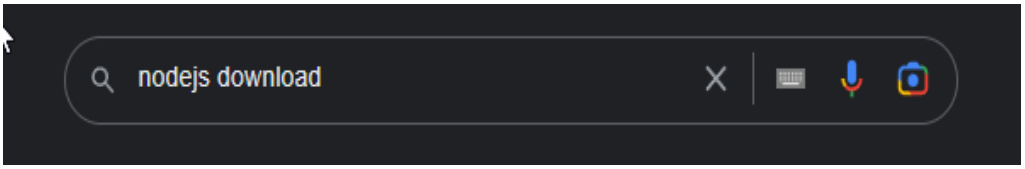
- Εφόσον ολοκληρωθεί η εγκατάσταση θα του εμφανιστεί στην επιφάνεια εργασίας το παρακάτω εικονίδιο.



Εικόνα 36. Εικονίδιο Visual Studio

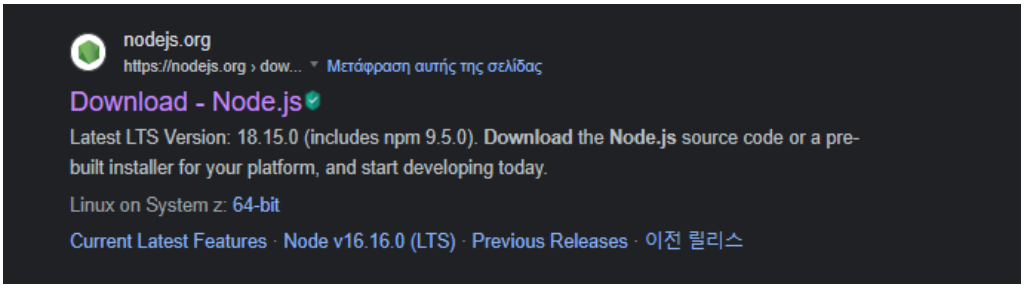
Βήμα 2ο: Τώρα ο χρήστης θα κάνει εγκατάσταση στην nodejs.

- Με τον ίδιο τρόπο όπως και παραπάνω ο χρήστης θα κάνει αναζήτηση σε μία μηχανή αναζήτησης nodejs download.



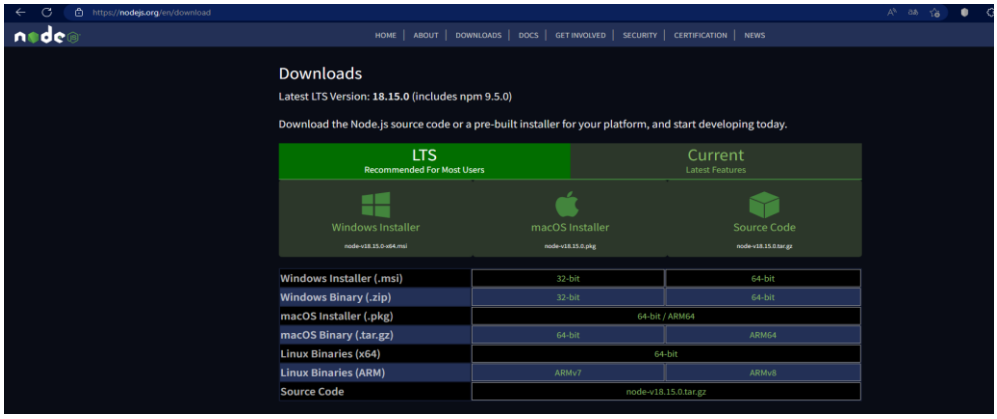
Εικόνα 37. Nodejs search

- Στην συνέχεια θα του εμφανίσει κάποια αποτελέσματα εκ των οποίων θα επιλέξει το παρακάτω.



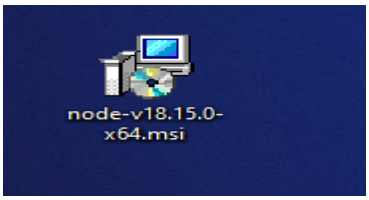
Εικόνα 38. Node.js Download

- Εφόσον ο χρήστης έχει μεταφερθεί στην ιστοσελίδα https://nodejs.org/en/download μπορεί να επιλέξει τον τρόπο εγκατάστασης που επιθυμεί (Για την συγκεκριμένη εργασία έχει χρησιμοποιηθεί η εγκατάσταση με LTS (Recommended For Most Users) Windows Istaller.



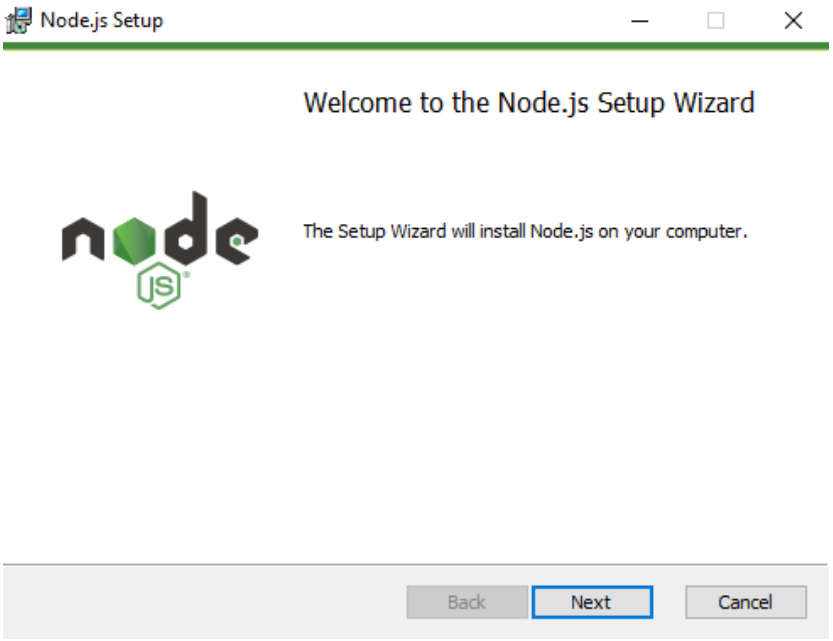
Εικόνα 39. Node.js Download

- Εφόσον ο Χρήστης έχει επιλέξει την εγκατάσταση του node.js με windows στον υπολογιστή του θα εμφανιστεί ένα εκτελέσιμο αρχείο για να μπορέσει να κάνει την εγκατάσταση του προγράμματος.



Εικόνα 40. Εκτελέσιμο εικονίδιο Node.js

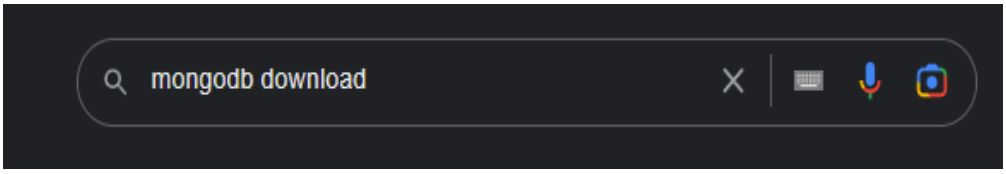
- Στην συνέχεια τρέχει το εκτελέσιμο αρχείο και κάνει εγκατάσταση του node.js.



Εικόνα 41. Set up Node.js

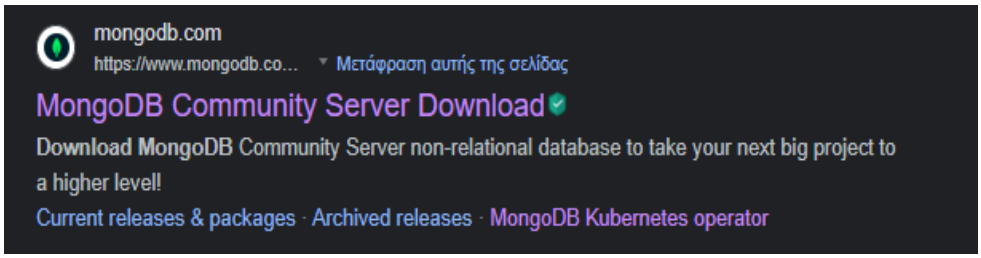
Βήμα 3ο: Ο χρήστης θα κάνει εγκατάσταση το MongoDB.

- Με τον ίδιο τρόπο όπως και παραπάνω ο χρήστης θα κάνει αναζήτηση σε μία μηχανή αναζήτησης mongobd download.



Εικόνα 42. Mongoddp download

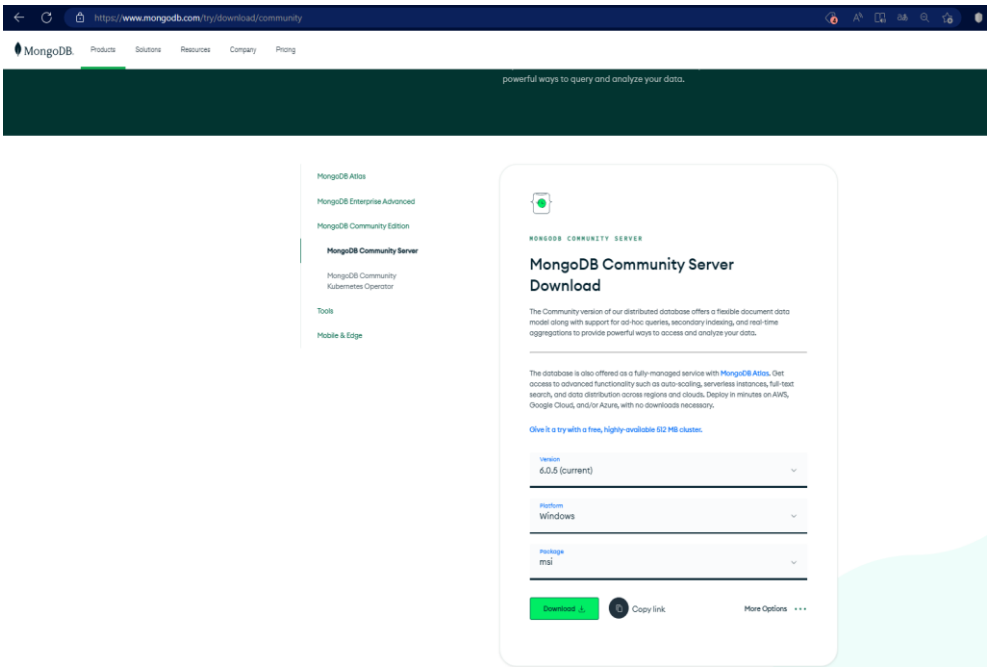
- Στην συνέχεια θα του εμφανίσει κάποια αποτελέσματα εκ των οποίων θα επιλέξει το παρακάτω.



Εικόνα 43. MongoDB download

- Εφόσον ο χρήστης έχει μεταφερθεί στην ιστοσελίδα https://www.mongodb.com/try/download/community μπορεί να επιλέξει τον

τρόπο εγκατάστασης που επιθυμεί (Για την συγκεκριμένη εργασία έχει χρησιμοποιηθεί η εγκατάσταση Windows.

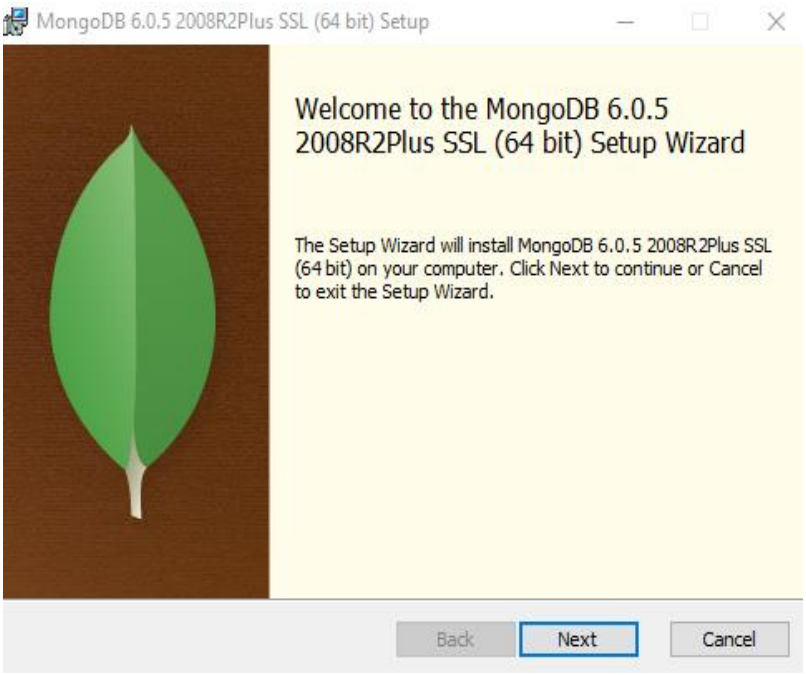


Εικόνα 44. MongoDB download

- Εφόσον ο χρήστης έχει επιλέξει την εγκατάσταση του mongobd με windows στον υπολογιστή του θα εμφανιστεί ένα εκτελέσιμο αρχείο για να μπορέσει να κάνει την εγκατάσταση του προγράμματος.

Εικόνα 45. Εκτελέσιμο εικονίδιο MongoDB

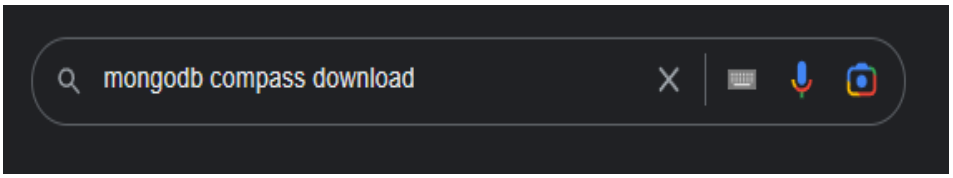
- Στην συνέχεια τρέχει το εκτελέσιμο αρχείο και κάνει εγκατάσταση του mongodb.



Εικόνα 46. Set up MongoDB

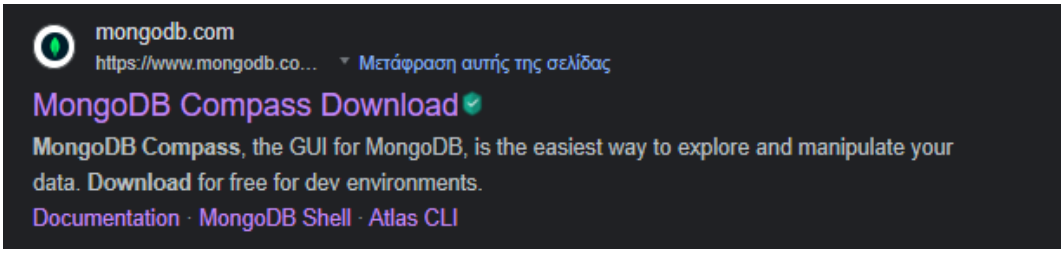
Βήμα 4ο: Ο χρήστης θα κάνει εγκατάσταση το MongoDB Compass.

- Με τον ίδιο τρόπο όπως και παραπάνω ο χρήστης θα κάνει αναζήτηση σε μία μηχανή αναζήτησης mongobd compass download.



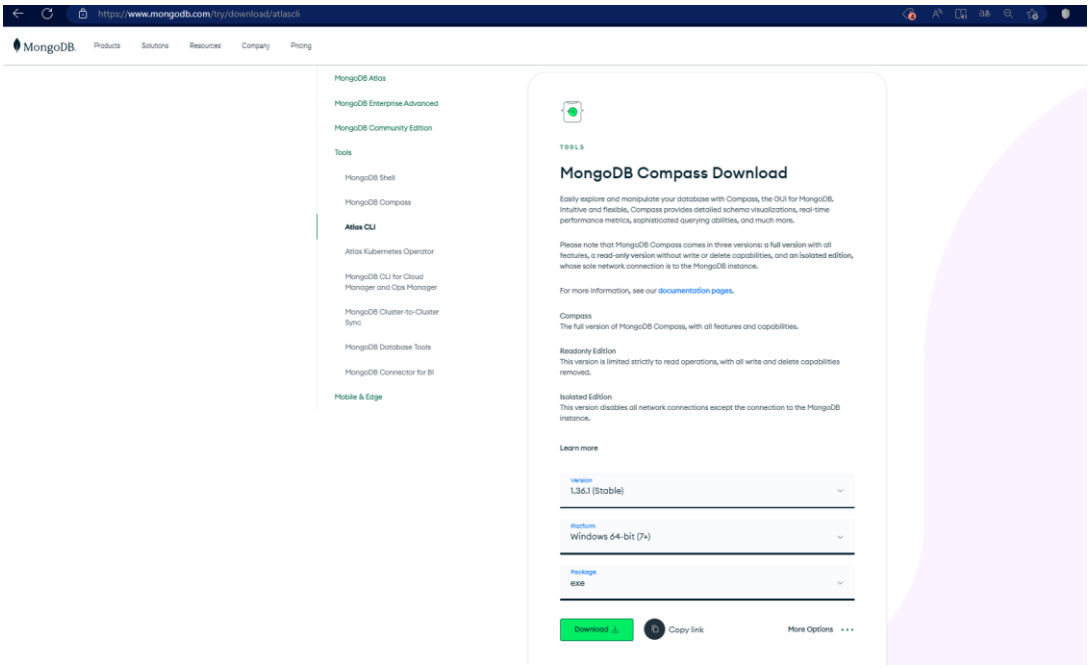
Εικόνα 47. Mongo DB Compass

- Στην συνέχεια θα του εμφανίσει κάποια αποτελέσματα εκ των οποίων θα επιλέξει το παρακάτω.



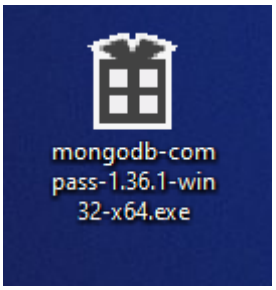
Εικόνα 48.Mongo DB Compass download

- Εφόσον ο χρήστης έχει μεταφερθεί στην ιστοσελίδα https://www.mongodb.com/try/download/compass μπορεί να επιλέξει τον τρόπο εγκατάστασης που επιθυμεί (Για την συγκεκριμένη εργασία έχει χρησιμοποιηθεί η εγκατάσταση Windows.



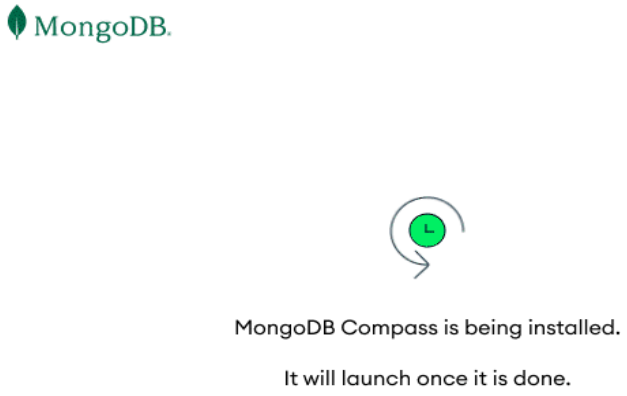
Εικόνα 49. Mongo DB Compass Download

- Εφόσον ο Χρήστης έχει επιλέξει την εγκατάσταση του mongobd compass με windows στον υπολογιστή του θα εμφανιστεί ένα εκτελέσιμο αρχείο για να μπορέσει να κάνει την εγκατάσταση του προγράμματος.



Εικόνα 50. Εκτελέσιμο εικονίδιο Mongo DB Compass

- Στην συνέχεια τρέχει το εκτελέσιμο αρχείο και κάνει εγκατάσταση του mongodb compass.



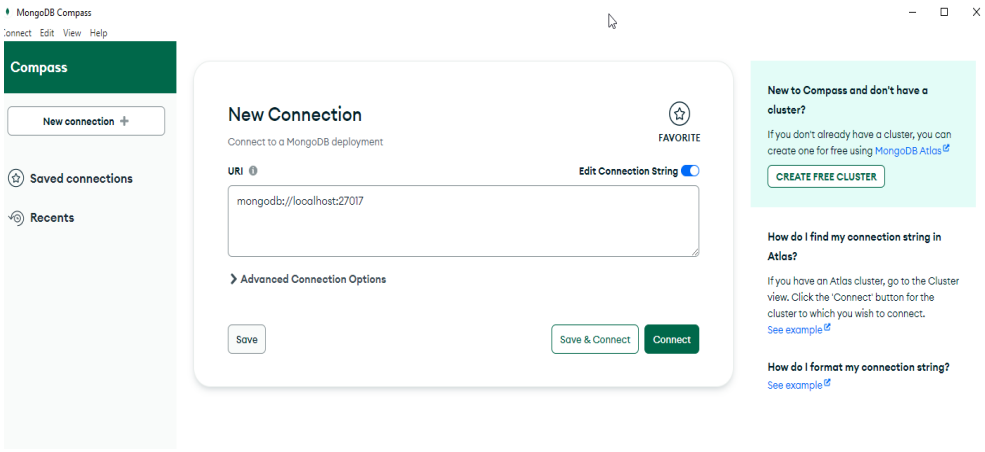
Εικόνα 51. Mongo DB Compass install

- Εφόσον ολοκληρωθεί η εγκατάσταση θα του εμφανιστεί στην επιφάνεια εργασίας το παρακάτω εικονίδιο.

Εικόνα 52. Mongo DB Compass

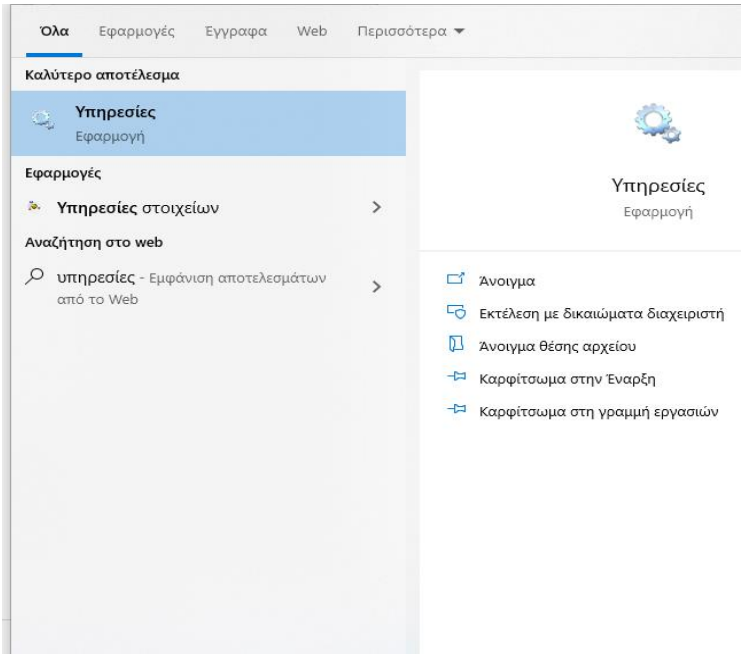
Βήμα 5ο: Από την στιγμή που έχει ολοκληρωθεί η εγκατάσταση των προγραμμάτων. Ο χρήστης θα συνεχίσει με τα παρακάτω βήματα για να τρέξει την εφαρμογή.

- Θα ανοίξει το MongoDBCompass και θα επιλέξει το Connect



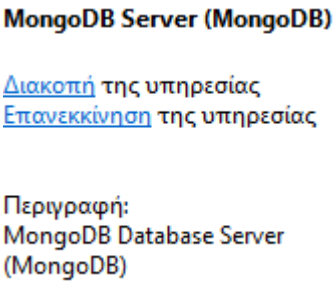
Εικόνα 53. Mongo DB Compass Connect

Σε περίπτωση που εμφανίσει αυτό το μηνυμά; (connect ECONNREFUSED 127.0.0.1:27017) τότε πρέπει να πάει ο χρήστης στην έναρξη των windows και να κατευθύνθει στις Υπηρεσίες.



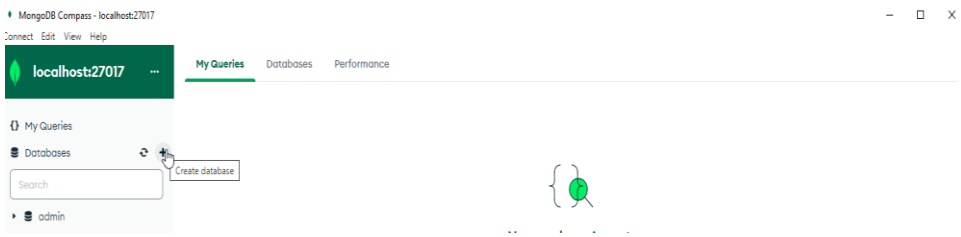
Εικόνα 54. Άνοιγμα Υπηρεσιών

Στην συνέχεια θα βρει το MongoDB Server (MongoDB) και θα δει αν είναι ενεργοποιημένο και θα το ενεργοποίηση ή θα κάνει επανεκκίνηση.

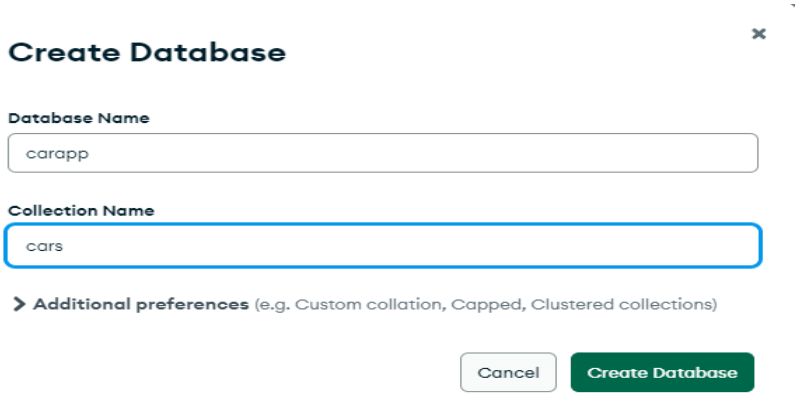


Εικόνα 55. MongoDB Server

- Στην συνέχεια εφόσον συνδεθεί θα πάει να δημιουργήσει μία βάση δεδομένων με το όνομα carapp και για collection cars.



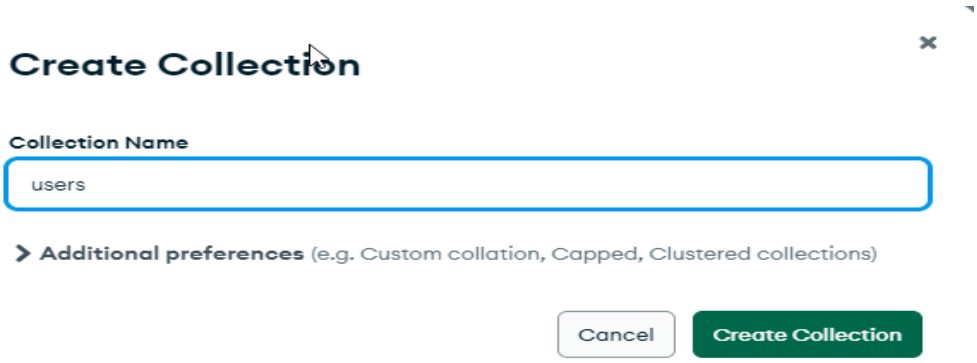
Εικόνα 56. Create Database



Εικόνα 57. Create Dabase & Collection

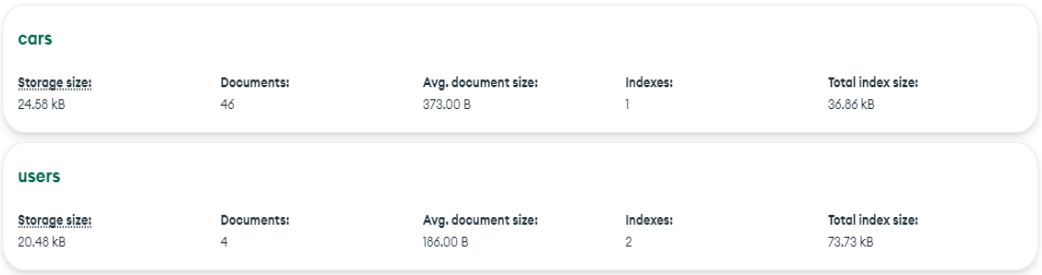
- Από την στιγμή που θα δημιουργήσει την βάση δεδομένων θα δημιουργήσει μία ακόμα collection με το όνομα users.

Εικόνα 58. Create collection



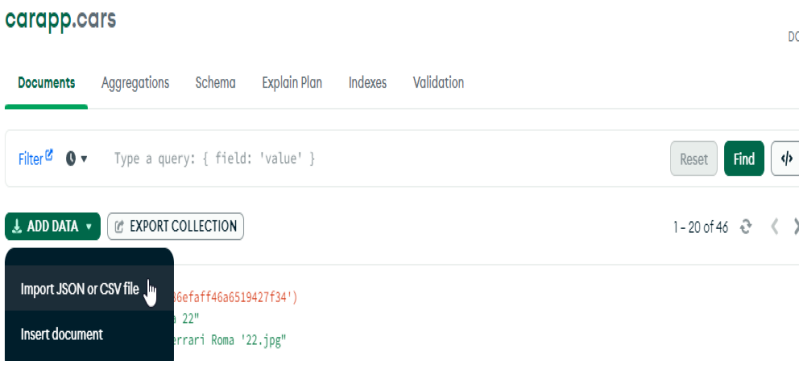
Εικόνα 59. Create Collection

- Στην συνέχεια το οι δύο collections που θα έχουν δημιουργηθεί θα είναι οι παρακάτω και ο χρήστης θα επιλέξει την collection cars.

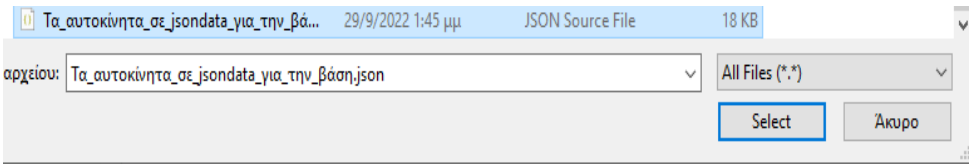


Εικόνα 60. Collection Cars & users

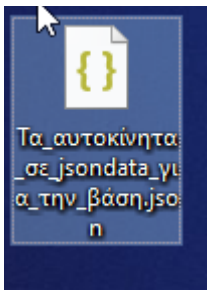
- Από την στιγμή που ο χρήστης θα μπει στην collection cars θα επιλέξει ADD DATA Import JSON or CSV file. Θα επιλέξει το αρχείο που θα βρίσκεται στον φάκελο 'Πτυχιακή-Εργασία' και στην συνέχεια τον φάκελο 'αρχείο για βάση δεδομένων' και από εκεί θα επιλέξει το αρχείο με το όνομα Τα_αυτοκίνητα_σε_jsondata_για_την_βάση.json.



Εικόνα 61. Data import

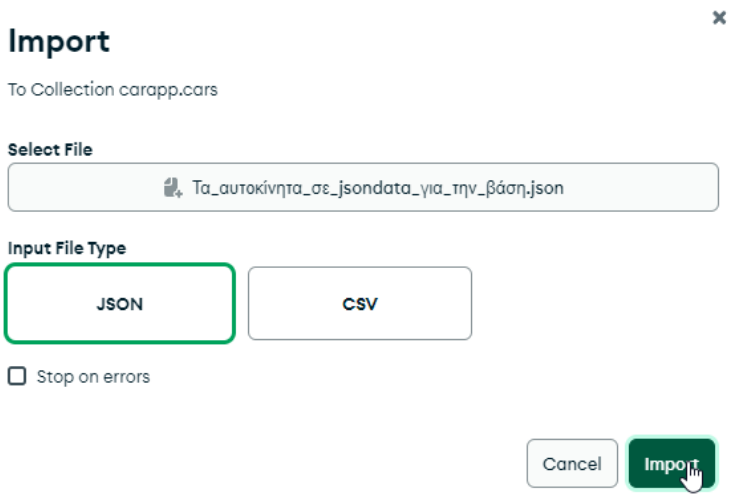


Εικόνα 62. Αρχείο Json



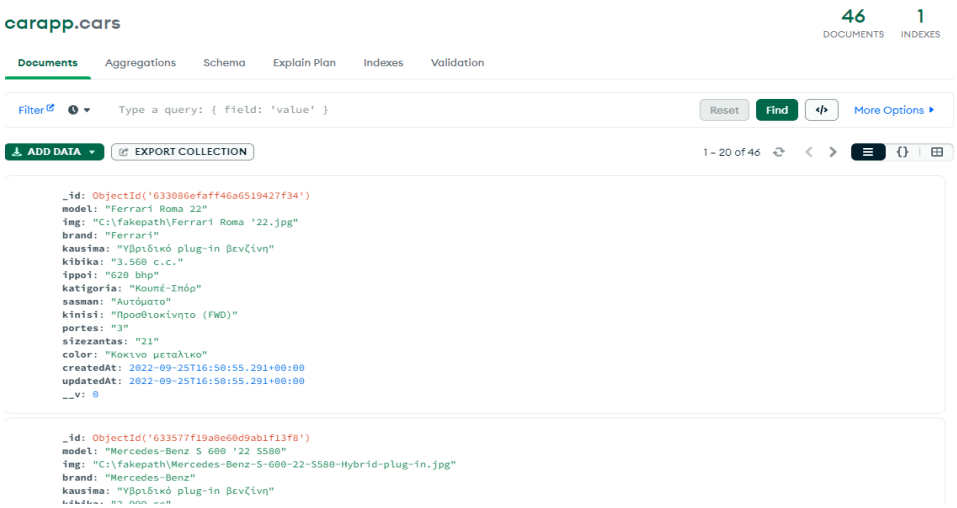
Εικόνα 63. Εικονίο αρχείου Json

- Εφόσον το επιλέξει θα πατήσει το JSON και θα κάνει import.



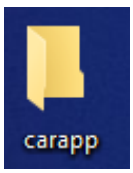
Εικόνα 64. Json import

- Τέλος θα εμφανιστούν τα όλα τα δεδομένα μέσα στην βάση δεδομένων.



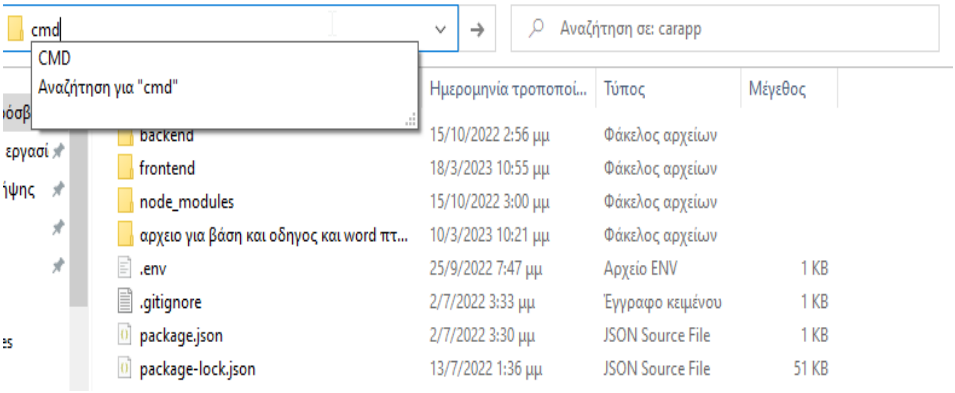
Εικόνα 65. Δεδομένα Βάσης Δεδομένων

- Στην συνέχεια ο χρήστης θα πάει στον φάκελο τις εφαρμογής και θα τον ανοίξει.

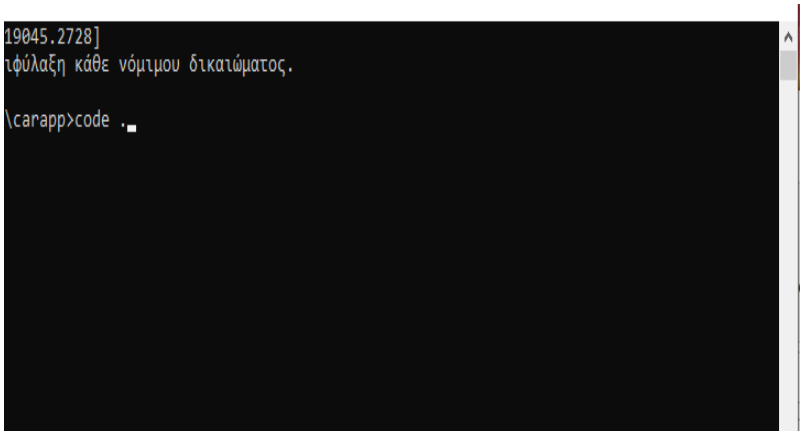


Εικόνα 66. Φακέλος Carapp

- Εφόσον ο χρήστης ανοίγει τον φάκελο στο σημείο δίπλα από την αναζήτηση γράφει cmd για να ανοίξει ένας τερματικός σταθμός και να γράψει code . για να ανοίξει το Visual Studio code με τον φάκελο που βρίσκεται το έργο.

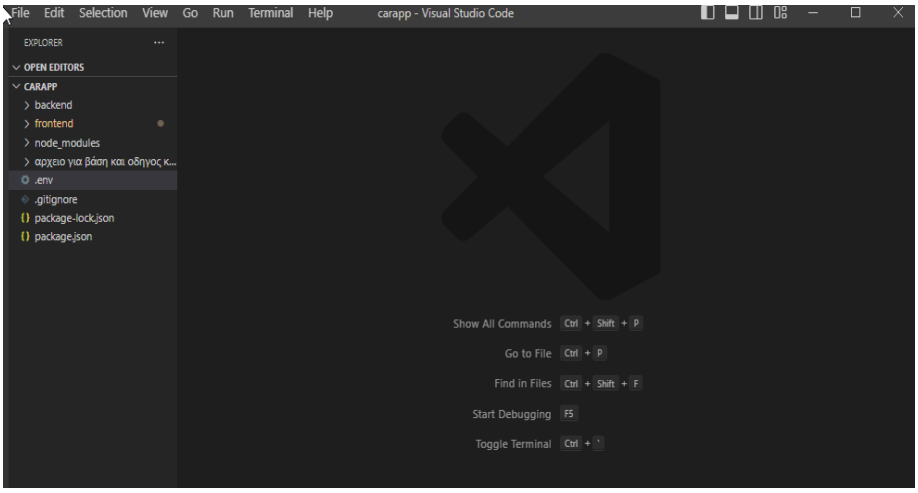


Εικόνα 67. Άνοιγμα terminal

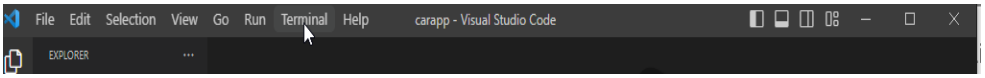


Εικόνα 68. Τερματικός (code .)

- Πλέον ο χρήστης βρίσκεται στον στο Visual Studio Code και θα ανοίξει έναν τερματικό.

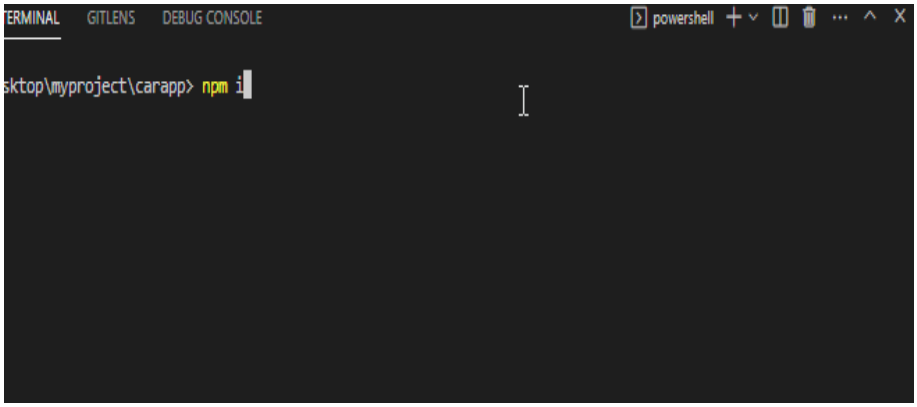


Εικόνα 69. Visual Studio Code



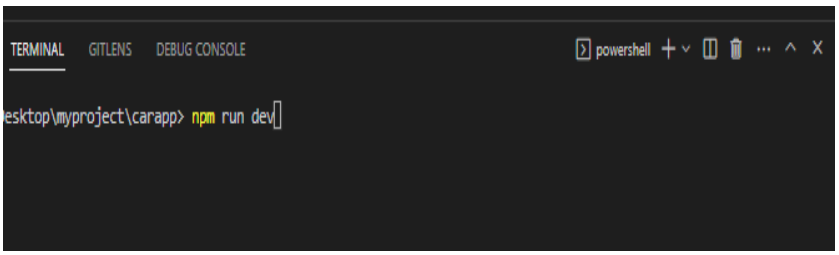
Εικόνα 70. New terminal

- Μόλις ανοίξει τον τερματικό θα πληκτρολογήσει npm i για να εγκατασταθούν στο συγκεκριμένο έργο όλες οι λειτουργίες που χρειάζεται για να τρέξει.



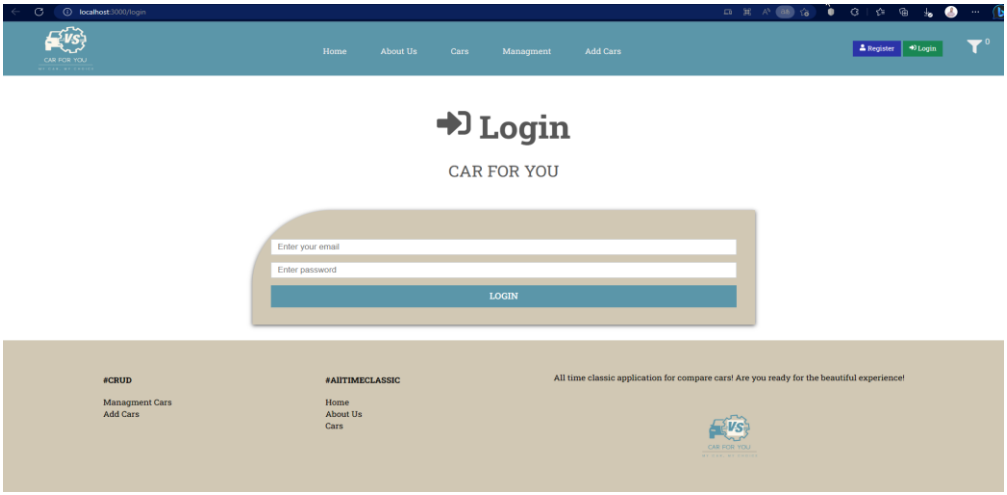
Εικόνα 71. Terminal npm i

- Στην συνέχεια θα ανοίξει έναν νέο τερματικό πατώντας το σύμβολο + δίπλα από την λέξη powershell και εφόσον ανοίξει ο καινούριος τερματικός πληκτρολογεί cd frontend και enter.
- Στην συνέχεια θα πληκτρολογήσει npm i.
- Τέλος, μόλις εγκατασταθούν όλα τα αρχεία και στο /carapp και στο carapp/frontend, τότε επιστρέφει στον προηγούμενο τερματικό και πληκτρολογεί npm run dev για να τρέξει η εφαρμογή.



Εικόνα 72. Npm run

Η εφαρμογή θα τρέξει στον κυρίαρχο browser που έχει ορίσει ο χρήστης και το αποτέλεσμα θα είναι το εξής.



Εικόνα 73. Log in

3.2 Παρουσίαση του έργου

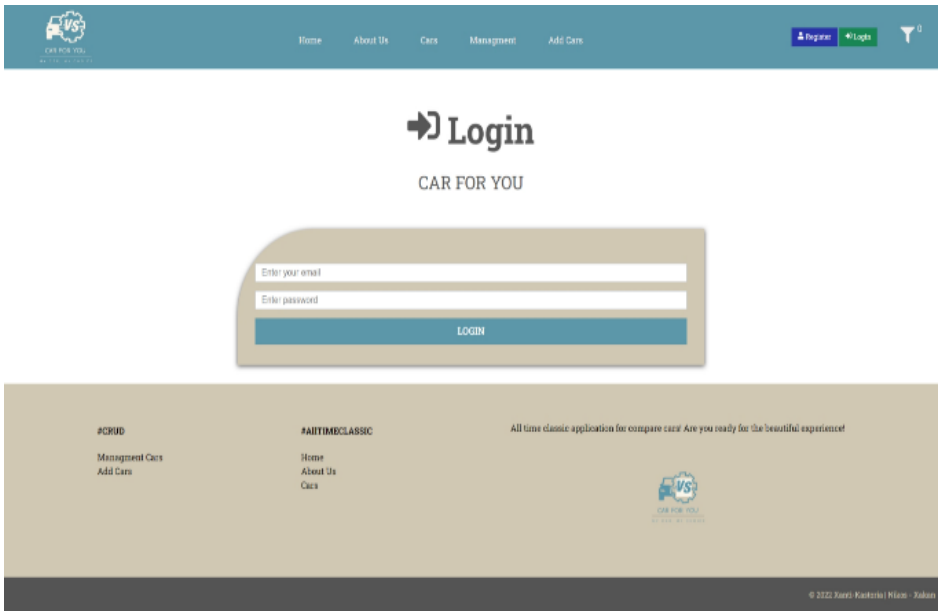
Μόλις ολοκληρωθεί η διαδικασία φόρτωσης της εφαρμογής θα εμφανιστεί στην σελίδα login.js με το url : http://localhost:3000/login (Car for You)

Από την σελίδα login οι μοναδικές σελίδες που μπορεί κάποιος να περιηγηθεί είναι η σελίδα register.js με url : http://localhost:3000/register. Αυτό συμβαίνει διότι στις υπόλοιπες σελίδες έχει προστεθεί ο κώδικας. Αν ο χρήστης δεν είναι συνδεδεμένος να μην μπορεί να περιηγηθεί στις υπόλοιπες σελίδες.

Ο κώδικας που δεν αφήνει τον χρήστη να περιηγηθεί στις υπόλοιπες σελίδες αν δεν είναι συνδεδεμένος με τα στοιχεία του. :

```
const navigate = useNavigate()

const { user } = useSelector((state) => state.auth)

useEffect(() => {

  if (!user) {

    navigate('/login')

  }
```

Login page

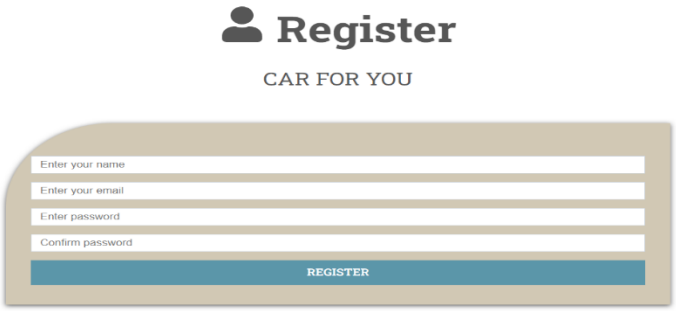


Εικόνα 74 . Login page

Η σελίδα σύνδεσης χρήστη (login page) είναι μια σελίδα που παρέχει στους χρήστες μια διαδικασία σύνδεσης. Η σελίδα σύνδεσης ζητά από τον χρήστη να εισάγει τα διαπιστευτήριά του, όπως το όνομα email και τον κωδικό πρόσβασης, προκειμένου να πιστοποιηθεί και να αποκτήσει πρόσβαση στην εφαρμογή.

Η σελίδα αυτή δεν έχει κάποια δυσκολία ως προς την χρήση της έτσι ώστε να κάνει τον χρήστη να μπορεί να προηγηθεί εύκολα χωρίς να δυσκολευτεί στην κατανόηση της.

Register page



Εικόνα 75 . Register page

Ο χρήστης σε αυτό το σημείο μπορεί να κάνει εγγραφή στην εφαρμογή ή να συνδεθεί με τον ήδη υπάρχων λογαριασμό που έχει δημιουργήσει στο παρελθόν. Μόλις ο χρήστης συνδεθεί με τα προσωπικά του στοιχεία θα τον μεταφέρει η εφαρμογή στην αρχική σελίδα.

Η σελίδα εγγραφής ζητά από τον χρήστη να εισάγει τα προσωπικά του στοιχεία, όπως το username, το email και έναν κωδικό πρόσβασης και έναν επιβεβαίωση του κωδικού πρόσβασης . Η σελίδα εγγραφής είναι εύκολη στη χρήση και παρέχει σαφείς οδηγίες για την είσοδο των προσωπικών δεδομένων του χρήστη. Επιπλέον, είναι ασφαλής και

προστατεύει τα δεδομένα του χρήστη από ανεπιθύμητη πρόσβαση. Και τέλος, ο χρήστης μπορεί να έχει πρόσβαση στην σελίδα εγγραφής με url : Car for You (http://localhost:3000/)

Εικόνα 76 . Home page

Η αρχική σελίδα, γνωστή και ως homepage, είναι η πρώτη σελίδα που βλέπει ο χρήστης όταν επισκέπτεται την ιστοσελίδα αυτή. Η homepage είναι σχεδιασμένη για να παρέχει γρήγορη και εύκολη πρόσβαση στις πλέον σημαντικές πληροφορίες και λειτουργίες της ιστοσελίδας.

Η homepage είναι απλή και εύκολη στη χρήση, ώστε οι χρήστες να μπορούν να βρουν τις πληροφορίες που χρειάζονται γρήγορα και χωρίς δυσκολία. Επιπλέον, είναι ελκυστική και προβάλει το brand της ιστοσελίδας, ώστε να ενθαρρύνει τους χρήστες να επιστρέψουν στη σελίδα.

Εικόνα 77 . Header

Εικόνα 78 . Footer

Σε αυτό το σημείο, ο χρήστης στο πάνω μέρος της σελίδας και στο κάτω μπορεί να παρατηρήσει διάφορες επιλογές όπως για να μεταφερθεί σε άλλη σελίδα της εφαρμογής ή για να αποσυνδεθεί.

Το header είναι η περιοχή στο πάνω μέρος μιας ιστοσελίδας, πάνω από το περιεχόμενο της σελίδας. Περιλαμβάνει το λογότυπο ιστοσελίδας, τη μπάρα πλοήγησης (menu) και άλλα στοιχεία που είναι χρήσιμα για το χρήστη όπως , το logout (αποσύνδεση).

Το footer είναι η περιοχή στο κάτω μέρος μιας ιστοσελίδας, κάτω από το περιεχόμενο της σελίδας. Περιλαμβάνει στοιχεία που σχετίζονται με την ιστοσελίδα, όπως σύνδεσμοι προς άλλες σελίδες της ιστοσελίδας καθώς και πληροφορίες σχετικά με την ιστοσελίδα όπως το copyright.

Στην συνέχεια μπορεί ο χρήστης να περιηγηθεί και στις υπόλοιπες σελίδες της εφαρμογής όπως.

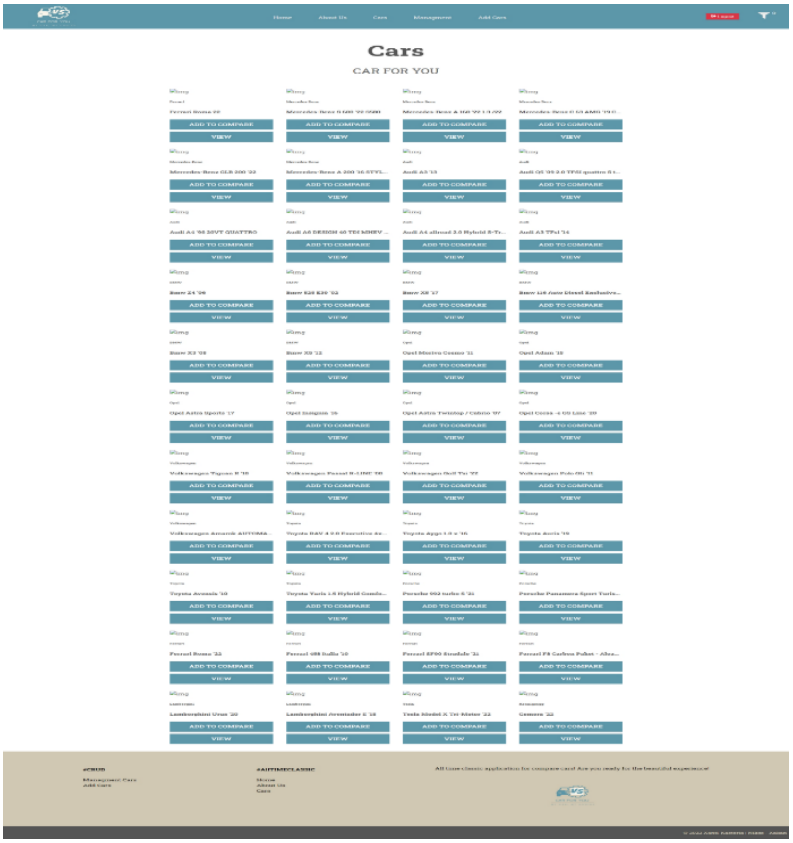
Την σελίδα About Us που σε αυτήν αναφέρονται λίγα λόγια για την εφαρμογή.

Η σελίδα "About Us" είναι μια σελίδα στην ιστοσελίδα μιας επιχείρησης ή ενός οργανισμού, που παρουσιάζει πληροφορίες σχετικά με την επιχείρηση, την ιστορία της, τις αξίες και τους στόχους της, το προσωπικό της και άλλα σχετικά θέματα.

Η σελίδα "About Us" αποτελεί σημαντικό στοιχείο της ιστοσελίδας και προσφέρει στους επισκέπτες της ιστοσελίδας μια βαθύτερη κατανόηση της εφαρμογής, των αξιών της. Περιλαμβάνει επίσης πληροφορίες για τα τις υπηρεσίες που προσφέρει η εφαρμογή.

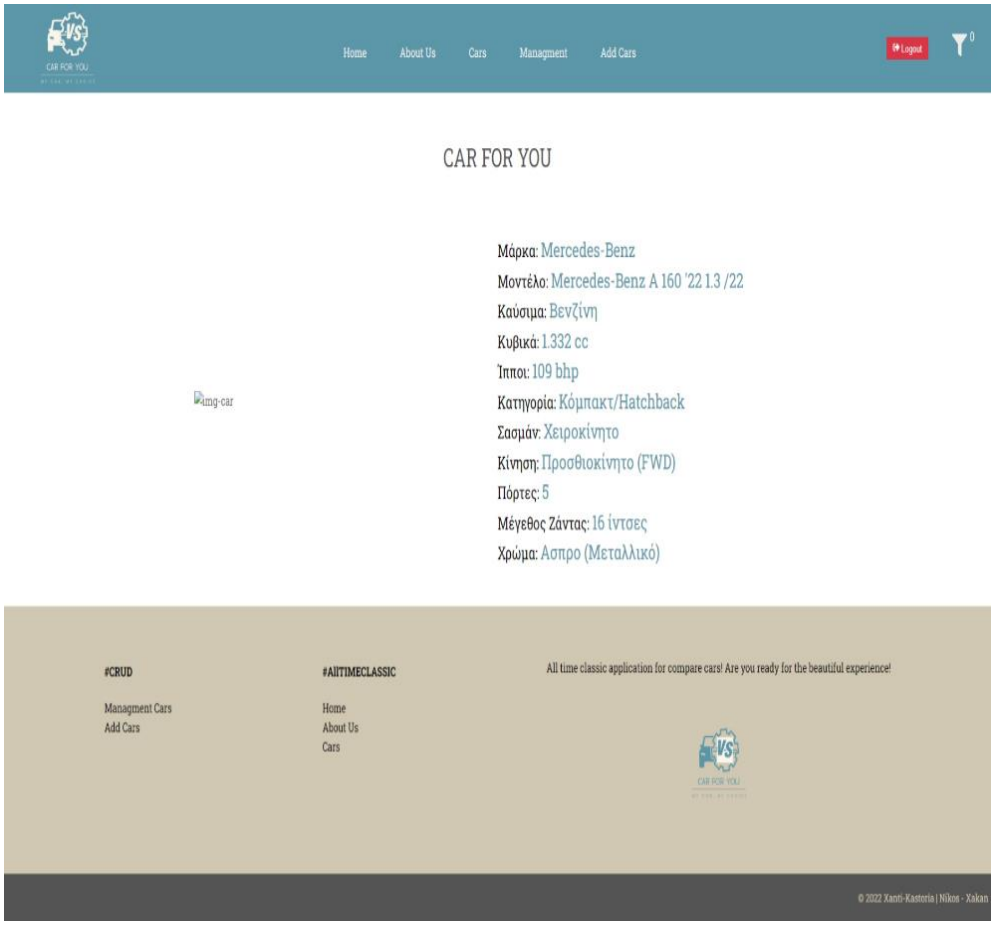
Εικόνα 79 . About us page

Την σελίδα που εμφανίζονται όλα τα αυτοκίνητα. Εδώ μπορεί ο χρήστης να δεί όλα τα αυτοκίνητα και εφόσον πατήσει το κουμπί compare να τα προσθέσει όποια αυτοκίνητα επιθυμεί στην λίστα σύγκρισης. Επίσης, μπορεί για να δει το κάθε αυτοκίνητο πιο αναλυτικά να μπει στο κάθε ένα ξεχωριστά. Ο χρήστης μπορεί να προηγηθεί σε αυτήν την σελίδα με url : Car for You (http://localhost:3000/cars)



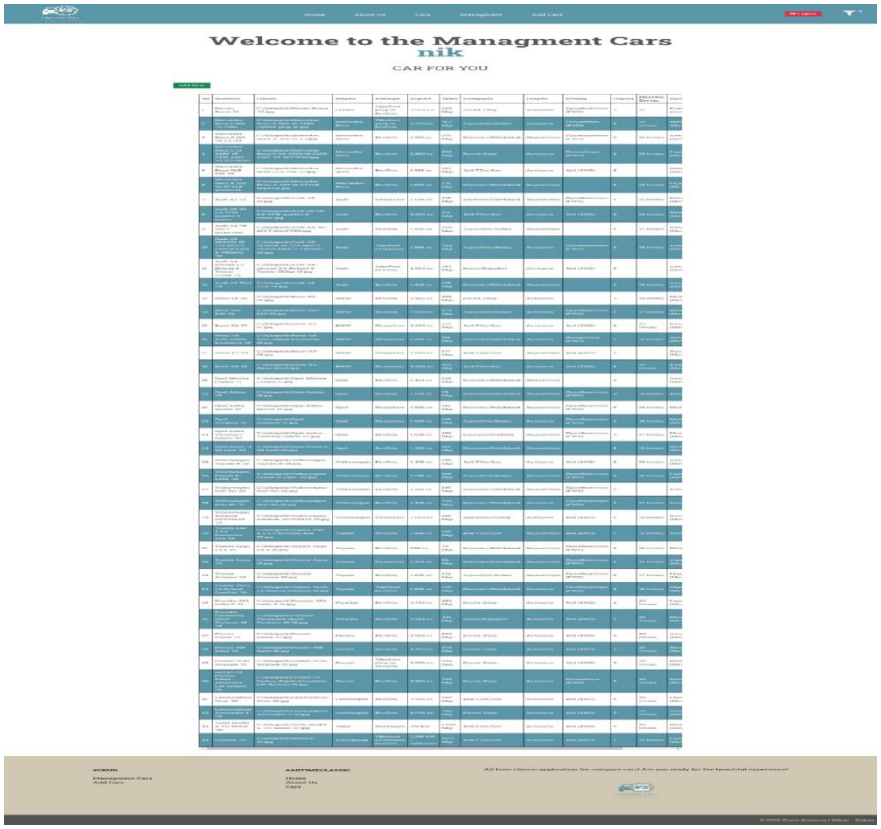
Εικόνα 80 . Cars page

Την σελίδα που εμφανίζεται το αυτοκίνητο και τα χαρακτηριστικά του είναι η σελίδα που ο χρήστης μπορεί να δει αναλυτικά όλες τις δυνατότητες του συγκεκριμένου αυτοκινήτου. Επίσης, αυτή η σελίδα είναι απλή στο design της έτσι ώστε να είναι εύχρηστη και κατανοητή κατά την περιήγηση του χρήστη. Ο χρήστης αν θέλει να περιηγηθεί σε αυτήν την σελίδα μπορεί με url : Car for You (http://localhost:3000/cars/633577f19a0e60d9ab1f13f9) < - - - Μετά το cars/ αναγράφεται το id του κάθε αυτοκινήτου που είναι μοναδικό.



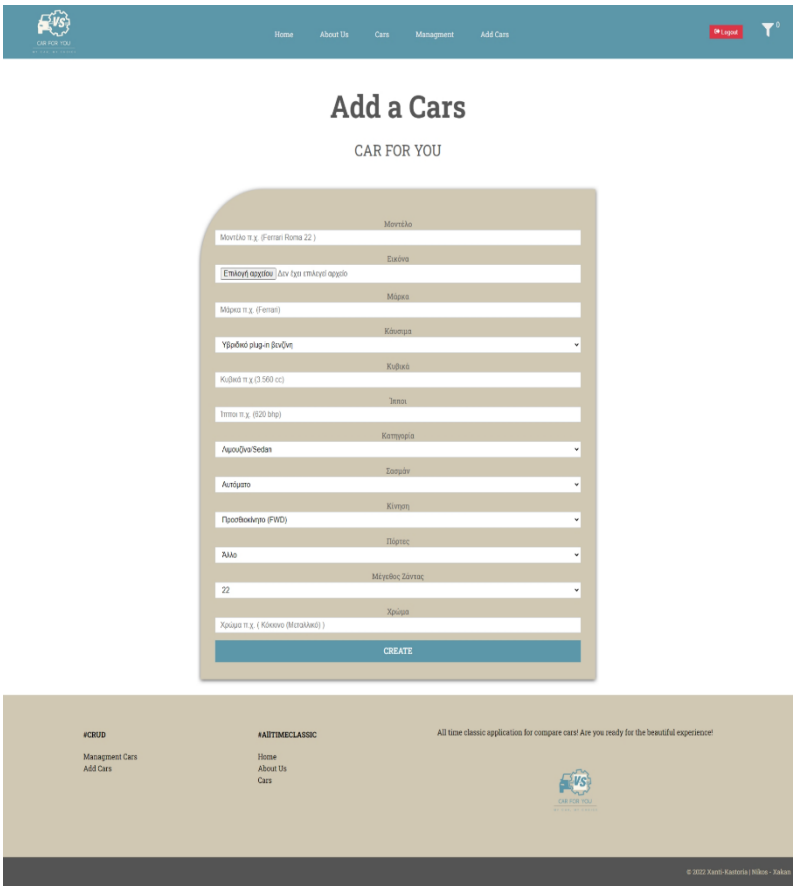
Εικόνα 81 . Car page

Εδώ φαίνεται η σελίδα διαχείρισης όλων των αυτοκινήτων της σελίδας. Εμφανίζει όλα τα αυτοκίνητα και τις πληροφορίες για το κάθε ένα ξεχωριστά. Μπορεί από αυτήν την σελίδα να διαγράψει οποιοδήποτε αυτοκίνητο επιθυμεί ο χρήστης, να κάνει αναβαθμίσει των πληροφοριών των αυτοκινήτων και ακόμα να μεταβεί στην σελίδα προσθήκης αυτοκίνητου για να προσθέσει ένα καινούριο αυτοκίνητο. Το url της σελίδας είναι: Car for You (http://localhost:3000/managment-cars).



Εικόνα 82 . Managment page

Την σελίδα που μπορεί ο χρήστης να προσθέσει ένα αυτοκίνητο. Στην συγκεκριμένη σελίδα μπορεί ο χρήστης να προσθέσει ένα καινούριο αυτοκίνητο βάζοντας τις πληροφορίες του εκάστοτε αυτοκινήτου. Αυτή η σελίδα βρίσκεται στο url : Car for You (http://localhost:3000/add-car).



Εικόνα 83 . Add car page

Τέλος την σελίδα της σύγκρισης αυτοκίνητων που εκεί μπορεί ο χρήστης να δει τις δυνατότητες του κάθε αυτοκινήτου σε σύγκρισή με κάποιο άλλο. Εδώ ο χρήστης μπορεί να επιλέξει όποιο αυτοκίνητο επιθυμεί και να το συγκρίνει με άλλα. Η σχεδίαση αυτής της σελίδας έγινε με τέτοιο τρόπο έτσι ώστε να είναι εύκολη η σύγκριση μεταξύ των αυτοκινήτων. Ο χρήστης μπορεί να περιηγηθεί σε αυτήν την σελίδα με url : Car for You (http://localhost:3000/compare-cars)

Συμπεράσματα

Η τεχνολογία έχει δεχθεί πολύ μεγάλη ανάπτυξη τα τελευταία χρόνια σε πολλούς κλάδους που ο καθένας βοηθάει με τον δικό του τρόπο. Συγκεκριμένα παρατηρείται πως οι διαδικτυακές εφαρμογές έχουν δεχθεί μεγάλη ανάπτυξη και αυτό οφείλετε στο γεγονός ότι πολλοί είναι αυτοί που τις χρησιμοποιούν στην καθημερινότητα τους π.χ. (εφαρμογή προσλήψεις τροφών ή νερού, εκμάθηση γλωσσών, προγράμματα γυμναστικής, ειδήσεις κλπ.). Όλες αυτές οι εφαρμογές με την σωστή χρήση μπορούν να βοηθήσουν σε πολλούς τομείς τους ανθρώπους. Επίσης, ένα πολύ καλό παράδειγμα για το πόσο βοήθησαν οι εφαρμογές στην εξοικονόμηση χρόνου είναι πως παλαιοτέρα έπρεπε να πάει κάποιος μέχρι την τράπεζα για να μεταφέρει χρήματα σε κάποιον, πλέον με το πάτημα ενός κουμπιού και από την άνεση του σπιτιού ή από οπουδήποτε μπορεί να μεταφέρει τα χρήματα. Η εξέλιξη είναι μεγάλη, τόση που έχει γίνει μέρος της καθημερινότητας και πως ακόμα και οι άνθρωποι μεγαλύτερης ηλικίας προσπαθούν να την εντάξουν στις ζωές τους.

Βιβλιογραφία

Erin Glass, 2020. *digitalocean.* [Ηλεκτρονικό]
Available at: https://www.digitalocean.com/community/tutorials/what-is-css
[Πρόσβαση 2023 Μάρτιος 2023].

Alexis Deveria, -. *caniuse.* [Ηλεκτρονικό]
Available at: https://.com/ciu/about
[Πρόσβαση 2023 Μάρτιος 2023].

Anon., χ.χ. *worldvectorlogo.* [Ηλεκτρονικό]
Available at: https://worldvectorlogo.com/

Anon., χ.χ. *worldvectorlogo.* [Ηλεκτρονικό]
Available at: https://worldvectorlogo.com/
[Πρόσβαση 29 Μάρτιος 2023].

Auth0, -. *auth0.* [Ηλεκτρονικό]
Available at: https://auth0.com/learn/json-web-tokens
[Πρόσβαση 29 Μάρτιος 2023].

Brian Boucheron, 2021. *digitalocean.* [Ηλεκτρονικό]
Available at: https://www.digitalocean.com/community/tutorials/what-is-node-js
[Πρόσβαση 29 Μάρτιος 2023].

businessinsider, 2020. *businessinsider.* [Ηλεκτρονικό]
Available at: https://www.businessinsider.com/guides/tech/what-is-canva
[Πρόσβαση 29 Μάρτιος 2023].

Chinmayee Deshpande, 2023. *simplilearn.* [Ηλεκτρονικό]
Available at: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs
[Πρόσβαση 29 Μάρτιος 2023].

Daragh Ó Tuama, 2022. *codeinstitute.* [Ηλεκτρονικό]
Available at: https://codeinstitute.net/nl/blog/mongodb-explained/
[Πρόσβαση 1 Μάρτιος 2023].

Jesse Hall, 2022. *mongodb.* [Ηλεκτρονικό]
Available at: https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/
[Πρόσβαση 29 Μάρτιος 2023].

Manjiri Gaikwad, 2022. *hevodata.* [Ηλεκτρονικό]
Available at: https://hevodata.com/learn/mongodb-compass/
[Πρόσβαση 29 Μάρτιος 2023].

Martin Heller, 2022. *infoworld.* [Ηλεκτρονικό]
Available at: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-

microsofts-extensible-code-editor.html
[Πρόσβαση 29 Μάρτιος 2023].

Matthew Tyson, 2022. *infoworld.* [Ηλεκτρονικό]
Available at: https://www.infoworld.com/article/3269878/what-is-an-api-application-programming-interfaces-explained.html
[Πρόσβαση 2023 Μάρτιος 2023].

MDN contributors, 2023. *developer.mozilla.org.* [Ηλεκτρονικό]
Available at: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design
[Πρόσβαση 29 Μάρτιος 2023].

MDN contributors, 2023. *developer.mozilla.org.* [Ηλεκτρονικό]
Available at: https://developer.mozilla.org/en-US/docs/Web/HTML
[Πρόσβαση 29 Μάρτιος 2023].

Mihir Patkar, 2014. *lifehacker.* [Ηλεκτρονικό]
Available at: https://lifehacker.com/compressor-io-shrinks-image-file-size-without-quality-l-1581592588
[Πρόσβαση 2023 Μάρτιος 2023].

Oleg Kopachovets, 2022. *procoders.* [Ηλεκτρονικό]
Available at: https://procoders.tech/blog/express-js-vs-node-js/
[Πρόσβαση 29 Μάρτιος 2023].

postman, -. *postman.* [Ηλεκτρονικό]
Available at: https://www.postman.com/
[Πρόσβαση 29 Μάρτιος 2023].

Pratik Das, 2022. *reflectoring.* [Ηλεκτρονικό]
Available at: https://reflectoring.io/tutorial-guide-axios/
[Πρόσβαση 29 Μάρτιος 2023].

Ravikiran A S, 2023. *simplilearn.* [Ηλεκτρονικό]
Available at: https://www.simplilearn.com/tutorials/reactjs-tutorial/react-with-redux#what_is_redux
[Πρόσβαση 29 Μάρτιος 2023].

Παράρτημα Κώδικα

```
<!-- backend -->

<!-- server.js -->

const path = require('path');

const express = require('express');

const cors = require('cors')

const colors = require('colors');

const dotenv = require('dotenv').config();

const { errorHandler } = require('./middleware/errorMiddleware');

const connectDB = require('./config/db');

const port = process.env.PORT || 5000;

connectDB();

const app = express();

app.use(cors())

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use('/api/users', require('./routes/userRoutes'));

app.use('/api/cars',  require('./routes/carRoutes'))

// Serve frontend

if (process.env.NODE_ENV === 'production') {

  app.use(express.static(path.join(__dirname, '../frontend/build')));

  app.get('*', (req, res) =>

    res.sendFile(

      path.resolve(__dirname, '../', 'frontend', 'build', 'index.html')

    )

  );
```

```
} else {
  app.get('/', (req, res) => res.send('Please set to production'));
}
app.use(errorHandler);
app.listen(port, () => console.log(`Server started on port ${port}`));
<!-- routes -->
<!-- carRoutes.js -->
const express = require('express')
const router = express.Router()
const { getCar, getCarById, postCar, putCar, deleteCar } =
require('../controllers/carController')
router.route('/').get(getCar).post(postCar)
router.route('/:id').delete(deleteCar).put(putCar).get(getCarById)
module.exports = router
<!-- userRoutes.js -->
const express = require('express')
const router = express.Router()
const {
  registerUser,
  loginUser,
  getMe,
} = require('../controllers/userController')
const { protect } = require('../middleware/authMiddleware')
router.post('/', registerUser)
router.post('/login', loginUser)
router.get('/me', protect, getMe)
module.exports = router
```

```
<!-- models -->
<!-- carModel.js -->
const mongoose = require('mongoose')
const carSchema = mongoose.Schema(
  {
    model: {
      type: String,
      required: true
    },
    img: {
      type: String,
      required: true
    },
    brand: {
      type: String,
      required: true
    },
    kausima: {
      type: String,
      required: true
    },
    kibika: {
      type: String,
      required: true
    },
    ippoi: {
```

```
  type: String,
  required: true
},
katigoria: {
  type: String,
  required: true
},
sasman: {
  type: String,
  required: true
},
kinisi: {
  type: String,
  required: true
},
portes: {
  type: String,
  required: true
},
sizezantas: {
  type: String,
  required: true
},
color: {
  type: String,
  required: true
},
```

```
  },
  {
    timestamps: true,
  }
)
module.exports = mongoose.model('Car', carSchema)
<!-- userModel.js -->
const mongoose = require('mongoose')
const userSchema = mongoose.Schema(
  {
    name: {
      type: String,
      required: [true, 'Please add a name'],
    },
    email: {
      type: String,
      required: [true, 'Please add an email'],
      unique: true,
    },
    password: {
      type: String,
      required: [true, 'Please add a password'],
    },
  },
  {
    timestamps: true,
```

```
  }
)
module.exports = mongoose.model('User', userSchema)

<!-- middleware -->
<!-- authMiddleware.js -->
const jwt = require('jsonwebtoken')
const asyncHandler = require('express-async-handler')
const User = require('../models/userModel')
const protect = asyncHandler(async (req, res, next) => {
  let token
  if (
    req.headers.authorization &&
    req.headers.authorization.startsWith('Bearer')
  ) {
    try {
      // Get token from header
      token = req.headers.authorization.split(' ')[1]
      // Verify token
      const decoded = jwt.verify(token, process.env.JWT_SECRET)
      // Get user from the token
      req.user = await User.findById(decoded.id).select('-password')
      next()
    } catch (error) {
      console.log(error)
      res.status(401)
      throw new Error('Not authorized')
```

```
    }
  }
  if (!token) {
    res.status(401)
    throw new Error('Not authorized, no token')
  }
})
module.exports = { protect }
<!-- errorMiddleware.js -->
const errorHandler = (err, req, res, next) => {
    const statusCode = res.statusCode ? res.statusCode : 500
    res.status(statusCode)
    res.json({
      message: err.message,
      stack: process.env.NODE_ENV === 'production' ? null : err.stack,
    })
  }
  module.exports = {
    errorHandler,
  }
<!-- controllers -->
<!-- carController.js -->
const asyncHandler = require('express-async-handler')
const Car = require('../models/carModel')
// @desc Get cars
// route Get /api/cars
// @access Private
```

```
const getCar = asyncHandler( async (req, res) => {
    const cars = await Car.find()
        res.status(200).json(cars)
    }
)
// @desc Get cars
// route Get /api/cars/:id
// @access Private
const getCarById = asyncHandler( async (req, res) => {
    const car = await Car.findById(req.params.id)
    if(!car) {
        res.status(400)
        throw new Error('Car not found')
    }
    res.status(200).json(car)
}
)
// @desc Create cars
// route POST /api/cars
// @access Private
const postCar = asyncHandler( async (req, res) => {
        if(!req.body
            // .text
            ) {
            res.status(400)
            throw new Error('Please add a text field')
        }
```

```
        const car = await Car.create({
            model: req.body.model,
            img: req.body.img,
            brand: req.body.brand,
            kausima: req.body.kausima,
            kibika: req.body.kibika,
            ippoi: req.body.ippoi,
            katigoria: req.body.katigoria,
            sasman: req.body.sasman,
            kinisi: req.body.kinisi,
            portes: req.body.portes,
            sizezantas: req.body.sizezantas,
            color: req.body.color,
        })
        res.status(200).json(car)
    }
)
// @desc Update cars
// route PUT /api/cars/:id
// @access Private
const putCar = asyncHandler( async (req, res) => {
    const car = await Car.findById(req.params.id)
    if(!car) {
        res.status(400)
        throw new Error('Car not found')
    }
    const updateCar = await Car.findByIdAndUpdate(req.params.id, req.body, {
```

```
        new: true,
    })
        res.status(200).json(updateCar)
    }
)
// @desc Delete cars
// route Delete /api/cars/:id
// @access Private
const deleteCar = asyncHandler( async (req, res) => {
    const car = await Car.findById(req.params.id)
    if(!car) {
        res.status(400)
        throw new Error('Car not found')
    }
    await car.remove()
        res.status(200).json({ id: req.params.id })
    }
)
module.exports = {
    getCar,
    getCarById,
    postCar,
    putCar,
    deleteCar
}
<!-- userController.js -->
const jwt = require('jsonwebtoken')
```

```
const bcrypt = require('bcryptjs')
const asyncHandler = require('express-async-handler')
const User = require('../models/userModel')
// @desc    Register new user
// @route   POST /api/users
// @access  Public
const registerUser = asyncHandler(async (req, res) => {
  const { name, email, password } = req.body
  if (!name || !email || !password) {
    res.status(400)
    throw new Error('Please add all fields')
  }
  // Check if user exists
  const userExists = await User.findOne({ email })
  if (userExists) {
    res.status(400)
    throw new Error('User already exists')
  }
  // Hash password
  const salt = await bcrypt.genSalt(10)
  const hashedPassword = await bcrypt.hash(password, salt)
  // Create user
  const user = await User.create({
    name,
    email,
    password: hashedPassword,
  })
```

```
  if (user) {
    res.status(201).json({
      _id: user.id,
      name: user.name,
      email: user.email,
      token: generateToken(user._id),
    })
  } else {
    res.status(400)
    throw new Error('Invalid user data')
  }
})
// @desc    Authenticate a user
// @route   POST /api/users/login
// @access  Public
const loginUser = asyncHandler(async (req, res) => {
  const { email, password } = req.body
  // Check for user email
  const user = await User.findOne({ email })
  if (user && (await bcrypt.compare(password, user.password))) {
    res.json({
      _id: user.id,
      name: user.name,
      email: user.email,
      token: generateToken(user._id),
    })
```

```
  } else {
    res.status(400)
    throw new Error('Invalid credentials')
  }
})
// @desc    Get user data
// @route   GET /api/users/me
// @access  Private
const getMe = asyncHandler(async (req, res) => {
  res.status(200).json(req.user)
})
// Generate JWT
const generateToken = (id) => {
  return jwt.sign({ id }, process.env.JWT_SECRET, {
    expiresIn: '30d',
  })
}
module.exports = {
  registerUser,
  loginUser,
  getMe,
}
<!-- config -->
<!-- db.js -->
const mongoose = require('mongoose')
const connectDB = async () => {
  try {
```

```
    const conn = await mongoose.connect(process.env.MONGO_URI)

    console.log(`MongoDB Connected: ${conn.connection.host}`.cyan.underline)
  } catch (error) {
    console.log(error)
    process.exit(1)
  }
}
module.exports = connectDB
<!-- frontend -->
<!-- src -->
<!-- app -->
<!-- store.js -->
import { configureStore } from '@reduxjs/toolkit'
import authReducer from '../features/auth/authSlice'
export const store = configureStore({
  reducer: {
    auth: authReducer,
  },
})
<!-- components -->
<!-- About.js -->
import React from 'react'
import { Navigate, useNavigate } from "react-router-dom";
import "../styles/about-us.css";
import image from "../assets/vintage-car.jpg"
// import login user
```

```
import { useSelector } from 'react-redux'
import { useEffect } from 'react'
function AboutUs() {
   const navigate = useNavigate()
   const { user } = useSelector((state) => state.auth)
   useEffect(() => {
    if (!user) {
      navigate('/login')
    }
   }, [user, navigate])
  return (
   <div className='container'>
    <div className='heading'>
       <h1>About Us</h1>
        <p>CAR FOR YOU</p>
    </div>
    <div className='section'>
       <div className='about-content '>
        <div className='image-wrapper'>
         <img src={image} alt='About Us' />
        </div>
        <div>
         <h2>#Carsforyou</h2>
         <p>
```

Τα τελευταία χρόνια η πληροφορική έχει εισχωρήσει σε πολλούς κλάδους και ένας από αυτούς είναι οι διαδικτυακές εφαρμογές.

Η ανάλυση των δυνατοτήτων δύο ή περισσότερων αυτοκινήτων είναι μία από αυτές τις εφαρμογές. Γενικότερα ο χρήστης με ένα μόνο κουμπί

μπορεί να συγκρίνει αυτοκίνητα, είτε για να λάβει πληροφορίες είτε για να αποκτήσει μία γνώση που μπορεί να τον βοηθήσει για το

μελλοντικό του αυτοκίνητο. Επίσης έχουμε δημιουργήσει ένα σύστημα διαχείρισής των χρηστών έτσι ώστε ο κάθε χρήστης να συνδέεται με

τα προσωπικά του στοιχεία και όλο αυτό με την βοήθεια μίας βάσης δεδομένων. Έχει παρατηρηθεί πως ο κόσμος ολοένα και εμπιστεύεται τέτοιου

είδους εφαρμογές γιατί κάνουν τους χρήστες να έχουν μία άμεση επαφή με την πληροφορία.

Οι εφαρμογές εξελίσσονται με ιλιγγιώδης ρυθμούς και οι ο ανταγωνισμός που υπάρχει βοηθάει στο να έχουμε καθημερινά πολύ

δημιουργικά και χρήσιμα εργαλεία.

```
            </p>
          </div>
        </div>
      </div>
    </div>
  )
}
export default AboutUs
<!-- AddCar.js -->
import React, { useState } from "react";
import axios from "axios";
import { Navigate, useNavigate } from "react-router-dom";
// import login user
import { useSelector } from 'react-redux'
import { useEffect } from 'react'
const AddCar = () => {
   const [model, setModel] = useState("")
   const [img, setImg] = useState("")
```

```
const [brand, setBrand] = useState("")
const [kausima, setKausima] = useState("Υβριδικό plug-in βενζίνη")
const [kibika, setKibika] = useState("")
const [ippoi, setIppoi] = useState("")
const [katigoria, setKatigoria] = useState("Λιμουζίνα/Sedan")
const [sasman, setSazman] = useState("Αυτόματο")
const [kinisi, setKinisi] = useState("Προσθιοκίνητο (FWD)")
const [portes, setPortes] = useState(" 7")
const [sizezantas, setSizezantas] = useState("22")
const [color, setColor] = useState("")
const navigate = useNavigate()
const postCar = async (e) => {
   e.preventDefault();
   try {
      await axios.post('http://localhost:5000/api/cars', {
         model,
         img,
         brand,
         kausima,
         kibika,
         ippoi,
         katigoria,
         sasman,
         kinisi,
         portes,
         sizezantas,
         color,
```

```
        })
        navigate("/")
      } catch (error) {
        console.log(error)
      }
  }
const { user } = useSelector((state) => state.auth)
useEffect(() => {
  if (!user) {
    navigate('/login')
  }
}, [user, navigate])
  return (
    <div className="grid">
      <div className='heading'>
        <h1>Add a Cars</h1>
        <p>CAR FOR YOU</p>
      </div>
      <div className="section">
        <div className='columns'>
          <div className='form'>
            <br />
            <form onSubmit={postCar}>
              <div className='field'>
                <label className='label'>Μοντέλο</label>
                <div className='control'>
                  <input
```

```
            type="text"
            className='input'
            value={model}
            onChange={(e) => setModel(e.target.value)}
            placeholder='Μοντέλο π.χ. (Ferrari Roma 22 )'
          />
        </div>
      </div>
      <div className='field'>
        <label className='label'>Εικόνα</label>
        <div className='control'>
          <input
            type="file"
            className='input'
            value={img}
            onChange={(e) => setImg(e.target.value)}
            placeholder='Εικόνα'
          />
        </div>
      </div>
      <div className='field'>
        <label className='label'>Μάρκα</label>
        <div className='control'>
          <input
            type="text"
            className='input'
            value={brand}
```

```
                        onChange={(e) => setBrand(e.target.value)}
                        placeholder='Μάρκα π.χ. (Ferrari)'
                      />
                    </div>
                  </div>
                  <div className='field'>
                    <label className='label'>Κάυσιμα</label>
                    <div className='control'>
                      <div className='select is-fullwidth'>
                        <select
                          value={kausima}
                          onChange={(e) => setKausima(e.target.value)}
                        >
                          <option value="Άλλο">Άλλο</option>
                          <option value="Υβριδικό plug-in βενζίνη">Υβριδικό
plug-in βενζίνη</option>
                          <option value="Βενζίνη">Βενζίνη</option>
                          <option value="Πετρέλαιο">Πετρέλαιο</option>
                          <option value="Υβριδικό πετρέλαιο">Υβριδικό
πετρέλαιο</option>
                          <option value="Υβριδικό βενζίνη">Υβριδικό
βενζίνη</option>
                          <option value="Ηλεκτρικό">Ηλεκτρικό</option>
                        </select>
                      </div>
                    </div>
                  </div>
                  <div className='field'>
```

```
    <label className='label'>Κυβικά</label>
    <div className='control'>
      <input
        type="text"
        className='input'
        value={kibika}
        onChange={(e) => setKibika(e.target.value)}
        placeholder='Κυβικά π.χ.(3.560 cc)'
      />
    </div>
  </div>
  <div className='field'>
    <label className='label'>Ίπποι</label>
    <div className='control'>
      <input
        type="text"
        className='input'
        value={ippoi}
        onChange={(e) => setIppoi(e.target.value)}
        placeholder='Ίπποι π.χ. (620 bhp)'
      />
    </div>
  </div>
  <div className='field'>
    <label className='label'>Κατηγορία</label>
    <div className='control'>
      <div className='select is-fullwidth'>
```

```
                                <select
                                    value={katigoria}
                                    onChange={(e) => setKatigoria(e.target.value)}
                                >
                                    <option value="Άλλο">Άλλο</option>
                                    <option
value="Λιμουζίνα/Sedan">Λιμουζίνα/Sedan</option>
                                    <option
value="Κόμπακτ/Hatchback">Κόμπακτ/Hatchback</option>
                                    <option value="Κουπέ-Σπόρ">Κουπέ-Σπόρ</option>
                                    <option value="4χ4/Τζιπ/Suv">4χ4/Τζιπ/Suv</option>
                                    <option
value="Κόμπι/Κάραβαν">Κόμπι/Κάραβαν</option>
                                    <option
value="Κάμπριο/Roadster">Κάμπριο/Roadster</option>
                                    <option
value="Αγροτικό/Pickup">Αγροτικό/Pickup</option>
                                </select>
                            </div>
                        </div>
                    </div>
                    <div className='field'>
                        <label className='label'>Σασμάν</label>
                        <div className='control'>
                            <div className='select is-fullwidth'>
                                <select
                                    value={sasman}
                                    onChange={(e) => setSazman(e.target.value)}
                                >
```

```
                        <option value="Άλλο">Άλλο</option>
                        <option value="Αυτόματο">Αυτόματο</option>
                        <option value="Χειροκίνητο">Χειροκίνητο</option>
                    </select>
                </div>
            </div>
        </div>
        <div className='field'>
            <label className='label'>Κίνηση</label>
            <div className='control'>
                <div className='select is-fullwidth'>
                    <select
                        value={kinisi}
                        onChange={(e) => setKinisi(e.target.value)}
                    >
                        <option value="Άλλο">Άλλο</option>
                        <option value="Πισωκίνητο (RWD)">Πισωκίνητο
(RWD)</option>
                        <option value="Προσθιοκίνητο (FWD)">Προσθιοκίνητο
(FWD)</option>
                        <option value="4x4 (4WD)">4x4 (4WD)</option>
                    </select>
                </div>
            </div>
        </div>
        <div className='field'>
            <label className='label'>Πόρτες</label>
            <div className='control'>
```

```
        <div className='select is-fullwidth'>
          <select
            value={portes}
            onChange={(e) => setPortes(e.target.value)}
          >
            <option value="Άλλο">Άλλο</option>
            <option value="7">7</option>
            <option value="6">6</option>
            <option value="5">5</option>
            <option value="4">4</option>
            <option value="3">3</option>
            <option value="2">2</option>
          </select>
        </div>
      </div>
    </div>
    <div className='field'>
      <label className='label'>Μέγεθος Ζάντας</label>
      <div className='control'>
        <div className='select is-fullwidth'>
          <select
            value={sizezantas}
            onChange={(e) => setSizezantas(e.target.value)}
          >
            <option value="Άλλο">Άλλο</option>
            <option value="21F/22R">21F/22R</option>
            <option value="22">22</option>
```

```
                <option value="21">21</option>
                <option value="20">20</option>
                <option value="19">19</option>
                <option value="18">18</option>
                <option value="17">17</option>
                <option value="16">16</option>
                <option value="15">15</option>
                <option value="14">14</option>
            </select>
        </div>
    </div>
</div>
<div className='field'>
    <label className='label'>Χρώμα</label>
    <div className='control'>
        <input
            type="text"
            className='input'
            value={color}
            onChange={(e) => setColor(e.target.value)}
            placeholder='Χρώμα π.χ. ( Κόκκινο (Μεταλλικό) )'
        />
    </div>
</div>
<div className='field'>
    <div className='control'>
```

```
                                <button type="submit" className='btn btn-
cars'>Create</button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
  )
}
export default AddCar
<!-- Car.js -->
import React, { useState, useEffect } from "react";
import axios from "axios";
import { useParams } from "react-router-dom";
import "../styles/car.css";
// import login user
import { useNavigate } from 'react-router-dom'
import { useSelector } from 'react-redux'
const Car = () => {
  const [model, setModel] = useState("")
  const [img, setImg] = useState("")
  const [brand, setBrand] = useState("")
  const [kausima, setKausima] = useState("Υβριδικό plug-in βενζίνη")
  const [kibika, setKibika] = useState("")
  const [ippoi, setIppoi] = useState("")
```

```
  const [katigoria, setKatigoria] = useState("Λιμουζίνα/Sedan")
  const [sasman, setSazman] = useState("Αυτόματο")
  const [kinisi, setKinisi] = useState("Προσθιοκίνητο (FWD)")
  const [portes, setPortes] = useState(" 7")
  const [sizezantas, setSizezantas] = useState("22")
  const [color, setColor] = useState("")
  const { id } = useParams()

  useEffect(() => {
     getCarById();
  }, []);
  const getCarById = async () => {
     const response = await axios.get(`http://localhost:5000/api/cars/${id}`);
     setModel(response.data.model);
     setImg(response.data.img);
     setBrand(response.data.brand);
     setKausima(response.data.kausima);
     setKibika(response.data.kibika);
     setIppoi(response.data.ippoi);
     setKatigoria(response.data.katigoria);
     setSazman(response.data.sasman);
     setKinisi(response.data.kinisi);
     setPortes(response.data.portes);
     setSizezantas(response.data.sizezantas);
     setColor(response.data.color);
  };
const navigate = useNavigate()
```

```
const { user } = useSelector((state) => state.auth)
useEffect(() => {
  if (!user) {
    navigate('/login')
  }
}, [user, navigate])
  return (
      <div className="container">
          <div className='heading'>
              <p>CAR FOR YOU</p>
          </div>
          <style>
              .item-car .content-car ul li{}
          </style>
          <div className="section">
              <div className="item-car">
                  <div className="img-car">
                      <img className="item-car-img" src={ img } alt='img-car' />
                  </div>
                  <div className="content-car">
                        <ul>
                            <li><span>Μάρκα:<span>  </span></span>{brand}</li>
                            <li><span>Μοντέλο:<span>  </span></span>{model}</li>
                            <li><span>Καύσιμα:<span>  </span></span>{kausima}</li>
                            <li><span>Κυβικά:<span>  </span></span>{kibika}</li>
                            <li><span>Ίπποι:<span>  </span></span>{ippoi}</li>
                            <li><span>Κατηγορία:<span>  </span></span>{katigoria}</li>
```

```
                        <li><span>Σασμάν:<span>  </span></span>{sasman}</li>
                        <li><span>Κίνηση:<span>  </span></span>{kinisi}</li>
                        <li><span>Πόρτες:<span>  </span></span>{portes}</li>
                        <li><span>Μέγεθος Ζάντας:<span>
</span></span>{sizezantas}</li>
                        <li><span>Χρώμα:<span>  </span></span>{color}</li>
                    </ul>
                </div>
            </div>
        </div>
    </div>
  )
}
export default Car
<!-- CarList.js -->
import React, { useState, useEffect } from "react";
import axios from "axios";
import { Link } from "react-router-dom"
// import login user
import { useNavigate } from 'react-router-dom'
import { useSelector } from 'react-redux'
const UserList = () => {
  const [cars, setCar] = useState([]);
  useEffect(() => {
    getCar();
  }, []);
  const getCar = async () => {
```

```
        const response = await axios.get('http://localhost:5000/api/cars/');
        setCar(response.data)
      }
      const deleteCar = async (id) => {
        try {
          await axios.delete(`http://localhost:5000/api/cars/${id}`);
          getCar();
        } catch (error) {
          console.log('error');
        }
      }
    const navigate = useNavigate()
    const { user } = useSelector((state) => state.auth)
    useEffect(() => {
      if (!user) {
        navigate('/login')
      }
    }, [user, navigate])
      return (
        <div className="container">
          <div className='heading'>
            <h1>Welcome to the Managment Cars <span>{user && user.name}</span></h1>
            <p>CAR FOR YOU</p>
          </div>
          <Link to="/add-car" className="btn btn-success">Add New</Link>
          <div className='columns'>
```

```
<div className="responsive-table">
    <table className='table table-bordered'>
        <thead>
            <tr>
                <th>No</th>
                <th>Μοντέλο</th>
                <th>Είκονα</th>
                <th>Μάρκα</th>
                <th>Καύσιμα </th>
                <th>Κυβικά </th>
                <th>Ίπποι</th>
                <th>Κατηγορία </th>
                <th>Σασμάν </th>
                <th>Κίνηση </th>
                <th>Πόρτες </th>
                <th>Μέγεθος Ζάντας </th>
                <th>Χρώμα </th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            {cars.map((car, index) => (
                <tr key={car._id}>
                    <td>{index + 1}</td>
                    <td>{car.model}</td>
                    <td>
                        {/* <img typeof="image/jpg" src={car.img} /> */}
```

```
                                {car.img}
                            </td>
                            <td>{car.brand}</td>
                            <td>{car.kausima}</td>
                            <td>{car.kibika}</td>
                            <td>{car.ippoi}</td>
                            <td>{car.katigoria}</td>
                            <td>{car.sasman}</td>
                            <td>{car.kinisi}</td>
                            <td>{car.portes}</td>
                            <td>{car.sizezantas}</td>
                            <td>{car.color}</td>
                            <td>
                                <Link to={`/edit-car/${car._id}`} className="btn btn-
success" >Edit</Link>
                                <button onClick={() => deleteCar(car._id)} className="btn
btn-danger" >Delete</button>
                            </td>
                        </tr>
                    ))}
                </tbody>
            </table>
        </div>
    </div>
</div>
    )
}
export default UserList
```

```
<!-- Cars.js -->
import React, { useState, useEffect } from "react";
import axios from "axios";
import { Link } from "react-router-dom"
import "../styles/cars.css";
// import login user
import { useNavigate } from 'react-router-dom'
import { useSelector } from 'react-redux'
function Cars({ handleClick }) {
  const [cars, setCar] = useState([]);
  useEffect(() => {
    getCar();
  }, []);
  const getCar = async () => {
    const response = await axios.get('http://localhost:5000/api/cars/');
    setCar(response.data)
  }
  const navigate = useNavigate()
  const { user } = useSelector((state) => state.auth)
  useEffect(() => {
    if (!user) {
      navigate('/login')
    }
  }, [user, navigate])
  return (
    <div className='container'>
        <div className='heading'>
```

```
            <h1>Cars</h1>
            <p>CAR FOR YOU</p>
          </div>
          <div className="section">
           <div className='items-cars'>
            {cars.map((car, index) => (
             <div className='item' key={car._id}>
              <img src={car.img}
              // src={require(`${car.img}`)}
              alt="img" />

              <div className='info-content'>
               <h6 className='title-card'>{car.brand}</h6>
               <h4 className='title-card'>{car.model}</h4>
               <div className='Buttons'>
                <button className='btn btn-cars' onClick={() => handleClick(car)}>Add to Compare</button>
                <button className='btn btn-cars'>
                 <Link to={`/cars/${car._id}`} >View</Link>
                </button>
               </div>
              </div>
             </div>
            ))}
           </div>
          </div>
        </div>
```

```
 )
}
export default Cars
<!-- CompareCars.js -->
// import login user
import { useNavigate } from 'react-router-dom'
import { useSelector } from 'react-redux'
import { useEffect } from 'react'
import "../styles/compare.css";
const CompareCars = ({ cart, setCart
 // , HandleChange
}) => {
 const navigate = useNavigate()
 const { user } = useSelector((state) => state.auth)
 useEffect(() => {
  if (!user) {
   navigate('/login')
  }
 }, [user, navigate])
 const handleRemove = (id) => {
  const arr = cart.filter((car) => car.id !== id);
  setCart(arr);
 };
 return (
  <div className="container">
   <div className='heading'>
    <h1>
```

```
    Compare Cars
  </h1>
  <p>CAR FOR YOU</p>
</div>
<div className="container-list">
  <div className="responsive-compare-table">
    <table className='table-stripes'>
      <tbody >
        <tr>
          <th> </th>
          {cart.map((car, index) => (
            <td key={car._id}><img src={car.img} alt="img" /></td>
          ))}
        </tr>
        <tr>
          <th>No</th>
          {cart.map((car, index) => (
            <td key={car._id}>{index + 1}</td>
          ))}
        </tr>
        <tr>
          <th>Μάρκα</th>
          {cart.map((car, index) => (
            <td key={car._id}>{car.brand}</td>
          ))}
        </tr>
        <tr>
```

```
  <th>Μοντέλο</th>
  {cart.map((car, index) => (
    <td key={car._id}>{car.model}</td>
  ))}
</tr>
<tr>
  <th>Καύσιμα</th>
  {cart.map((car, index) => (
    <td key={car._id}>{car.kausima}</td>
  ))}
</tr>
<tr>
  <th>Κυβικά</th>
  {cart.map((car, index) => (
    <td key={car._id}>{car.kibika}</td>
  ))}
</tr>
<tr>
  <th>Ίπποι</th>
  {cart.map((car, index) => (
    <td key={car._id}>{car.ippoi}</td>
  ))}
</tr>
<tr>
  <th>Κατηγορία</th>
  {cart.map((car, index) => (
    <td key={car._id}>{car.katigoria}</td>
```

```
  ))}
</tr>
<tr>
  <th>Σασμάν</th>
  {cart.map((car, index) => (
    <td key={car._id}>{car.sasman}</td>
  ))}
</tr>
<tr>
  <th>Κίνηση</th>
  {cart.map((car, index) => (
    <td key={car._id}>{car.kinisi}</td>
  ))}
</tr>
<tr>
  <th>Πόρτες</th>
  {cart.map((car, index) => (
    <td key={car._id}>{car.portes}</td>
  ))}
</tr>
<tr>
  <th>Μέγεθος Ζάντας</th>
  {cart.map((car, index) => (
    <td key={car._id}>{car.sizezantas}</td>
  ))}
</tr>
<tr>
```

```
                <th>Χρώμα</th>
                {cart.map((car, index) => (
                  <td key={car._id}>{car.color}</td>
                ))}
              </tr>
              <tr>
                <th></th>
                {cart.map((car, index) => (
                  <td key={car._id}><button onClick={() =>
handleRemove(car.id)}>Remove</button></td>
                ))}
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </div>
  );
};
export default CompareCars;
<!-- Copyright.js -->
import React from 'react'
function Copyright() {
  return (
    <div className='copyright'>
      <p className='copyright-leter'> &copy; {new Date().getFullYear()} Xanti-
Kastoria | Nikos - Xakan</p>
    </div>
```

```
 )
}
export default Copyright
<!-- EditCar.js -->
import React, { useState, useEffect } from "react";
import axios from "axios";
import { useNavigate, useParams } from "react-router-dom";
// New
import { useSelector } from 'react-redux'
const EditCar = () => {
    const [model, setModel] = useState("")
    const [brand, setBrand] = useState("")
    const [kausima, setKausima] = useState("Υβριδικό plug-in βενζίνη")
    const [kibika, setKibika] = useState("")
    const [ippoi, setIppoi] = useState("")
    const [katigoria, setKatigoria] = useState("Λιμουζίνα/Sedan")
    const [sasman, setSazman] = useState("Αυτόματο")
    const [kinisi, setKinisi] = useState("Προσθιοκίνητο (FWD)")
    const [portes, setPortes] = useState(" 7")
    const [sizezantas, setSizezantas] = useState("22")
    const [color, setColor] = useState("")
    const navigate = useNavigate()
    const { id } = useParams()
    useEffect(() => {
        getCarById();
    }, []);
    const getCarById = async () => {
```

```
    const response = await axios.get(`http://localhost:5000/api/cars/${id}`);
    setModel(response.data.model);
    setBrand(response.data.brand);
    setKausima(response.data.kausima);
    setKibika(response.data.kibika);
    setIppoi(response.data.ippoi);
    setKatigoria(response.data.katigoria);
    setSazman(response.data.sasman);
    setKinisi(response.data.kinisi);
    setPortes(response.data.portes);
    setSizezantas(response.data.sizezantas);
    setColor(response.data.color);
};
const putCar = async (e) => {
    e.preventDefault();
    try {
        await axios.put(`http://localhost:5000/api/cars/${id}`, {
            model,
            brand,
            kausima,
            kibika,
            ippoi,
            katigoria,
            sasman,
            kinisi,
            portes,
            sizezantas,
```

```
            color,
        })
        navigate("/managment-cars")
    } catch (error) {
        console.log(error)
    }
}
const { user } = useSelector((state) => state.auth)
useEffect(() => {
  if (!user) {
    navigate('/login')
  }

}, [user, navigate])
return (
    <div className="container">
        <div className='heading'>
            <h1>Edit a Car</h1>
            <p>CAR FOR YOU</p>
        </div>

        <div className="section">
            <div className='columns'>
                <div className='form'>
                    <form onSubmit={putCar}>
                        <div className='field'>
                            <label className='label'>Μοντέλο</label>
```

```
                        <div className='control'>
                            <input type="text"  className='input' value={model} onChange={(e) => setModel(e.target.value)}  placeholder='Μοντέλο π.χ. (Ferrari Roma 22 )'  />
                        </div>
                    </div>
                    <div className='field'>
                        <label className='label'>Μάρκα</label>
                        <div className='control'>
                            <input type="text" className='input'  value={brand} onChange={(e) => setBrand(e.target.value)} placeholder='Μάρκα π.χ. (Ferrari)'  />
                        </div>
                    </div>

                    <div className='field'>
                        <label className='label'>Κάυσιμα</label>
                        <div className='control'>
                            <div className='select is-fullwidth'>
                                <select value={kausima} onChange={(e) => setKausima(e.target.value)} >
                                    <option value="Άλλο">Άλλο</option>
                                    <option value="Υβριδικό plug-in βενζίνη">Υβριδικό plug-in βενζίνη</option>
                                    <option value="Βενζίνη">Βενζίνη</option>
                                    <option value="Πετρέλαιο">Πετρέλαιο</option>
                                    <option value="Υβριδικό πετρέλαιο">Υβριδικό πετρέλαιο</option>
                                    <option value="Υβριδικό βενζίνη">Υβριδικό βενζίνη</option>
```

```
                                <option value="Ηλεκτρικό">Ηλεκτρικό</option>
                            </select>
                        </div>
                    </div>
                </div>
                <div className='field'>
                    <label className='label'>Κυβικά</label>
                    <div className='control'>
                        <input type="text" className='input'  value={kibika}
onChange={(e) => setKibika(e.target.value)} placeholder='Κυβικά π.χ.(3.560 cc)' />
                    </div>
                </div>
                <div className='field'>
                    <label className='label'>Ίπποι</label>
                    <div className='control'>
                        <input type="text" className='input' value={ippoi}
onChange={(e) => setIppoi(e.target.value)} placeholder='Ίπποι π.χ. (620 bhp)' />
                    </div>
                </div>
                <div className='field'>
                    <label className='label'>Κατηγορία</label>
                    <div className='control'>
                        <div className='select is-fullwidth'>
                            <select value={katigoria} onChange={(e) =>
setKatigoria(e.target.value)}  >
                                <option value="Άλλο">Άλλο</option>
                                <option
value="Λιμουζίνα/Sedan">Λιμουζίνα/Sedan</option>
```

```
                                    <option
value="Κόμπακτ/Hatchback">Κόμπακτ/Hatchback</option>
                                    <option value="Κουπέ-Σπόρ">Κουπέ-Σπόρ</option>
                                    <option value="4χ4/Τζιπ/Suv">4χ4/Τζιπ/Suv</option>
                                    <option
value="Κόμπι/Κάραβαν">Κόμπι/Κάραβαν</option>
                                    <option
value="Κάμπριο/Roadster">Κάμπριο/Roadster</option>
                                    <option
value="Αγροτικό/Pickup">Αγροτικό/Pickup</option>
                                </select>
                            </div>
                        </div>
                    </div>
                    <div className='field'>
                        <label className='label'>Σασμάν</label>
                        <div className='control'>
                            <div className='select is-fullwidth'>
                                <select value={sasman} onChange={(e) =>
setSazman(e.target.value)} >
                                    <option value="Άλλο">Άλλο</option>
                                    <option value="Αυτόματο">Αυτόματο</option>
                                    <option value="Χειροκίνητο">Χειροκίνητο</option>
                                </select>
                            </div>
                        </div>
                    </div>
                    <div className='field'>
                        <label className='label'>Κίνηση</label>
```

```
                        <div className='control'>
                            <div className='select is-fullwidth'>
                                <select value={kinisi} onChange={(e) =>
setKinisi(e.target.value)} >
                                    <option value="Άλλο">Άλλο</option>
                                    <option value="Πισωκίνητο (RWD)">Πισωκίνητο
(RWD)</option>
                                    <option value="Προσθιοκίνητο (FWD)">Προσθιοκίνητο
(FWD)</option>
                                    <option value="4x4 (4WD)">4x4 (4WD)</option>
                                </select>
                            </div>
                        </div>
                    </div>
                    <div className='field'>
                        <label className='label'>Πόρτες</label>
                        <div className='control'>
                            <div className='select is-fullwidth'>
                                <select value={portes} onChange={(e) =>
setPortes(e.target.value)} >
                                    <option value="Άλλο">Άλλο</option>
                                    <option value="7">7</option>
                                    <option value="6">6</option>
                                    <option value="5">5</option>
                                    <option value="4">4</option>
                                    <option value="3">3</option>
                                    <option value="2">2</option>
                                </select>
```

```
                    </div>
                </div>
            </div>
            <div className='field'>
                <label className='label'>Μέγεθος Ζάντας</label>
                <div className='control'>
                    <div className='select is-fullwidth'>
                        <select value={sizezantas} onChange={(e) =>
setSizezantas(e.target.value)} >
                            <option value="Άλλο">Άλλο</option>
                            <option value="21F/22R">21F/22R</option>
                            <option value="22">22</option>
                            <option value="21">21</option>
                            <option value="20">20</option>
                            <option value="19">19</option>
                            <option value="18">18</option>
                            <option value="17">17</option>
                            <option value="16">16</option>
                            <option value="15">15</option>
                            <option value="14">14</option>
                        </select>
                    </div>
                </div>
            </div>
            <div className='field'>
                <label className='label'>Χρώμα</label>
                <div className='control'>
```

```
                                <input type="text" className='input' value={color}
onChange={(e) => setColor(e.target.value)} placeholder='Χρώμα π.χ. ( Κόκκινο
(Μεταλλικό) )' />

                            </div>

                        </div>

                        <div className='field'>

                            <div className='control'>

                                <button type="submit" className='btn btn-
cars'>Update</button>

                            </div>

                        </div>

                    </form>

                </div>

            </div>

        </div>

    </div>

    )

}

export default EditCar

<!-- Footer.js -->

import React from 'react'

import { Link } from 'react-router-dom'

import logo from "../assets/1.png"

function Footer() {

  return (

    <footer className='footer'>

      <div className='col col-1'>

        <div className='content-footer'>
```

```
            <p>
            All time classic application for compare cars! Are you ready for the beautiful experience!
            </p>
          </div>
          <img className='logo-img-footer' src={logo} alt="img-1" />
        </div>
        <div className='col col-2'>
          <h4>#AllTIMECLASSIC</h4>
          <ul>
            <li><Link to="/">Home</Link></li>
            <li><Link to="/about-us">About Us</Link></li>
            <li> <Link to="/cars">Cars</Link></li>
          </ul>
        </div>
        <div className='col col-3'>
          <h4>#CRUD</h4>
          <ul>
            <li><Link to="/managment-cars" className="button is-success">Managment Cars</Link></li>
            <li><Link to="/add-car" className="button is-success">Add Cars</Link></li>
          </ul>
        </div>
      </footer>
    )
  }
  export default Footer
```

```
<!-- Header.js -->
import { FaSignInAlt, FaSignOutAlt, FaUser } from 'react-icons/fa'
import { Link, useNavigate } from 'react-router-dom'
import { useSelector, useDispatch } from 'react-redux'
import { logout, reset } from '../features/auth/authSlice'
import React from 'react'
import { useRef } from "react";
import { FaBars, FaTimes, FaFilter } from "react-icons/fa";
import logo from "../assets/2.png"
function Header({ size }) {
  const navigate = useNavigate()
  const dispatch = useDispatch()
  const { user } = useSelector((state) => state.auth)
  const onLogout = () => {
    dispatch(logout())
    dispatch(reset())
    navigate('/')
  }
  const navRef = useRef();
  const showNavbar = () => {
    navRef.current.classList.toggle("responsive_nav");
  }
  return (
    <header>
      <Link to="/"><img src={logo} alt="logo" /></Link>
      <nav ref={navRef}>
        <Link to="/">Home</Link>
```

```
  <Link to="/about-us">About Us</Link>
  <Link to="/cars">Cars</Link>
  <Link to="/managment-cars">Managment</Link>
  <Link to="/add-car">Add Cars</Link>
  <button className='nav-btn nav-close-btn' onClick={showNavbar}>
    <FaTimes />
  </button>
</nav>
<ul className='compare'>
  <div className='buttons-header'>
    <button className='nav-btn' onClick={showNavbar}>
      <FaBars />
    </button>
    <div className='compare-btn'>
      <Link to="/compare-cars">
        <FaFilter />
      </Link>
    </div>
    <span>{size}</span>
  </div>
    {user ? (
      <li>
        <button className='btn btn-danger' onClick={onLogout}>
          <FaSignOutAlt /> Logout
        </button>
      </li>
    ) : (
```

```
            <>
              <li>
                <Link className='btn btn-success' to='/login'>
                  <FaSignInAlt /> Login
                </Link>
              </li>
              <li>
                <Link className='btn btn-main' to='/register'>
                  <FaUser /> Register
                </Link>
              </li>
            </>
          )}
      </ul>
    </header>
  )
}
export default Header
<!-- HomePage.js -->
import React from 'react'
import { Link } from 'react-router-dom';
import "../styles/homepage.css";
// image import
import image from "../assets/car-4.png"
// video import
import carForYou from "../assets/video-main.mp4"
// New
```

```
import { useNavigate } from 'react-router-dom'
import { useSelector} from 'react-redux'
import { useEffect } from 'react'
function HomePage() {
 const navigate = useNavigate()
 const { user } = useSelector((state) => state.auth)
 useEffect(() => {
  if (!user) {
   navigate('/login')
  }
 }, [user, navigate])
 return (
  <main>
   <>
          <video>
           <source src={carForYou} type="video/mp4" />
          </video>
         <div className='container'>
          <div className='section'>
           <div className='info-home'>
            <div className='image-wrapper'>
             <img src={image} alt='vintage' />
            </div>
            <div>
             <h2>#Carsforyou</h2>
             <p className='info-text'>
```

Τα τελευταία χρόνια η πληροφορική έχει εισχωρήσει σε πολλούς κλάδους και ένας από αυτούς είναι οι διαδικτυακές εφαρμογές.

Η ανάλυση των δυνατοτήτων δύο ή περισσότερων αυτοκινήτων είναι μία από αυτές τις εφαρμογές. Γενικότερα ο χρήστης με ένα μόνο κουμπί

μπορεί να συγκρίνει αυτοκίνητα, είτε για να λάβει πληροφορίες είτε για να αποκτήσει μία γνώση που μπορεί να τον βοηθήσει για το

μελλοντικό του αυτοκίνητο. Επίσης έχουμε δημιουργήσει ένα σύστημα διαχείρισής των χρηστών έτσι ώστε ο κάθε χρήστης να συνδέεται με

τα προσωπικά του στοιχεία και όλο αυτό με την βοήθεια μίας βάσης δεδομένων. Έχει παρατηρηθεί πως ο κόσμος ολοένα και εμπιστεύεται τέτοιου

είδους εφαρμογές γιατί κάνουν τους χρήστες να έχουν μία άμεση επαφή με την πληροφορία.

Οι εφαρμογές εξελίσσονται με ιλιγγιώδης ρυθμούς και οι ο ανταγωνισμός που υπάρχει βοηθάει στο να έχουμε καθημερινά πολύ

δημιουργικά και χρήσιμα εργαλεία.

</p>

</>

</main>

```
 )
}
export default HomePage
<!-- Login.js -->
import { useState, useEffect } from 'react'
import { FaSignInAlt } from 'react-icons/fa'
import { useSelector, useDispatch } from 'react-redux'
import { useNavigate } from 'react-router-dom'
import { toast } from 'react-toastify'
import { login, reset } from '../features/auth/authSlice'
function Login() {
  const [formData, setFormData] = useState({
    email: '',
    password: '',
  })
  const { email, password } = formData
  const navigate = useNavigate()
  const dispatch = useDispatch()
  const { user,
    isError, isSuccess, message } = useSelector(
    (state) => state.auth
  )
  useEffect(() => {
    if (isError) {
      toast.error(message)
    }
```

```
  if (isSuccess || user) {
    navigate('/')
  }
  dispatch(reset())
}, [user, isError, isSuccess, message, navigate, dispatch])
const onChange = (e) => {
  setFormData((prevState) => ({
    ...prevState,
    [e.target.name]: e.target.value,
  }))
}
const onSubmit = (e) => {
  e.preventDefault()
  const userData = {
    email,
    password,
  }
  dispatch(login(userData))
}
return (
  <>
    <div className='heading'>
      <h1>
        <FaSignInAlt /> Login
      </h1>
      <p>CAR FOR YOU</p>
    </div>
```

```
<div className="columns">
<div className='form'>
  <form onSubmit={onSubmit}>
    <br/>
    <div className="field">
    <div className='control'>
      <input
        type='email'
        className='input'
        id='email'
        name='email'
        value={email}
        placeholder='Enter your email'
        onChange={onChange}
      />
    </div>
      </div>
      <div className="field">
      <div className='control'>
      <input
        type='password'
        className='input'
        id='password'
        name='password'
        value={password}
        placeholder='Enter password'
        onChange={onChange}
```

```
            />
          </div>
            </div>
            <div className="field">
            <div className='control'>
            <button type='submit' className='btn btn-cars'>
              Login
            </button>
          </div>
            </div>
        </form>
      </div>
      </div>
    </>
  )
}
export default Login
<!-- Register.js -->
import { useState, useEffect } from 'react'
import { useSelector, useDispatch } from 'react-redux'
import { useNavigate } from 'react-router-dom'
import { toast } from 'react-toastify'
import { FaUser } from 'react-icons/fa'
import { register, reset } from '../features/auth/authSlice'
function Register() {
  const [formData, setFormData] = useState({
    name: '',
```

```
  email: '',
  password: '',
  password2: '',
})
const { name, email, password, password2 } = formData
const navigate = useNavigate()
const dispatch = useDispatch()
const { user,
  isError, isSuccess, message } = useSelector(
  (state) => state.auth
)
useEffect(() => {
  if (isError) {
    toast.error(message)
  }
  if (isSuccess || user) {
    navigate('/')
  }
  dispatch(reset())
}, [user, isError, isSuccess, message, navigate, dispatch])
const onChange = (e) => {
  setFormData((prevState) => ({
    ...prevState,
    [e.target.name]: e.target.value,
  }))
}
const onSubmit = (e) => {
```

```
  e.preventDefault()
  if (password !== password2) {
    toast.error('Passwords do not match')
  } else {
    const userData = {
      name,
      email,
      password,
    }
    dispatch(register(userData))
  }
}
return (
  <>
    <div className='heading'>
      <h1>
        <FaUser /> Register
      </h1>
      <p>CAR FOR YOU</p>
    </div>
    <div className="columns">
    <div className='form'>
      <form onSubmit={onSubmit}>
      <br/>
        <div className="field">
        <div className='control'>
          <input
```

```
    type='text'
    className='input'
    id='name'
    name='name'
    value={name}
    placeholder='Enter your name'
    onChange={onChange}
  />
</div>
</div>
  <div className="field" >
  <div className='control'>
  <input
    type='email'
    className='input'
    id='email'
    name='email'
    value={email}
    placeholder='Enter your email'
    onChange={onChange}
  />
</div>
  </div>
  <div className="field" >
  <div className='control'>
  <input
    type='password'
```

```
    className='input'
    id='password'
    name='password'
    value={password}
    placeholder='Enter password'
    onChange={onChange}
  />
</div>
  </div>
  <div className="field" >
  <div className='control'>
  <input
    type='password'
    className='input'
    id='password2'
    name='password2'
    value={password2}
    placeholder='Confirm password'
    onChange={onChange}
  />
</div>
  </div>
  <div className="field" >
  <div className='control'>
  <button type='submit' className='btn btn-cars'>
    Register
  </button>
```

```
          </div>
            </div>
        </form>
      </div>
        </div>
    </>
  )
}
export default Register
<!-- ScrollToTop.js -->
import React, { useState, useEffect } from "react";
import { FaAngleUp } from "react-icons/fa";
const ScrollToTop = () => {
    const [showTopBtn, setShowTopBtn] = useState(false);
    useEffect(() => {
        window.addEventListener("scroll", () => {
            if (window.scrollY > 400) {
                setShowTopBtn(true);
            } else {
                setShowTopBtn(false);
            }
        });
    }, []);
    const goToTop = () => {
        window.scrollTo({
            top: 0,
            behavior: "smooth",
```

```
        });
    };
    return (
        <div className="top-to-btm">
            {" "}
            {showTopBtn && (
                <FaAngleUp
                    className="icon-position icon-style"
                    onClick={goToTop}
                />
            )}{" "}
        </div>
    );
};
export default ScrollToTop;
<!-- features/auth -->
<!-- authService.js -->
import axios from 'axios'
const API_URL = 'http://localhost:5000/api/users/'
// Register user
const register = async (userData) => {
  const response = await axios.post(API_URL, userData)
  if (response.data) {
    localStorage.setItem('user', JSON.stringify(response.data))
  }
  return response.data
}
```

```
// Login user
const login = async (userData) => {
  const response = await axios.post(API_URL + 'login', userData)
  if (response.data) {
    localStorage.setItem('user', JSON.stringify(response.data))
  }
  return response.data
}
// Logout user
const logout = () => {
  localStorage.removeItem('user')
}
const authService = {
  register,
  logout,
  login,
}
export default authService
<!-- authSlice -->
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit'
import authService from './authService'
// Get user from localStorage
const user = JSON.parse(localStorage.getItem('user'))
const initialState = {
  user: user ? user : null,
  isError: false,
  isSuccess: false,
```

```
  isLoading: false,
  message: '',
}
// Register user
export const register = createAsyncThunk(
  'auth/register',
  async (user, thunkAPI) => {
    try {
      return await authService.register(user)
    } catch (error) {
      const message =
        (error.response &&
          error.response.data &&
          error.response.data.message) ||
        error.message ||
        error.toString()
      return thunkAPI.rejectWithValue(message)
    }
  }
)
// Login user
export const login = createAsyncThunk('auth/login', async (user, thunkAPI) => {
  try {
    return await authService.login(user)
  } catch (error) {
    const message =
      (error.response && error.response.data && error.response.data.message) ||
```

```
      error.message ||
      error.toString()
    return thunkAPI.rejectWithValue(message)
  }
})
export const logout = createAsyncThunk('auth/logout', async () => {
  await authService.logout()
})
export const authSlice = createSlice({
  name: 'auth',
  initialState,
  reducers: {
    reset: (state) => {
      state.isLoading = false
      state.isSuccess = false
      state.isError = false
      state.message = ''
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(register.pending, (state) => {
        state.isLoading = true
      })
      .addCase(register.fulfilled, (state, action) => {
        state.isLoading = false
        state.isSuccess = true
```

```
        state.user = action.payload
      })
      .addCase(register.rejected, (state, action) => {
        state.isLoading = false
        state.isError = true
        state.message = action.payload
        state.user = null
      })
      .addCase(login.pending, (state) => {
        state.isLoading = true
      })
      .addCase(login.fulfilled, (state, action) => {
        state.isLoading = false
        state.isSuccess = true
        state.user = action.payload
      })
      .addCase(login.rejected, (state, action) => {
        state.isLoading = false
        state.isError = true
        state.message = action.payload
        state.user = null
      })
      .addCase(logout.fulfilled, (state) => {
        state.user = null
      })
  },
})
```

```
export const { reset } = authSlice.actions

export default authSlice.reducer

<!-- styles -->

<!-- about-us.css -->

.about-content p { text-align: justify; }

@media only screen and (min-width: 1024px) {

    .about-content { display: grid; gap: 3em; align-items: center; grid-template-columns: 35% auto; }

}

<!-- back-to-top.css -->

.top-to-btm {  position: relative;}

.icon-position {  position: fixed;  bottom: 40px;  right: 25px;  z-index: 20;}

.icon-style {  background-color: var(--mainColor);  border: 2px solid #fff;  border-radius: 50%;  height: 50px;  width: 50px;  color: #fff;  cursor: pointer;  animation: movebtn 3s ease-in-out infinite;  transition: all .5s ease-in-out;}

.icon-style:hover {  -webkit-animation: none;  animation: none;  background: #fff;  color: var(--mainColor);  border: 2px solid var(--mainColor);}

@-webkit-keyframes movebtn {

    0% { transform: translateY(0px); }

    25% { transform: translateY(20px); }

    50% { transform: translateY(0px); }

    75% { transform: translateY(-20px); }

    100% { transform: translateY(0px); }

}

@keyframes movebtn {

    0% { transform: translateY(0px); }

    25% { transform: translateY(20px); }

    50% { transform: translateY(0px); }
```

```
    75% { transform: translateY(-20px); }

    100% { transform: translateY(0px); }

}

<!-- basic.css -->

@import
url('https://fonts.googleapis.com/css2?family=Roboto+Slab:wght@100;200;300;400;500;
600;700;800;900&display=swap');

* { padding: 0; box-sizing: border-box; }

:root { --mainColor: #5B96A9; --mainColorLight: #545454; --secontaryColor:
#d1c8b4; --textColor: #eee; --blackColor: #000; }

body { font-size: 16px; line-height: 1.5; font-family: 'Roboto Slab', serif; color:
#565656; background-color: #fff; margin: 0; position: relative; }

div { position: relative;}

section { position: relative; }

.section{ margin: 2em 0; }

.container { max-width: 1200px; padding: 0 25px; margin: auto; }

img,

video { max-width: 100%; display: block; margin: 0 auto; }

a { color: inherit; cursor: pointer; outline: none; text-decoration: none; }

ul { list-style: none; }

.heading {  text-align: center; line-height: 1;  margin: 2em;}

.heading span{ color: var(--mainColor);}

.heading h1:hover {  color: var(--secontaryColor);}

@media only screen and (min-width: 1024px) {

  .container { padding: 0 15px;}

  .heading {font-size: 2em;}

}

<!-- car.css -->
```

```
.item-car { display: grid; grid-template-columns: 1fr 1fr; align-items: center; justify-content: center; }

.item-car .content-car ul li{ color: var(--mainColor);  font-size: 25px;}

.item-car .content-car ul li span {font-size: 22px;  color: #000;}

.title-card { text-overflow: ellipsis; overflow: hidden; white-space: nowrap; }

<!-- cars.css -->

.items-cars { display: grid; grid-template-columns: repeat(auto-fit, 250px); justify-content: center; gap: 2em; }

<!-- compare.css -->

.container-list { margin: 1rem; }

.responsive-compare-table {overflow-x: auto; width: 90%; }

@media only screen and (min-width: 1024px) {

   .responsive-compare-table { width: 100%; }

   table.table-stripes { width: 100%; }

}

<!-- footer.css -->

.logo-img-footer { width: auto; height: 180px; }

footer.footer { background: var(--secontaryColor); display: flex; justify-content: space-around; text-align: start; padding: 3rem; color: var(--blackColor); flex-direction: row-reverse; }

footer ul li:hover a{ color: var(--textColor); }

.copyright { background: var(--mainColorLight); color: var(--secontaryColor); text-align: end; font-size: 14px; padding: 5px; }

@media only screen and (max-width: 1024px) {

   footer.footer { top: 0; left: 0; height: 100%; width: 100%; display: flex; flex-direction: column; align-items: center; justify-content: center; gap: 1.5rem; z-index: 2; }

   .col-1 { justify-content: center; text-align: center; }

 }

<!-- global.css -->
```

```
.btn-success {  color: #fff;  background-color: #198754;  border-color: #198754;}

.btn-success:hover {  color: #fff;  background-color: #157347;  border-color: #146c43;}

.btn-main {  color: #fff;  background-color: #2b33a8;  border-color: #2b33a8;}

.btn-main:hover {  color: #fff;  background-color: #1c2494;  border-color: #1c2494;}

.btn-danger {  color: #fff;  background-color: #dc3545;  border-color: #dc3545;}

.btn-danger:hover {  color: #fff;  background-color: #bb2d3b;  border-color: #b02a37;}

.btn {  display: inline-block;  font-weight: 400;  margin: 2px;  text-align: center;  text-decoration: none;  vertical-align: middle;  cursor: pointer;  user-select: none;  border: 1px solid transparent;  padding: .3rem 0.7rem;  font-size: 13px;  transition: color .15s ease-in-out, background-color .15s ease-in-out, border-color .15s ease-in-out, box-shadow .15s ease-in-out, -webkit-box-shadow .15s ease-in-out;}

.Buttons {  display: grid;  gap: 10px;}

button.btn.btn-cars {  position: relative;  width: 100%;  height: 45px;  padding: 0px 1.5rem;  margin: 0px;  color: white;  text-decoration: none;  border-radius: 0px;  box-sizing: border-box;  border: 1px solid transparent;  font-size: 1.1rem;  line-height: 1rem;  font-family: var(--font-urw-din);  font-weight: 500;  text-transform: uppercase;  word-break: keep-all;  cursor: pointer;  background-color: var(--mainColor);  display: inline-flex;  justify-content: center;  align-items: center;}

button.btn.btn-cars:hover {  border: 1px solid transparent;  background-image: linear-gradient(rgb(55, 55, 55), var(--mainColor));  transition: background-color 400ms ease-in 0s;  text-decoration: none;  color: white;}

.button-all-cars {  width: 20%;  margin: 0 auto;}

/* form */

.form {  padding: 20px;  width: 50%;  background: var(--secontaryColor);  border-radius: 150px 0 0 0;  box-shadow: 0px 2px 8px 0px;}

.field {  text-align: center;  margin: 15px;}

.columns { display: flex; justify-content: center; align-items: center; margin: 2em auto }

table { border-collapse: collapse; border-spacing: 0; width: 100%; border: 1px solid black; }

.responsive-table { width: 90%; overflow: auto; }
```

```
th,

td { border: 1px solid black; text-align: left; padding: 5px; font-size: 13px; }

tr:nth-child(even) { background-color: var(--mainColor); color: var(--textColor); }

input.input {  width: 100%;  padding: .375rem .75rem;  font-size: 1rem; background-color: #fff; border: 1px solid #ced4da; }

select {width: 100%; padding: .375rem .75rem;  font-size: 1rem; background-color: #fff; border: 1px solid #ced4da; }

@media only screen and (max-width: 1024px) {

  .form { width: 80%; }

  .columns { margin: 10px; }

}

<!-- header.css -->

header { display: flex; align-items: center; justify-content: space-between; height: 110px; padding: 0 2rem; background-color: var(--mainColor); color: var(--textColor); }

header img { height: 180px;}

header .nav-btn { padding: 5px; cursor: pointer; background: transparent; border: none; outline: none; color: var(--textColor); visibility: hidden; opacity: 0; font-size: 1.8rem; }

header nav a { margin: 0 2rem; color: var(--textColor); text-decoration: none; }

header nav a:hover { color: var(--secontaryColor); }

header nav Link { margin: 0 2rem; color: var(--textColor); text-decoration: none; }

header nav Link:hover { color: var(--secontaryColor);}

header .buttons-header {  display: inherit;}

.compare {  display: flex;  flex-direction: row-reverse;  align-items: center;}

.compare-btn {  padding: 5px;  cursor: pointer;  background: transparent;  border: none;  outline: none;  color: var(--textColor);  font-size: 1.8rem;}

@media only screen and (max-width: 1024px) {

    header .nav-btn { visibility: visible; opacity: 1; }
```

```
    header nav { position: fixed; top: 0; left: 0; height: 100%; width: 100%; display: flex; flex-direction: column; align-items: center; justify-content: center; gap: 1.5rem; background-color: var(--mainColor); transition: 1s; transform: translateY(-100vh); z-index: 2; }

    header .responsive_nav { transform: none !important; }

    nav .nav-close-btn { position: absolute; top: 2rem; right: 2rem; }

    nav a { font-size: 1.5rem; }

}

@media only screen and (max-width: 1024px) {

    header .nav-btn { visibility: visible; opacity: 1; }

}

<!-- homepage.css -->

.info-text {  text-align: justify;}

.model-title {  font-weight: bolder;  color: var(--mainColor);}

.image-wrapper { height: 0; padding-bottom: 100%; border-radius: 50%; overflow: hidden; z-index: 1; }

.image-wrapper img { width: 100%; height: 100%; -o-object-fit: cover; object-fit: cover; position: absolute; left: 0; top: 0; }

@media only screen and (min-width: 1024px) {

    .info-home { display: grid; gap: 3em; align-items: center; grid-template-columns: 35% auto; }

    .cars-home {     display: grid; column-gap: 2em; grid-template-columns: repeat(3, 1fr); }

}

<!-- App.js -->

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

import React, { useState } from "react";

import { ToastContainer } from 'react-toastify'

import 'react-toastify/dist/ReactToastify.css'
```

```
// import Components
import Header from './components/Header'
import Login from './components/Login'
import Register from './components/Register'
import CarList from "./components/CarList";
import AddCar from "./components/AddCar";
import EditCar from "./components/EditCar";
import HomePage from "./components/HomePage";
import AboutUs from "./components/AboutUs";
import Cars from "./components/Cars";
import CompareCars from "./components/CompareCars";
import Car from "./components/Car";
import Footer from "./components/Footer";
import Copyright from "./components/Copyright";
import ScrollToTop from "./components/ScrollToTop";
function App() {
  const [cart, setCart] = useState([]);
  const handleClick = (car) => {
    if (cart.indexOf(car) !== -1) return;
    setCart([...cart, car]);
  };
  const handleChange = (item, d) => {
    const ind = cart.indexOf(item);
    const arr = cart;
    arr[ind].amount += d;
    if (arr[ind].amount === 0) arr[ind].amount = -1;
    setCart([...arr]);
```

```
  };
  return (
    <BrowserRouter>
      <ScrollToTop />
      <Header  size={cart.length} />
      <Routes>
      <Route path='/login' element={<Login />} />
      <Route path='/register' element={<Register />} />
      <Route path='/' element={<HomePage />} />
      <Route path="about-us" element={<AboutUs />} />
      <Route path="cars" element={<Cars handleClick={handleClick} />} />
      <Route path="cars/:id" element={<Car />} />
      <Route path="compare-cars" element={<CompareCars cart={cart}
setCart={setCart} handleChange={handleChange} /* handleChange={handleChange} */
/>} />
      <Route path="add-car" element={<AddCar />} />
      <Route path="edit-car/:id" element={<EditCar />} />
      <Route path="managment-cars" element={<CarList />} />
      </Routes>
      <Footer />
      <Copyright />
      <ToastContainer />
    </BrowserRouter>
  )
}
export default App
<!-- index.js -->
import React from 'react';
```

```
import { createRoot } from 'react-dom/client';
import { Provider } from 'react-redux';
import { store } from './app/store';
import App from './App';
import "./styles/global.css";
import "./styles/basic.css";
import "./styles/header.css";
import "./styles/footer.css";
import "./styles/back-to-top.css";
const container = document.getElementById('root');
const root = createRoot(container);
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
);
```