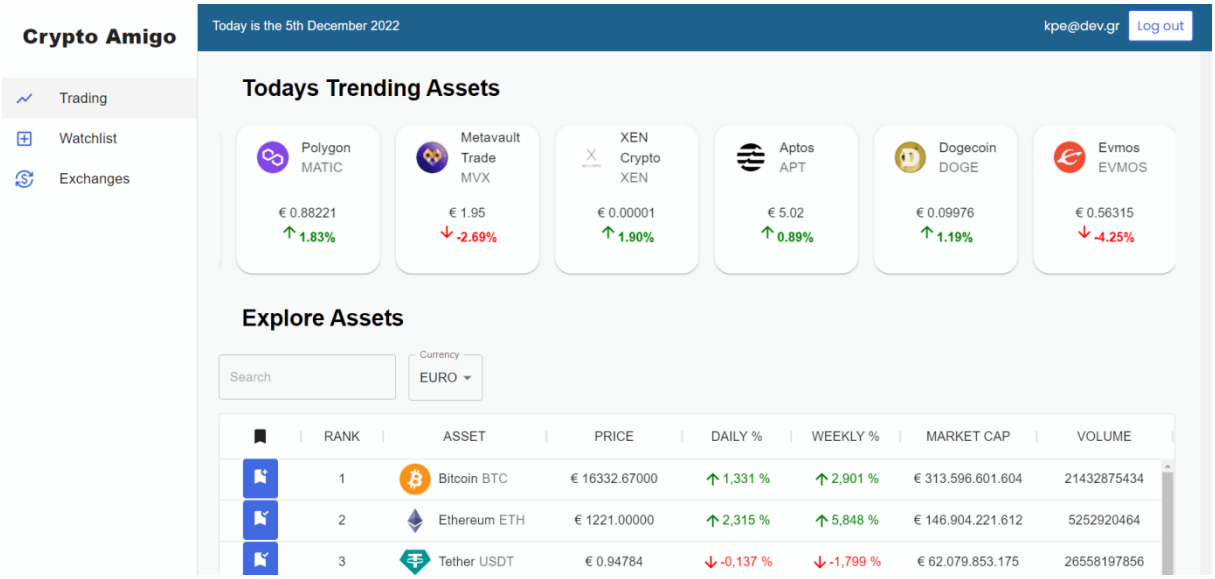
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη Διαδικτυακής Εφαρμογής Παρακολούθησης Κρυπτονομισμάτων»



Του φοιτητή
Πεχλιβανίδη Κωνσταντίνου
Αρ. Μητρώου: 174908

Επιβλέπων
Σιδηρόπουλος Αντώνιος
Βαθμίδα: Επίκουρος Καθηγητής

Ιανουάριος 2023

Τίτλος Δ.Ε.: Ανάπτυξη Διαδικτυακής Εφαρμογής Παρακολούθησης Κρυπτονομισμάτων
Κωδικός Δ.Ε.:22228
Ονοματεπώνυμο φοιτητή: Πεχλιβανίδης Κωνσταντίνος
Ονοματεπώνυμο εισηγητή: Σιδηρόπουλος Αντώνιος
Ημερομηνία ανάληψης Δ.Ε.:12-05-2022
Ημερομηνία περάτωσης Δ.Ε.:

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

««Την παρούσα διπλωματική εργασία την αφιερώνω στους ανθρώπους που με στήριξαν σε όλα αυτά τα χρόνια των σπουδών μου»»

Πρόλογος

Ο ενθουσιασμός ο οποίος έχει παρατηρηθεί τα τελευταία χρόνια λόγω της υψηλής αυξομείωσης των τιμών των κρυπτονομίσματα είναι πολύ μεγάλος. Αρωγός σε αυτή την αυξανόμενη τάση του αγοραστικού κοινού αλλά και των επιχειρήσεων που δραστηριοποιούνται στο ηλεκτρονικό εμπόριο αποτελούν οι εφαρμογές παρακολούθησης των τιμών των κρυπτονομισμάτων. Αυτός ήταν ο αρχικός λόγος για τον οποίο επέλεξα αυτή την Διπλωματική Εργασία καθώς ήμουν περίεργος να ανακαλύψω τον τρόπο λειτουργίας των κρυπτονομισμάτων αλλά και το πως θα μπορούσα να δημιουργήσω μια ολοκληρωμένη διαδικτυακή εφαρμογή για την παρακολούθηση της αυξομείωσης της τιμής τους. Η διπλωματική αυτή με βοήθησε να εξασκηθώ και εμβαθύνω τις γνώσεις μου στην ανάπτυξη μιας διαδικτυακής εφαρμογής που αλληλεπιδρά με διαφορετικά application programming interfaces (APIs). Καταληκτικά, είναι βέβαιο ότι μου πρόσφερε πολύ σημαντικά εφόδια και γνώσεις για το μέλλον μου στην αναζήτηση εργασίας.

Περίληψη

Η παρούσα διπλωματική εργασία αναπτύχθηκε στο τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου Ελλάδος με έδρα την Θεσσαλονίκη. Το αντικείμενο της παρούσας Διπλωματικής Εργασίας είναι η δημιουργία μίας ολοκληρωμένης διαδικτυακής εφαρμογής η οποία προσφέρει στους χρήστες της την δυνατότητα να παρακολουθήσουν την εξέλιξη της τιμής των κρυπτονομισμάτων που επιθυμούν. Οι χρήστες κάνοντας "login" χρησιμοποιώντας το προσωπικό τους "email" και "password" έχουν πρόσβαση σε ένα ευρύ φάσμα πληροφοριών όπως οι τρέχων τιμές των κρυπτονομισμάτων. Τα δεδομένα αυτά αναπαρίστανται σε ένα πίνακα όπου οι χρήστες μπορούν αναζητούν κάποιο συγκεκριμένο κρυπτονόμισμα αλλά και να ταξινομούν τα δεδομένα του, βάσει του στοιχείου του πίνακα που τους ενδιαφέρει. Επιπλέον, ο πίνακας διαθέτει την επιλογή αποθήκευσης ώστε ο χρήστης να μπορεί να βλέπει τα κρυπτονομίσματα της αρέσκειας του ξεχωριστά σε μια άλλη σελίδα η οποία ονομάζεται λίστα παρακολούθησης "watchlist". Τέλος, με την επιλογή κάποιου συγκεκριμένου κρυπτονομίσματος μέσα από τον πίνακα παρουσιάζονται οι αυξομειώσεις της τιμής του με την χρήση διαγραμμάτων αλλά και με πρόσθετα στατιστικά στοιχεία και λεπτομέρειες για την δράση του. Μια ακόμη σελίδα περιέχει τα υπάρχοντα ανταλλακτήρια με χρήσιμες πληροφορίες για την αγορά των κρυπτονομισμάτων.

Η εφαρμογή αναπτύχθηκε με την βοήθεια της στοίβας MERN δηλαδή τη συλλογή τεχνολογιών:

- MongoDB - βάση δεδομένων εγγράφων.
- Express(.js) - Πλαίσιο ιστού Node.js από την πλευρά του εξυπηρετητή.
- React(.js) - ένα πλαίσιο JavaScript από την πλευρά του πελάτη.

Τα application programming interfaces (APIs) από το οποία λαμβάνονται τα δεδομένα είναι το CoinGecko API για τα κρυπτονομίσματα και το Fixer.io για τις τρέχων συναλλαγματικές ισοτιμίες.

«Development of Online Cryptocurrency Monitoring Application»

«Pechlivanidis Konstantinos»

Abstract

The subject of this Diploma Thesis is the development of an integrated web application which offers to users the ability to monitor the price of cryptocurrencies that they desire. Users login by using their personal “email” and “password” have access to a wide range of information such as current cryptocurrency prices. These data are represented in a table where users can search for a specific cryptocurrency but also to sort its data, based on the element of the table that Interest. In addition, the table has the save option so that the user is able to see the cryptocurrencies of his preference in another page called “watchlist”. Finally, by selecting a specific cryptocurrency is presented the price using charts and also with additional statistics and details for its action. One more page contains existing exchange brokers with useful information for the cryptocurrency market.

The application was developed with the help of the MERN stack i.e. the collection of technologies:

- MongoDB - document database.
- Express(.js) - Server-side Node.js web framework.
- React(.js) - a client-side JavaScript framework.

The application programming interfaces (APIs) from which the data are obtained, is the CoinGecko API for cryptocurrencies and Fixer.io for current exchange rates.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον υπεύθυνο καθηγητή μου, κ. Σιδηρόπουλο, για την εμπιστοσύνη που μου έδειξε για την ανάθεση της παραπάνω Διπλωματικής εργασίας. Οι συμβουλές, οι οδηγίες, η καθοδήγηση, το ενδιαφέρον του και η συμπαράστασή του κατά τη συγγραφή της εργασίας, ήταν καθοριστικός παράγοντας. Ευχαριστώ πολύ επίσης την οικογένεια μου που με υποστήριξε από την αρχή μέχρι το τέλος των σπουδών μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
API	Application Programming Interface
AWS	Amazon Web Services
BSON	Binary JavaScript Object Notation
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
P2P	Peer-To-Peer
JSON	JavaScript Object Notation
JSX	JavaScript Syntax Extension
JWT	Json Web Token
NPM	Node Package Manager
ODM	Object Document Mapping
REST	Representational State Transfer
VDOM	Virtual Document Object Model
URL	Uniform Resource Locator
XML	Extensible Markup Language

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή στην ανάπτυξη της εργασίας

Η εξέλιξη της τεχνολογίας και του διαδικτυακού δικτύου συναλλαγών, έφερε στην επιφάνεια μια από τις πιο ενδιαφέρουσες καινοτομίες χρηματοοικονομικής φύσεως, τη δημιουργία των κρυπτονομισμάτων. Τα κρυπτονομίσματα ορίζονται ως η τελευταία εξέλιξη των ψηφιακών νομισμάτων. Επινοήθηκαν ως σύστημα ομότιμης σύνδεσης για ηλεκτρονικές πληρωμές, που δεν απαιτούν αξιόπιστη κεντρική αρχή και δεν έχουν υλική υπόσταση. Στηρίζονται σε μια καινοτόμο τεχνολογία που ονομάζεται «Blockchain» και παράγονται με διαφορετικό τρόπο από εκείνον που παράγονται τα παραδοσιακά νομίσματα.

Τα τελευταία χρόνια, το ενδιαφέρον γύρω από τα κρυπτονομίσματα συνεχώς αυξάνεται καθώς όλο και περισσότεροι επενδυτές και επιχειρήσεις τα χρησιμοποιούν, με αποτέλεσμα ο όγκος συναλλαγών βάσει αυτών να φτάνει σε όλο και υψηλότερα επίπεδα χρόνο με το χρόνο. Επιπλέον, η χρηματοοικονομική κρίση συνέβαλε στο να κοινοποιηθούν και στους απλούς πολίτες ως πιθανή διέξοδος από το παραστατικό χρήμα, δηλαδή το μέσον πληρωμής το οποίο επιβάλλεται στις συναλλαγές από κάποια συγκεκριμένη αρχή, πχ το κράτος. Ωστόσο, σήμερα, το ποσοστό των πολιτών που επιχειρούν να τα χρησιμοποιήσουν παραμένει περιορισμένο. Αυτό συμβαίνει γιατί ο κόσμος αδυνατεί να καταλάβει επακριβώς την πολύπλοκη τεχνολογία του και επομένως τα αξιοσημείωτα οφέλη που μπορεί να επιτύχει. Γι' αυτό τον λόγο ενώ παρατηρείται σταδιακή αύξηση ανά έτος στην αξία τους, υπάρχουν περίοδοι που η αξία τους μειώνεται σημαντικά. Τις μεγάλες αυτές αυξομειώσεις που δημιουργούνται στο εμπόριο των κρυπτονομισμάτων προσπαθούν να εκμεταλλευτούν επενδυτές και επιχειρήσεις ώστε να βγάλουν χρηματικό κέρδος και να αυξήσουν τις οικονομίες τους.

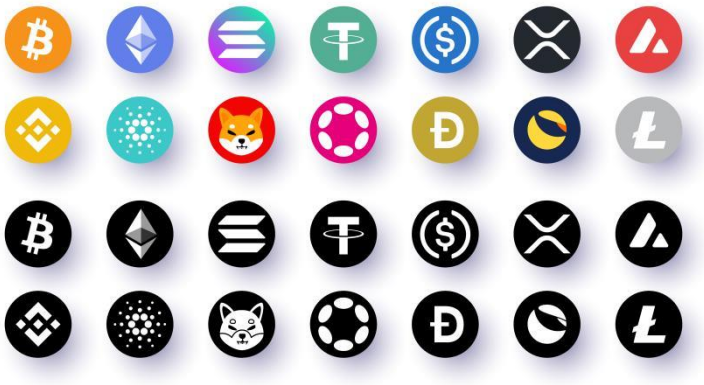
Για την επίτευξη χρηματικού κέρδους οι επενδυτές χρησιμοποιούν οικονομικές μεθόδους χρηματιστηρίου πάνω σε ιστοσελίδες που παρακολουθούν την τιμή των κρυπτονομισμάτων και περιέχουν στατιστικά στοιχεία και διαγράμματα για αυτά.

Η παρούσα διπλωματική εργασία αναφέρεται στην ανάπτυξη μίας διαδικτυακής εφαρμογής για την παρακολούθηση της τιμής των κρυπτονομισμάτων. Παρέχει στον χρήστη στατιστικά στοιχεία και διαγράμματα για την μελέτη-ανάλυση μιας γκάμας πάνω από δέκα χιλιάδες κρυπτονομισμάτων. Επίσης επιτρέπει την αποθήκευση των κρυπτονομισμάτων που επιθυμεί ο χρήστης για την ευκολότερη και γρηγορότερη καθημερινή ανάλυση τους. Σκοπός της εργασίας είναι να βοηθήσει τον χρήστη να πάρει πληροφορίες για οποιοδήποτε κρυπτονόμισμα ή ανταλλακτήριο (market) αλλά και να λειτουργεί ως το δεξί του χέρι πριν και αφού επενδύσει σε κάποιο κρυπτονόμισμα.

1.2 Περιγραφή των κεφαλαίων

Αρχικά θα παρουσιαστεί μια αντιπροσωπευτική επεξήγηση της έννοιας και της δράσης των κρυπτονομισμάτων. Έπειτα θα γίνει μια όσο το δυνατόν πιο επεξηγηματική ανάλυση των τεχνολογιών που χρησιμοποιήθηκαν για την κάλυψη του θεωρητικού υπόβαθρου που θα χρειαστεί για την ανάπτυξη της εφαρμογής. Στη συνέχεια θα παρουσιαστεί μια εκτενής ανάλυση και σχολιασμός διάφορων τμημάτων κώδικα για το σχεδιασμό υλοποίησης της εφαρμογής και έπειτα μια παρουσίαση της από την οπτική ενός χρήστη. Τέλος, θα αναφερθούν τα συμπεράσματα και οι μελλοντικές βελτιώσεις ως τρόποι επέκτασης της υπάρχουσας εφαρμογής.

Κεφάλαιο 2ο: Η έννοια των κρυπτονομισμάτων



2.1: Εξώφυλλο Κρυπτονομισμάτων

2.1 Ιστορική Αναδρομή

Η έννοια του κρυπτονομίσματος πρωτοεμφανίστηκε δημόσια το 1998 από τον Wei Dai στη πλατφόρμα αλληλογραφίας cypherpunks. Η καινοτόμο ιδέα του Wei Dai αναφερόταν στην δημιουργία ενός νέου χρήματος το οποίο θα χρησιμοποιεί την κρυπτογραφία για τον έλεγχο και την ασφάλεια των συναλλαγών. Έτσι, μετά από 10 χρόνια, την διάρκεια του 2008-2009, εμφανίζεται η πρώτη σχετική εφαρμογή της τεχνολογίας των κρυπτονομισμάτων μέσα από ένα άρθρο με όνομα «Bitcoin: ένα ομότιμο σύστημα ηλεκτρονικού χρήματος». Το συγκεκριμένο άρθρο δημοσιοποιήθηκε από έναν προγραμματιστή η μια ομάδα προγραμματιστών με την ονομασία Satoshi Nakamoto. Συγκεκριμένα, ο Nakamoto (2008) δημιούργησε το πρώτο κρυπτονόμισμα το οποίο και ονόμασε Bitcoin. Ένα εικονικό ή ψηφιακό νόμισμα που λειτουργεί μέσω της τεχνολογίας «blockchain» και έχει σχεδιαστεί για να λειτουργεί ως μέσο συναλλαγής στο διαδίκτυο για την αγορά αγαθών και υπηρεσιών αλλά και γενικότερα την πραγματοποίηση πληρωμών [1].

Παρακάτω θα δούμε μια σύντομη ιστορική αναδρομή στην δράση και τα γεγονότα που έχουν καταγραφεί από την δημιουργία του bitcoin μέχρι και σήμερα [2]:

• Νοέμβριος 2008. Ο Satoshi Nakatomo δημοσιεύει ένα έγγραφο με όνομα Bitcoin: A Peer-to-Peer Electronic Cash System.

• Ιανουάριος 2009. Δημιουργία του πρώτο μπλοκ συναλλαγής στην τεχνολογία «blockchain» με όνομα «genesis» ξεκινώντας την δημιουργία των bitcoins. Επιπρόσθετα, πραγματοποιείται η πρώτη συναλλαγή ανάμεσα στον Satoshi Nakatomo και στον Hal Finley.

• Μάιος 2010. Ένας προγραμματιστής ονόματι Lasso Hanyecz στέλνει 10.000 btc σε έναν εθελοντή των κρυπτονομισμάτων στην Αγγλία για την αγορά μιας πίτσας αξίας 25 δολαρίων.

• Ιανουάριος 2011. Δημιουργείται το «the silk road». Μια πλατφόρμα του διαδικτύου που επιτρέπει την αγορά ναρκωτικών και άλλων μη αποδεκτών αγαθών από τις κυβερνήσεις χρησιμοποιώντας το bitcoin ως μέσο συναλλαγής.

• Ιούνιος το 2012. Έχουμε την πρώτη μεγάλη κλοπή καθώς ο Alleviant δημιουργός της διαδικτυακής πλατφόρμας bitcoin forum καταγγέλλει ότι 25.000 bitcoin αξίας 375.000 δολάρια χάθηκαν από το

ηλεκτρονικό του πορτοφόλι. Το κενό αυτό στην ασφάλεια του συστήματος ρίχνει την αξία του bitcoin από 17,51 σε 0.01 δολάρια.

• Ιούνιος 2014. Η μαύρη αγορά «the silk road» διαλύεται και αυτό βοηθά στην νομιμότητα του bitcoin καθώς αποδεικνύεται ότι κάποιος δεν μπορεί πλέον να χρησιμοποίει το bitcoin για παράνομες αγορές πωλήσεις χωρίς τη συνέπεια του νόμου.

• Αύγουστος 2015.Ο Διευθυντής της Mt.Gox (εταιρεία ανταλλακτηρίων bitcoins), Mark karpeles συλλαμβάνεται στην Ιαπωνία μετά την απώλεια τεράστιων ποσών.

• Ιανουάριος 2016. Ψηφίζεται ως η καλύτερη, βάσει απόδοσης, οικονομία της χρονιάς.

• Αύγουστος 2017. Εκτόξευση της τιμής σε 19.000 $/btc.

• 2019. 7 μεγάλα χρηματιστήρια κρυπτονομισμάτων έχουν υποστεί επίθεση και έχουν παραβιαστεί.

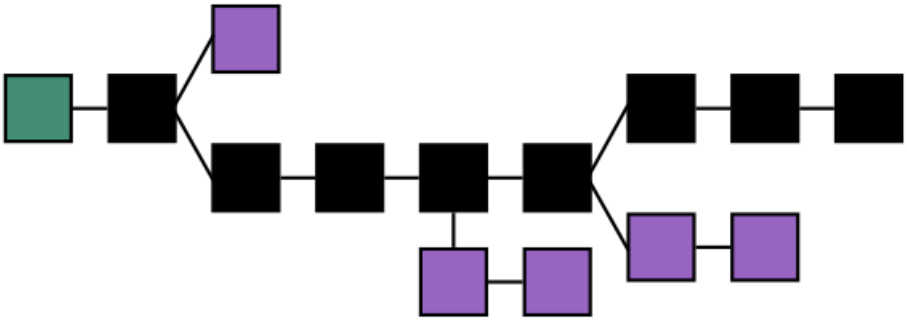
• 2020. Οι επενδύσεις και το εμπόριο των κρυπτονομισμάτων γίνονται τάση λόγω της Πανδημίας

• 2021. Η κεφαλαιοποίηση της αγοράς των κρυπτονομισμάτων ξεπερνά τα 3 τρισεκατομμύρια δολάρια.

2.2 Ο ορισμός των κρυπτονομισμάτων

Όπως έχει ήδη προαναφερθεί, η έννοια του κρυπτονομίσματος ορίζεται το εικονικό ή ψηφιακό νόμισμα που λειτουργεί μέσω της τεχνολογίας «blockchain» και έχει σχεδιαστεί για να λειτουργεί ως μέσο συναλλαγής στο διαδίκτυο για την αγορά αγαθών και υπηρεσιών αλλά και γενικότερα την πραγματοποίηση πληρωμών. Τα κρυπτονομίσματα πήραν την ονομασία τους από την τεχνική την οποία χρησιμοποιούν για την ολοκλήρωση συναλλαγών τους, την κρυπτογραφία, και είναι το πρώτο συνθετικό της λέξης (κρύπτος - νόμισμα). Η κρυπτογραφία χρησιμοποιείται στην επίτευξη της ασφαλούς και με απόλυτο έλεγχο δημιουργίας συναλλαγών αφού η τεχνική αυτή επιτρέπει μόνο στον αποστολέα και στον προβλεπόμενο παραλήπτη ενός μηνύματος να έχει πρόσβαση στο περιεχόμενό του. Ο όρος προήλθε από την ελληνική λέξη κρύπτος, δηλαδή κρυμμένος. [3], [4]

2.3 Η τεχνολογία Blockchain



2.2: Σχήμα Blockchain. Πηγή: [5]

2.3.1 Ο ορισμός του Blockchain

Η τεχνολογία του «Blockchain» δημιουργήθηκε και αναπτύχθηκε στο πεδίο της επιστήμης της πληροφορικής και επεκτάθηκε και στον τομέα των οικονομικών με την εφεύρεση των κρυπτονομισμάτων. Η έννοια του αναπτύσσεται πάνω στα δύο συνθετικά του (Block-Chain) και περιγράφεται σε απλή μορφή ως μια «αλυσίδα» η οποία χρησιμοποιεί τα λεγόμενα «blocks» ώστε να καταγράψει και να αποθηκεύσει πληροφορίες μέσα σε αυτά. Σύμφωνα με το Satoshi Nakamoto (2008),

η αλυσίδα του «Blockchain» παρουσιάζεται ως μια ταξινομημένη back - linked λίστα με ομάδες καταχωρήσεων οι οποίες ονομάζονται «blocks». Τα «blocks» περιέχουν τις πληροφορίες των συναλλαγών που έχουν επιτευχθεί με την χρήση των κρυπτονομισμάτων και η σύνδεση τους γίνεται προς τα πίσω με το καθ' ένα να έχει αναφορά στο προηγούμενο «block» το οποίο βρίσκεται στη λίστα. Για την δημιουργία ενός «block», το σύνολο των εγκεκριμένων συναλλαγών που έχουν ολοκληρωθεί ομαδοποιείται σε αυτό και αποστέλλεται σε όλους τους κόμβους του δικτύου ώστε όλα τα μέλη του δικτύου να μπορούν να έχουν πρόσβαση σε συγκεκριμένες πληροφορίες των συναλλαγών του «block». Τα μέλη του δικτύου με τη σειρά τους ελέγχουν και επικυρώνουν το νέο αυτό «block» καθιστώντας το «Blockchain» ως μια ενιαία πηγή αλήθειας αφού τα μέλη σε ένα δίκτυο μπορούν να δουν τις συναλλαγές που σχετίζονται με αυτά και να τις επικυρώσουν.[6]

2.3.2 Ο τρόπος λειτουργίας του Blockchain

Όπως έχει ήδη προαναφερθεί η δομή των δεδομένων στο Blockchain ορίζεται ως μια ταξινομημένη back – linked λίστα των blocks των συναλλαγών όπου κάθε block συνδέεται με το προηγούμενο του σχηματίζοντας μια αλυσίδα. Παρακάτω παρουσιάζονται τα σημαντικότερα χαρακτηριστικά του τρόπου λειτουργίας του Blockchain.[6], [7]

2.3.2.1 Blocks

Κάθε block εντός του Blockchain αποτελείται από 3 οντότητες. Αρχικά τα blocks προσδιορίζονται από ένα hash (κατακερματισμό) δηλαδή μια συνάρτηση η οποία καλύπτει όλες τις κρυπτογραφημένες απαιτήσεις που ζητούνται για την επίλυση των υπολογισμών που πρέπει να υλοποιήσει το Blockchain. Το hash δημιουργείται με την χρήση του sha256 αλγόριθμου κρυπτογράφησης και περιγράφεται ως ένας μοναδικός κωδικός ο οποίος προσδιορίζει το block και όλα τα περιεχόμενά του. Ακόμη και μια μικροσκοπική μετατροπή στα δεδομένα που υπάρχουν μέσα στο block, θα οδηγήσει σε έναν νέο και εντελώς αγνώριστο hash από το αρχικό. Πιο συγκεκριμένα, το hash είναι μια μονόδρομη συνάρτηση η οποία χρησιμοποιείται μόνο για τον έλεγχο ότι τα δεδομένα του Block που δημιούργησαν το hash είναι τα αρχικά δεδομένα και δεν έχουν υποστεί επεξεργασία. Η δεύτερη σημαντική οντότητα από την οποία αποτελούνται τα blocks είναι το previous block hash (κατακερματισμός του προηγούμενου block), το οποίο χρησιμοποιείται στην ανίχνευση και εγκυρότητα σύνδεσης του προηγούμενου block με το τρέχων δημιουργώντας έτσι μια αλυσίδα η οποία πηγαίνει πίσω σε όλη την διαδρομή μέχρι το πρώτο block που δημιουργήθηκε. Τέλος, η τρίτη οντότητα από την οποία αποτελείται ένα block είναι τα data (δεδομένα), δηλαδή το περιεχόμενο των πληροφοριών των συναλλαγών που έχουν πραγματοποιηθεί μέσα στο block.

2.3.2.2 Peer-To-Peer

Το Peer-To-Peer δίκτυο, επίσης γνωστό και ως P2P είναι ένα μοντέλο αποκεντρωμένης επικοινωνίας μεταξύ των μελών (κόμβων) του δικτύου. Οι κόμβοι είναι μια συστάδα ηλεκτρονικών συσκευών οι οποίες αποθηκεύουν και μοιράζονται τα δεδομένα, έτσι ώστε ο κάθε κόμβος να κρατάει ένα πλήρες αντίγραφο των block του Blockchain και το συγκρίνει με άλλους κόμβους ώστε να εξασφαλίσει την εγκυρότητα των δεδομένων. Ειδικότερα, στις συναλλαγές μεταξύ κρυπτονομισμάτων, η αρχιτεκτονική της αποκέντρωσης που χρησιμοποιεί το δίκτυο επιτρέπει την επίτευξη τους σε οποιοδήποτε σημείο στο κόσμο, χωρίς την ανάγκη να εγκριθεί η συναλλαγή από μια ενδιάμεση αρχή (όπως η τράπεζα), όπου μια τέτοια έγκριση θα μπορούσε να κοστίσει επιπλέον χρεώσεις ή ακόμη και να οδηγήσει στην αποτυχία της συναλλαγής. Τέλος, αναφορικά με την δημιουργία και την εγκυρότητα των συναλλαγών επιδιώκεται ύψιστου βαθμού ασφάλεια διότι αν κάποιος θελήσει να προσθέσει μια συναλλαγή στην

αλυσίδα, η συναλλαγή θα φανερωθεί σε όλους τους κόμβους στο δίκτυο όπου με τη σειρά τους θα την ελέγξουν και θα την επικυρώσουν. Η συναλλαγή θα είναι έγκυρη αν η πλειοψηφία των μελών (κόμβων) του δικτύου συμφωνήσει στην εγκυρότητα της.

2.3.2.3 Mining

Η εξόρυξη στο Blockchain (Blockchain Mining) αναφέρεται στην διαδικασία που χρησιμοποιείται για την δημιουργία νέων κρυπτονομισμάτων αλλά και στην γρήγορη ανταπόκριση και επαλήθευση νέων συναλλαγών. Η εξόρυξη περιλαμβάνει εξορύκτες Blockchain (Blockchain Miners) οι οποίοι χαρακτηρίζονται ως κόμβοι στο δίκτυο του Blockchain. Οι εξορύκτες είναι άτομα, ομάδες ατόμων ή ακόμη και μεγάλες εταιρίες οι οποίοι χρησιμοποιούν ηλεκτρονικό εξοπλισμό (κυρίως υπολογιστές) για να τρέξουν ένα πρόγραμμα του blockchain (Blockchain Mining). Τα προγράμματα αυτά απαιτούν πολύ χρόνο και πολύ μεγάλη υπολογιστική ισχύ για να επιλύσουν τους υπολογισμούς που απαιτούνται. Σε αντάλλαγμα για τη συνεισφορά της επεξεργαστικής τους ισχύος, οι υπολογιστές στο δίκτυο ανταμείβονται με νέα κρυπτονομίσματα. Επιπρόσθετα, με την βοήθεια των εξορύχων λόγω την υψηλής υπολογιστικής ισχύς που παρέχουν στο δίκτυο, το Blockchain διασφαλίζει και επαληθεύει με πολύ μεγάλη ταχύτητα τις συναλλαγές που δημιουργούνται. Τέλος, όσο η αξία κάποιου κρυπτονομίσματος μεγαλώνει, τόσο περισσότεροι εξορύκτες αποσκοπούν να ενταχθούν στο δίκτυο, αυξάνοντας την ισχύ και την ασφάλειά του. Λόγω του όγκου της επεξεργαστικής ισχύος που εμπλέκεται για ένα νόμισμα υψηλής αξίας, καθίσταται πολύ δύσκολο για κάποιο άτομο ή ομάδα να ανακατευτεί με την εξόρυξη στο Blockchain. Για αυτό πολλές εταιρίες ανά τον κόσμο έχουν αναπτυχθεί για να καλύψουν αυτές τις απαιτήσεις.

2.4 Σημαντικότερα Πλεονεκτήματα κρυπτονομισμάτων έναντι παραστατικών

1) Ελευθερία στις συναλλαγές μέσω crypto wallet (πορτοφόλι κρυπτονομισμάτων)

Λόγω της αποκεντρωμένης αρχιτεκτονικής του δικτύου των κρυπτονομισμάτων λόγω του Blockchain, οι χρήστες μπορούν να αναπτύξουν συναλλαγές σε οποιοδήποτε σημείο του κόσμου με πολύ μεγάλη ταχύτητα και χωρίς να χρειάζεται να ελέγξει και στη συνέχεια και να εγκρίνει τη συναλλαγή αυτή μια κεντρική αρχή (όπως η τράπεζα). Αυτό συμβαίνει διότι οι τράπεζες περιορίζουν την ανάπτυξη συναλλαγών με το εξωτερικό με την άσκηση υπέρβασης ορίου μεταφοράς και με την χρήση υψηλών χρεώσεων. Από την άλλη, με την χρήση των κρυπτονομισμάτων, ο χρήστης δημιουργώντας ένα διαδικτυακό πορτοφόλι στο ίντερνετ διατηρεί τον πλήρη έλεγχο των χρημάτων του, χωρίς συμφωνίες, υπογραφές και περιορισμούς. Δεν υπάρχει όριο μεταφοράς χρημάτων και δημιουργεί τις συναλλαγές του χωρίς καθυστερήσεις ή επιπλέον χρεώσεις για διεθνείς αγορές.[8]

2) Έλεγχος και ασφάλεια συναλλαγών

Στην τεχνολογία Blockchain, όπως έχει ήδη προαναφερθεί, όλες οι συναλλαγές που έχουν πραγματοποιηθεί είναι διαθέσιμες προς τους χρήστες. Αυτό σημαίνει πως οι χρήστες γνωρίζουν ακόμη και τις συναλλαγές που έχουν ολοκληρωθεί μεταξύ εμπόρων και προμηθευτών καθιστώντας ανέφικτο για τους εμπόρους να χρεώσουν υπερβολικά την τιμή ενός προϊόντος ή υπηρεσίας χωρίς να το παρατηρήσει κάποιος χρήστης. Επιπλέον, λόγω της αρχιτεκτονικής του Blockchain, όπως αναφέρθηκε παραπάνω βάσει των συστατικών του τρόπου λειτουργίας του, δηλαδή με την χρήση των κρυπτογραφημένων blocks, του δικτύου Peer To Peer με την βοήθεια των εξορύχων (miners) , οι συναλλαγές κρυπτονομισμάτων δεν μπορούν να χειραγωγηθούν - παραβιαστούν από κάποιο άτομο, οργάνωση ή οποιαδήποτε αρχή (τράπεζα, κυβέρνηση). [8]

2.5 Επενδύσεις πάνω στα κρυπτονομίσματα

Λόγω της συνεχούς εξελισσόμενης τεχνολογίας αλλά και ιδεολογίας στην οποία αναπτύσσονται τα κρυπτονομίσματα, όλο και περισσότεροι άνθρωποι αλλά και εταιρίες επιχειρούν να επενδύσουν πάνω τους. Ωστόσο, ακόμη και σήμερα, σημειώνεται μεγάλο ποσοστό ανθρώπων που αποφεύγουν την χρησιμοποίηση κρυπτονομισμάτων στις συναλλαγές τους. Αυτό συμβαίνει γιατί ο κόσμος αδυνατεί να καταλάβει επακριβώς την πολύπλοκη τεχνολογία του και επομένως τα αξιοσημείωτα οφέλη που μπορεί να επιτύχει. Για αυτό τον λόγο ενώ παρατηρείται σταδιακή αύξηση ανά έτος στην αξία τους, υπάρχουν περίοδοι που η αξία τους μειώνεται σημαντικά. Τις μεγάλες αυτές αυξομειώσεις που δημιουργούνται στο εμπόριο των κρυπτονομισμάτων προσπαθούν να εκμεταλλευτούν επενδυτές και επιχειρήσεις οι οποίοι ακόμη και αν δεν γνωρίζουν για την τεχνολογία, χρησιμοποιούν την εμπειρία και άλλες μεθόδους που έχουν αναπτύξει με την ενασχόληση τους με το χρηματιστήριο ώστε να βγάλουν χρηματικό κέρδος και να αυξήσουν τις οικονομίες τους. Τα σημαντικότερα προτερήματα για την επένδυση πάνω στα κρυπτονομίσματα είναι τα παρακάτω:[9]

1) Είναι πολύ ασταθής

Ο σημαντικότερος λόγος για την αστάθεια των κρυπτονομισμάτων είναι η προσφορά και η ζήτηση τους. Για παράδειγμα, ως προς το σχεδιασμό, το bitcoin είναι περιορισμένο στην δημιουργία μέχρι και 21 εκατομμυρίων νομισμάτων. Όσο πιο κοντά βρίσκεται η προσφορά για την αγορά του σε αυτό το όριο, τόσο και υψηλότερη θα είναι η αξία του. Έτσι πολλοί επενδυτές αξιοποιούν αυτήν την ιδιότητα, αγοράζοντας bitcoins ή ποσοστά ενός bitcoin σε όσο το δυνατόν χαμηλότερη τιμή και πουλώντας το όταν η τιμή αυτή αυξηθεί, δημιουργώντας μεγάλα χρηματικά κέρδη.

2) Επένδυση 24/7

Η αγορά κρυπτονομισμάτων βασίζεται σε ανταλλακτήρια (Exchange markets). Σε αντίθεση με τα παραδοσιακά χρηματιστήρια τα οποία έχουν συγκεκριμένη ώρα ανοίγματος και κλεισίματος, τα ανταλλακτήρια είναι διαδικτυακές πλατφόρμες στις οποίες οι χρήστες δημιουργώντας ένα λογαριασμό μπορούν να αγοράσουν και να πωλήσουν κρυπτονομίσματα της αρέσκειας τους οποιαδήποτε ώρα σε οποιαδήποτε μέρα θελήσουν.

2.6 Επίλογος

Στο παραπάνω κεφάλαιο αναλύθηκε η έννοια των κρυπτονομισμάτων, ο ορισμός και η τεχνολογία στην οποία είναι χτισμένα και τα πλεονεκτήματα που παρουσιάζουν έναντι των παραστατικών νομισμάτων. Τέλος, παρουσιάστηκαν οι κύριοι λόγοι για τους οποίους τα κρυπτονομίσματα αποτελούν μία αρκετά φημισμένη επενδυτική δραστηριότητα.

Κεφάλαιο 3ο: Θεωρητικό Υπόβαθρο

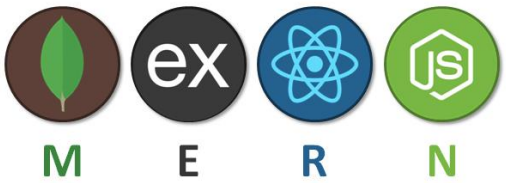
3.1 Εισαγωγή

3.1.1 Προγραμματισμός Διαδικτύου (web development)

Ο προγραμματισμός διαδικτύου αναφέρεται στην εργασία που απαιτείται για την ανάπτυξη ενός ιστότοπου στο Διαδίκτυο (World Wide Web) ή σε ένα ιδιωτικό δίκτυο. Η εργασία που ζητείται για την ανάπτυξη ιστού κυμαίνεται από την δημιουργία μιας απλής στατικής σελίδας, απλού κειμένου που δεν μεταβάλλεται δυναμικά, μέχρι και την κατασκευή διαδικτυακών εφαρμογών διαδικτύου όπως ηλεκτρονικές επιχειρήσεις.

Η παρούσα διπλωματική εργασία αναφέρεται στην δημιουργία μιας πλήρους διαδικτυακής εφαρμογής που αποσκοπεί στο να παρέχει στους χρήστες την δυνατότητα να παρακολουθούν την εξέλιξη της τιμής των κρυπτονομισμάτων που επιθυμούν. Για την υλοποίηση απαιτείται η συνεργασία τριών τμημάτων προγραμματισμού. Η ιεραρχία ανάπτυξης των τμημάτων αυτών αποτελείται από τον προγραμματισμό από την πλευρά του πελάτη (Front-end), από την πλευρά του διακομιστή (Back-end) αξιοποιώντας το τρίτο τμήμα, την τεχνολογία των βάσεων δεδομένων (database technology). Για την υλοποίηση της εφαρμογής εφαρμόστηκε η στοίβα MERN (MERN Stack) η οποία χρησιμοποιείται ως ένα πλαίσιο τεχνολογιών που συνδυάζονται για την ανάπτυξη των παραπάνω τριών τμημάτων προγραμματισμού.[10]

3.1.2 MERN Stack



3.1: Λογότυπο MERN Stack

Η στοίβα MERN είναι το πλαίσιο των τεχνολογιών που επέλεξα για την ανάπτυξη της εφαρμογής. Παίρνει το όνομα της από τις τεχνολογίες που αποτελείται, MongoDB, Express.js, East και Node.js.

Βασικό στοιχείο της στοίβας MERN είναι ότι όλες οι τεχνολογίες που διαθέτει βασίζονται στην γλώσσα προγραμματισμού JavaScript. Ωστόσο, κάθε τεχνολογία προσφέρει μια διαφορετική λειτουργία. Η React είναι μία βιβλιοθήκη χτισμένη στην JavaScript και αξιοποιείται για τη δημιουργία λειτουργιών από την πλευρά του πελάτη (Front-End) παρέχοντας ειδικά κομμάτια κώδικα που μπορούν να αναπαραχθούν και να επαναχρησιμοποιηθούν όπως κρίνουν οι προγραμματιστές. Η Node.js και η Express.js από την πλευρά του διακομιστή (Back-End), συμβάλλουν στην ανάπτυξη λειτουργιών και διαχείριση των αιτημάτων των χρηστών, επιτρέποντας σε αυτές τις εφαρμογές να εκτελούν κώδικα JavaScript εκτός ενός προγράμματος περιήγησης ιστού. Η MongoDB, είναι η βάση δεδομένων της στοίβας και χρησιμοποιείται για αποθήκευση, ταξινόμηση και επεξεργασία των δεδομένων της εφαρμογής.

Η επιλογή της στοίβας MERN για την εκπόνηση της εργασίας έγινε καθώς οι τεχνολογίες από τις οποίες αποτελείται βασίζονται στην γλώσσα προγραμματισμού JavaScript πάνω στην οποία διέθετα αρκετή εμπειρία. Επιπλέον, η βιβλιοθήκη React ήταν σημαντικός παράγοντας καθώς έχω εργαστεί αρκετά σε αυτή. Τέλος, οι τεχνολογίες αυτές διαθέτουν μία πολύ μεγάλη κοινότητα με αποτέλεσμα να υπάρχει μεγάλη υποστήριξη για οποιοδήποτε πρόβλημα θα προέκυπτε.

Σε αυτό το κεφάλαιο, παρουσιάζονται οι τεχνολογίες, τα εργαλεία και οι βιβλιοθήκες που εφαρμόστηκαν για την εκπόνηση της εργασίας. Πιο συγκεκριμένα, αναλύονται και καταγράφονται αρχικά οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του Front-End τμήματος της εφαρμογής, δηλαδή οι έννοιες που κατανέμονται στον προγραμματισμό διαδικτύου για την δημιουργία της διεπαφής στην οποία αλληλοεπιδρούν οι χρήστες με κύρια τεχνολογία την React, και αφετέρου οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του Back-End τμήματος της εφαρμογής, δηλαδή οι έννοιες που αναπτύσσονται από την πλευρά του διακομιστή και λειτουργούν στο παρασκήνιο για να παρέχουν πληροφορίες στον χρήστη με κύριες τεχνολογίες την Node και Express. Τέλος, περιγράφεται αναλυτικά η βάση δεδομένων MongoDB της στοίβας MERN η οποία χρησιμοποιείται για την αποθήκευση και συντήρηση των δεδομένων.

Καθώς η δημιουργία διαδικτυακών εφαρμογών αποτελεί μια εκτεταμένη διαδικασία και δεδομένου του μεγάλου όγκου των τεχνολογιών που απαιτούνται για την ανάπτυξη των Front-End και Back-End τμημάτων προγραμματισμού, θα δοθεί μεγαλύτερη βαρύτητα στην περιγραφή των τεχνολογιών που αποτελούν το σκελετό της εργασίας και λιγότερο αυτές που εφαρμόστηκαν σε μικρότερο βαθμό ως βοηθητικές. Παρακάτω λοιπόν, παρουσιάζονται οι τεχνολογίες , οι έννοιες και οι λειτουργίες που εφαρμόστηκαν για την εκπλήρωση της εργασίας.[11]–[13]

3.2 Τεχνολογίες Front-End

Η κωδικοποίηση από πλευράς πελάτη (Front-End) αναλαμβάνουν να αποδώσουν την εικόνα και τις λειτουργίες που μπορεί να χρησιμοποιήσει ένας χρήστης που επισκέπτεται έναν ιστότοπο. Η κωδικοποίηση επιτυγχάνεται μέσω των κομματιών κώδικα (scripts) τα οποία αλληλεπιδρούν με στοιχεία που είναι καταγεγραμμένα στην HTML (HyperText Markup Language) του ιστότοπου ώστε να παρέχουν μια διαδραστική εμπειρία στους χρήστες. Επιπρόσθετα, τα στοιχεία της HTML συνδυάζονται με αρχεία CSS (Cascading Style Sheet) τα οποία διαμορφώνουν τον τρόπο εμφάνισης της ιστοσελίδας. Πλέον, υπάρχουν πολλά διαφορετικά πλαίσια (frameworks) τα οποία συνδυάζουν τις παραπάνω τεχνολογίες τα οποία είναι εύκολα στην εκμάθηση και στη συντήρηση και επέκταση της εφαρμογής από πλευράς πελάτη (Front-End). Το framework το οποίο επιλέχθηκε είναι η React.js. Είναι ένα από τα πιο διαδεδομένα frameworks κατά την περίοδο συγγραφής και αξιοποιείται από το πρότυπο της στοίβας MERN, η οποία είναι δομή ανάπτυξης της παρούσας εφαρμογής.[14]

3.2.1 HTML/CSS/JavaScript

3.2.1.1 Εισαγωγή

Η HTML, τα CSS και η JavaScript αποτελούν την ραχοκοκαλιά δημιουργίας ιστού. Εφαρμόζονται στην ανάπτυξη ιστοσελίδων του τμήματος Front-End και απευθύνονται σε αυτό που μπορεί να δει και να κάνει ένας χρήστης κατά την επίσκεψή του σε έναν ιστότοπο. Οι τρεις αυτές τεχνολογίες αναφέρονται ως client side γλώσσες καθώς για την εκτέλεση τους απαιτείται ένα πρόγραμμα περιήγησης όπως (Google Chrome, Firefox κλπ.). Το πρόγραμμα περιήγησης είναι προγραμματισμένο στο να μεταφράζει την κωδικοποίηση των σελίδων αυτών. Το αποτέλεσμα αυτής της μετάφρασης, είναι η ιστοσελίδα.[15]

3.2.1.2 HTML

Η HTML είναι μια γλώσσα σήμανσης η οποία χρησιμοποιείται για τη δημιουργία ιστοσελίδων. Το όνομα της προκύπτει από τις έννοιες από τις οποίες αποτελείται (HyperText Markup Language). Η έννοια "Hypertext" αντιπροσωπεύει την παροχή πρόσβασης σε διαφορετικά κείμενα μέσω συνδέσμων ενώ η "Markup" αντικατοπτρίζει την δομή που απαιτείται για την εμφάνιση του κειμένου και επιτυγχάνεται μέσω μίας γλώσσας (Language) σήμανσης. Με πιο απλά λόγια η HTML είναι μία γλώσσα σήμανσης η οποία δημιουργεί ένα έγγραφο κειμένου χρησιμοποιώντας συγκεκριμένες ετικέτες οι οποίες ορίζουν την δομή για την ανάπτυξη ιστοσελίδων. Αυτή η γλώσσα εφαρμόζει τις ετικέτες εύκολα κατανοητές από τον άνθρωπο που λειτουργούν ως σημάνσεις για να σχολιάσουν το κείμενο που παράγεται, έτσι ώστε να γίνεται κατανοητό και να μπορεί να διαβαστεί από ένα μηχάνημα. Το μηχάνημα αυτό είναι ένα πρόγραμμα περιήγησης το οποίο είναι προγραμματισμένο ώστε να χειρίζεται εικόνες, βίντεο, κείμενο και άλλους τύπους περιεχομένου και να το εμφανίζει στην απαιτούμενη μορφή ώστε να μπορεί να αξιοποιηθεί από τους ανθρώπους.[16]

Οι ετικέτες (tags), ορίζονται μέσα σε σύμβολα "< >" ανά ζεύγη, (μικρότερο από) και (μεγαλύτερο από). Για παράδειγμα, για την δημιουργία μιας απλής παραγράφου εφαρμόζονται οι ετικέτες " " ως ετικέτα έναρξης και " " ως ετικέτα λήξης και ανάμεσα από αυτές τοποθετείται το κείμενο.

Η HTML αναπτύχθηκε από τον φυσικό Tim Berners-Lee το 1991. Είναι ο πρώτος προγραμματιστής ιστού και εφευρέτης του WWW (World Wide Web). Με την πάροδο του χρόνου, η HTML εξελίσσεται συνεχώς εκδίδοντας νέες βελτιωμένες εκδόσεις με πρόσθετο περιεχόμενο και δυνατότητες. Η HTML5 εκδόθηκε το 2014 και είναι η πέμπτη και τελευταία ως σήμερα έκδοση της.

3.2.1.3 CSS

Το CSS (Cascading Style Sheet) είναι μία γλώσσα εμφάνισης η οποία χρησιμοποιείται για να παραχωρήσει στυλ σε ένα έγγραφο γραμμένο σε μια γλώσσα σήμανσης, όπως η HTML. Είναι σχεδιασμένη στο να διαχωρίζει σε ένα έγγραφο σήμανσης HTML τον τρόπο παρουσίασης του από το περιεχόμενο του. Τα CSS είναι υπεύθυνα για τα χρώματα, τα εφέ, τις γραμματοσειρές την διάταξη και τοποθέτηση του περιεχομένου και την παράλλαξη του ανάλογα το μέγεθος της οθόνης των χρηστών σε έναν ιστότοπο. Εν ολίγοις, η CSS χρησιμοποιεί τα στοιχεία ή τις ετικέτες που εμπεριέχονται σε μια γλώσσα σήμανσης όπως της HTML για να προσφέρει μια φιλική προς τον άνθρωπο παρουσίαση ιστοσελίδων στα προγράμματα περιήγησης ιστού.[17]

3.2.1.4 JavaScript

Η JavaScript είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού σεναρίων. Τα σενάρια ορίζονται ως κομμάτια προγραμματισμού τα οποία επιτελούν μια λειτουργία. Ενεργοποιούνται άμεσα με το άνοιγμα μια ιστοσελίδας ή έμμεσα όταν προκληθούν διάφορα συμβάντα από την πλευρά των χρηστών. Με τον τρόπο αυτό επιτυγχάνει την βελτίωση της λειτουργικότητας αλλά της δια δραστικότητας μίας ιστοσελίδας με τους χρήστες που την επισκέπτονται. Στην αρχή, εφαρμοζόταν τα προγράμματα περιήγησης ιστού μόνο από την πλευρά του πελάτη (client-side), ωστόσο πλέον οι μηχανές JavaScript ενσωματώνονται σε πολλούς άλλους τύπους λογισμικού. Αξιοποιείται στους διακομιστές διαδικτύου (web servers) προσφέροντας (server-side) προγραμματισμό, στις βάσεις δεδομένων (databases) καθώς και σε προγράμματα εκτός δικτύου. Επιπρόσθετα, JavaScript χρησιμοποιείται στη σύνταξη εφαρμογών για κινητά τηλέφωνα. Για τον λόγο αυτό, παράλληλα με την HTML και το CSS, η JavaScript αποτελεί βασική τεχνολογία του World Wide Web. Επιπρόσθετα, τα πιο διαδεδομένα προγράμματα περιήγησης του Διαδικτύου διαθέτουν μια αποκλειστική μηχανή η οποία είναι σχεδιασμένη στο να διαβάζει με

ταχύτητα τον κώδικα JavaScript για να την εκτελέσουν,όπως είναι η V8 JavaScript Engine του Google Chrome.[18]

3.2.2 JSON

Το JSON (JavaScript Object Notation) ορίζεται ως ένα πρότυπο μεταφοράς δεδομένων μεταξύ των τμημάτων της διεπαφής χρήστη (Front-End) και της εφαρμογής διακομιστή (Back-End) μιας διαδικτυακής εφαρμογής. Ουσιαστικά, αντιπροσωπεύει έναν τύπο κειμένου ο οποίος χρησιμοποιεί ειδικές συμβάσεις ώστε να μπορεί να αξιοποιείται σε όλες τις γλώσσες προγραμματισμού. Διαθέτει δύο τρόπους για την αποτύπωση του.[19]

1) Ζεύγη ονόματος/τιμής: Σε διαφορετικές γλώσσες προγραμματισμού χρησιμοποιείται ως object, record, hash table κ.α.
2) Ταξινομημένη λίστα που περιέχει τιμές ή ζεύγη ονόματος/τιμής: Σε διαφορετικές γλώσσες προγραμματισμού χρησιμοποιείται ως array, vector, list κ.α.

3.2.3 NPM

Το NPM (Node Package Manager) είναι το μεγαλύτερο μητρώο παροχής λογισμικού στον κόσμο. Είναι γραμμένο εξ ολοκλήρου στην γλώσσα προγραμματισμού JavaScript και εφευρέθηκε από τον Isaac Z. Schlueter. Πιο συγκεκριμένα, είναι ένα πρόγραμμα το οποίο διαχειρίζεται πακέτα λογισμικού για τη γλώσσα προγραμματισμού JavaScript διαθέτοντας έναν client γραμμής εντολών και μια ηλεκτρονική βάση δεδομένων για την παροχή δημόσιων αλλά και για ιδιωτικών πακέτων. Ο client γραμμής εντολών είναι προγραμματισμένος να αλληλεπίδραση με την εν λόγω ηλεκτρονική βάση δεδομένων προσφέροντας την εγκατάσταση πακέτων, την διαχείριση και συντήρηση τους. Ως προς το περιβάλλον προγραμματισμού, παρουσιάζεται ως ένας φάκελος που ονομάζεται “package.json” και διαθέτει όλα τα πακέτα τα οποία έχουν εγκατασταθεί και μπορούν να τεθούν σε λειτουργία.[20]

Για την εκπόνηση της Διπλωματικής εργασίας χρησιμοποιήθηκαν αρκετές βιβλιοθήκες με την βοήθεια του NPM (Node Package Manager) οι οποίες θα αναλυθούν παρακάτω στην περιγραφή της αρχιτεκτονικής της διαδικτυακής εφαρμογής.

3.2.4 React.js

3.2.4.1 Εισαγωγή

Η τεχνολογία της React(.js) είναι μια βιβλιοθήκη ανοιχτού κώδικα JavaScript που χρησιμοποιείται από την πλευρά του πελάτη (Front-End) για την ανάπτυξη διεπαφών χρήστη. Εφευρέθηκε από την ομάδα του Facebook και συνεχίζει να διατηρείται και να εξελίσσεται από αυτήν και από μια τεράστια κοινότητα προγραμματιστών και εταιρειών που την χρησιμοποιούν. Βασικό χαρακτηριστικό για τον προγραμματισμό διεπαφών χρήστη με την React είναι η χρήση components (στοιχείων), δηλαδή κομμάτια κώδικα προγραμματισμένα να παράγουν μια συγκεκριμένη λειτουργία.[21]

Η React χρησιμοποιείται ως βάση για την ανάπτυξη σύνθετων διαδικτυακών εφαρμογών διαθέτοντας πρόσθετες βιβλιοθήκες όπως την React Router Dom, η οποία διαθέτει σημαντικές λειτουργίες όπως το state management, τη δρομολόγηση και τον αλληλεπίδρασης με ένα API (Application Programming Interface).

3.2.4.2 Κύρια χαρακτηριστικά της React.js

1) JSX – JavaScript Syntax Extension

Η JSX ορίζεται ως ένας εξελιγμένος τρόπος σύνταξης της γλώσσας προγραμματισμού JavaScript. Το κύριο πλεονέκτημα της JSX είναι προσφέρει την δυνατότητα συγγραφής HTML και JavaScript στο ίδιο αρχείο κώδικα. Η δυνατότητα αυτή καθιστά ευκολότερο τον κώδικα ως προς το να κατανοηθεί και να εντοπιστούν σφάλματα.[22], [23]

```
const name= “Konstantinos”;

const greet = <h2> Hello {name}! </h2>
```

Η μεταβλητή “greet” δεν είναι ούτε string ούτε HTML. Αντί αυτού, γίνεται ενσωμάτωση κώδικα HTML σε κώδικα JavaScript.

2) Virtual DOM

Το DOM (Document Object Model) ενός εγγράφου XML ή HTML χαρακτηρίζεται ως μία δομή δέντρου η οποία περιέχει κόμβους. Κάθε κόμβος αναπαριστάται ως ένα αντικείμενο που αντιπροσωπεύει ένα τμήμα του συνολικού εγγράφου. Όταν παρουσιαστεί κάποια αλλαγή στην κατάσταση ενός αντικειμένου, τότε DOM ενημερώνει το αντικείμενο που τροποποιήθηκε αλλά μαζί με αυτό ενημερώνει και όλα τα υπόλοιπα αντικείμενα, ακόμη και αν το περιεχόμενο τους δεν έχει υποστεί κάποια επεξεργασία. Η λειτουργία αυτή καθίσταται αρκετά αργή για την δημιουργία μιας σύνθετης εφαρμογής.

Η React ως τεχνολογία δε λειτουργεί άμεσα στο DOM (Document Object Model) του προγράμματος περιήγησης, αλλά σε ένα εικονικό DOM το οποίο ονομάζεται virtual DOM ή αλλιώς VDOM. Το VDOM αποθηκεύεται εξ ολοκλήρου στη μνήμη. Όταν επιτευχθεί κάποια αλλαγή στη κατάσταση ενός αντικειμένου, το VDOM ενημερώνει μόνο το συγκεκριμένο αντικείμενο στο πραγματικό DOM αντί να ενημερώνει και όλα τα υπόλοιπα. Δηλαδή, όλες οι αλλαγές στα αντικείμενα γίνονται αρχικά στο VDOM, έπειτα συγκρίνονται με το αρχικό DOM και στο τέλος ενημερώνονται μόνο τα αντικείμενα του αρχικού DOM τα οποία έχουν υποστεί επεξεργασία. Το VDOM καταναλώνει πολύ λιγότερο χρόνο στη CPU και βοηθά στην βελτιστοποίηση και γρήγορη ενημέρωση των αντικειμένων του εκάστοτε εγγράφου.[22], [23]

3) Components

Ένα βασικό χαρακτηριστικό της δομής ανάπτυξης μια εφαρμογής με την χρήση της React είναι η δυνατότητα χρήσης components. Τα components είναι επαναχρησιμοποιήσιμα κομμάτια κώδικα που μπορούν να επεξεργαστούν ξεχωριστά και αντιπροσωπεύουν τμήματα της διεπαφής χρήστη. Διαθέτουν προσαρμοσμένα στοιχεία HTML και μπορούν να επαναχρησιμοποιηθούν όσες φορές χρειαστεί ή και να χρησιμοποιηθούν σε άλλο component. Για την εκπόνηση της εργασίας εφαρμόστηκαν function components. Δηλαδή, απλές συναρτήσεις JavaScript οι οποίες μπορούν να δεχτούν props και επιστρέφουν στοιχεία React (React Elements).[22]–[24]

4) useState

Το state είναι ένα αντικείμενο το οποίο προσφέρει η React (built-in React object) και λειτουργεί ως μεταβλητή η οποία περιέχει δεδομένα ή πληροφορίες σχετικά με το component στο οποίο υλοποιείται. Αποτελείται από ένα ζεύγος κλειδιού-τιμής όπου το πρώτο κλειδί χρησιμοποιείται για να επιστρέψει την τιμή του, ενώ το δεύτερο για την επεξεργασία του. Το state ενός component μπορεί να υποστεί επεξεργασία. Όταν συμβεί αυτό, το component στο οποίο εμπεριέχεται το state γίνεται re-render, δηλαδή κάνει μια νέα εμφάνιση στο πρόγραμμα περιήγησης με την νέα τιμή. Η αλλαγή του state μπορεί να συμβεί ως αποτέλεσμα σε ενέργειες του χρήστη ή σε συμβάντα που δημιουργούνται αυτόματα από το σύστημα. [22], [25]

5) Props

Τα props είναι συντομογραφία των properties. Είναι επίσης είναι ένα αντικείμενο το οποίο προσφέρει η React (built-in React object) που χρησιμοποιείται στο να αποθηκεύει τιμές να περνάει δεδομένα από ένα γονικό στοιχείο στα θυγατρικά του. Πιο συγκεκριμένα, παρέχει τον τρόπο μεταφοράς δεδομένων από ένα component σε άλλα με τον ίδιο τρόπο που περνάμε ορίσματα σε όπως σε μια συνάρτηση. Τα δεδομένα που περνάνε στα θυγατρικά components ως props παρόλο δεν μπορούν να αλλάξουν. [22]

6) Hooks

Τα Hooks είναι μέθοδοι οι οποίοι αξιοποιούνται για να προσφέρουν σε components επιπλέον δυνατότητες και λειτουργίες. Ως προς τη σύνταξη τους, δέχονται κάποιες παραμέτρους όπως ένα αντικείμενο (object), μία τιμή ή ακόμη και μία άλλη μέθοδο και επιστρέφουν αντίστοιχα ένα αντικείμενο ή και μια λίστα αντικειμένων ή μία συγκεκριμένη τιμή. Η βιβλιοθήκη της React παρέχει built-in hooks όπως το useState. Επιπλέον, υπάρχουν περισσότερες βιβλιοθήκες της React για την αξιοποίηση των hook όπως η react-router-dom. Πέρα από τα hooks που παρέχονται μέσω της βιβλιοθήκης, υπάρχει δυνατότητα και για δημιουργία hook από τον προγραμματιστή (custom) σύμφωνα με τις εκάστοτε ανάγκες του. Κατά την δημιουργία ενός hook (custom hook) υπάρχει δυνατότητα χρησιμοποίησης των ήδη υπάρχων hooks και επέκταση τους.[22], [25], [26]

7) Local Storage

Στην εργασία αυτή έγινε χρήση του Local Storage, ενός απλού μηχανισμού αποθήκευσης ο οποίος χρησιμοποιεί ένα σύστημα κλειδιού/τιμής για την αποθήκευση δεδομένων. Αποθηκεύει απλές τιμές αλλά και σύνθετα δεδομένα με την χρήση JSON. Στη συγκεκριμένη εργασία αξιοποιήθηκε για να κρατάει σε σύνδεση τους εγγεγραμμένους χρήστες της εφαρμογής.[27]

8) Context

Το Context ορίζεται ως ένα πλαίσιο διαχείρισης δεδομένων το οποίο προσφέρει η React και επιτρέπει την παροχή καθολικών δεδομένων σε θυγατρικά components ανεξάρτητα από το πόσο χαμηλά έχουν τοποθετηθεί σε ένα στο δέντρο από components. Το δέντρο από components είναι μία δομή κόμβων που διατηρεί την ιεραρχία γονέα-παιδιού και κάθε κόμβος αντιπροσωπεύει ένα component. Η βασική ιδέα της ανάπτυξης του context είναι ότι λύνει το πρόβλημα μετάδοσης των props μεταξύ των components παρέχοντας καθολικά δεδομένα που ο κάθε component μπορεί να χρησιμοποιήσει απευθείας. Με λίγα λόγια, το context διαχειρίζεται με ευκολία τα δεδομένα σε μια σύνθετη εφαρμογή, αντί αυτού, η μετάδοση συνεχόμενων props καθιστά την εφαρμογή δύσκολή ως προς την διαχείριση.

Η αξιοποίηση του context στο περιβάλλον της React προϋποθέτει την ύπαρξη τριών βημάτων: την δημιουργία (create), παροχή (provide) και κατανάλωση (consume) του.[22], [28]

Δημιουργία του Context

Για την δημιουργία του Context, η React προσφέρει την εργοστασιακή συνάρτηση (built-in React Function) createContext(προεπιλογή). Το όρισμα "προεπιλογή" είναι προαιρετικό και αντιπροσωπεύει την προεπιλεγμένη τιμή του.

```
import {createContext} from "react";

const context = createContext("value");
```

Παροχή του Context

Ο τρόπος με τον οποίο παρέχονται τα δεδομένα από το Context στα επιμέρους θυγατρικά components γίνεται με την χρήση του Context Provider. Η React προσφέρει το εργοστασιακό αντικείμενο (built-in React Object) Context.Provider. Ένα κρίσιμο στοιχείο για την αξιοποίηση του, είναι ότι για να ανακτήσουν τα components τα δεδομένα του Context, πρέπει να είναι τυλιγμένα μέσα στον πάροχο Context.Provider. Η τιμή "value" αναφέρεται στα δεδομένα που μπορεί να επιστρέψει το Context. Για την τροποποίηση των δεδομένων του Context απαιτείται η χρησιμοποίηση ειδικών συναρτήσεων (Actions) οι οποίες συμπεριλαμβάνονται στο "value" και επιτελούν μια συγκεκριμένη λειτουργία.

```
function Main () {
	const context-value = "My new context value";
	return (
	<Context.Provider value={context-value}>
		<AComponent />
	</Context.Provider>
	);
}
```

Κατανάλωση του Context

Για την διαχείριση των δεδομένων του Context που παρέχει ο Context Provider, η React προσφέρει μια επιπλέον εργοστασιακή συνάρτηση (built-in React Function), την useContext.

Η συνάρτηση επιστρέφει την τιμή του context και φροντίζει επίσης να αποδώσει ξανά το νέο στοιχείο στο περιβάλλον του χρήστη αν προηγηθεί οποιαδήποτε μεταβολή στην τιμή του.

```
import {useContext} from "react"
function AComponent() {
	const context-value = useContext(Context);
	return <div> {context-value} </div>;
}
```

3.2.4.3 Σημαντικότερες βοηθητικές βιβλιοθήκες

1) React Query

Η βασική προσέγγιση της React είναι η εφαρμογή του fetch API, δηλαδή η ανάκτηση στοιχείων μέσω του προγράμματος περιήγησης και έπειτα η διαχείριση της απόκρισης αυτής από την εργοστασιακή συνάρτηση “useEffect”. Η συνάρτηση αυτή λειτουργεί ορθά για απλές ανακτήσεις δεδομένων αλλά χάνει εύκολα τον έλεγχο και γίνεται πολύπλοκη όταν απαιτείται να προσωρινή αποθήκευση (cache) και πλήρης συγχρονισμός των δεδομένων. Για την επίλυση των παραπάνω λόγων δημιουργήθηκε η React Query.

Η React Query είναι μία βιβλιοθήκη η οποία αξιοποιείται από την React και στοχεύει στην απλοποίηση του τρόπου ανάκτησης (fetch), προσωρινής αποθήκευσης (cache) και επίτευξης πλήρη συγχρονισμού των δεδομένων που παρέχονται από ένα διακομιστή (server). Εγκαθίσταται στην εφαρμογή πελάτη με την βοήθεια του λογισμικού NPM. Ως προς τον τρόπο χρησιμοποίησης της, αρχικά απαιτείται η δημιουργία της οντότητας “query-client” και η εφαρμογή της πάνω σε ένα σύνολο στοιχείων (components) έτσι ώστε τα στοιχεία αυτά να μπορούν να εφαρμόσουν τις λειτουργίες της React Query. Παρακάτω παρουσιάζεται ένα απλό παράδειγμα με χρήση κώδικα.

```
const queryClient = new QueryClient();
function App() {
 return (
  <QueryClientProvider client={queryClient}>
   <componentA />
  </QueryClientProvider>
 )
}
```

Για την εφαρμογή της React Query εντός ενός στοιχείου (components) για την ανάκτηση δεδομένων από ένα διακομιστή, απαιτείται η χρησιμοποίηση της συνάρτησης “useQuery” η οποία δέχεται ως παραμέτρους το όνομα και τη συνάρτηση της ανάκτησης. Το όνομα είναι ένα κλειδί το οποίο μπορεί να χρησιμοποιηθεί αργότερα για κάποια επανάκτηση και συγχρονισμό δεδομένων και η συνάρτηση είναι η μέθοδος η οποία ζητάει τα δεδομένα από μια διαδρομή διακομιστή. Επιπλέον, η συνάρτηση “useQuery” διαθέτει χρήσιμα στοιχεία για την ανάκτηση όπως ο χρόνος φόρτωσης (isLoading) των δεδομένων και η κατάσταση στην οποία έχει προκληθεί κάποιο σφάλμα (isError). Παρακάτω παρουσιάζεται ένα απλό παράδειγμα με χρήση κώδικα.[29]

```
const {
  isLoading: Loading,
  isError: Error,
  data: boats
  } = useQuery(["crypto-chart"],fetchBoats);

if (Loading) {
  return <p> Loading </p>;
}
if (Error) {
  return <p> Error </p>;
}
```

2) MUI (Material UI)

Το Material UI (Material User Interface) είναι μια βιβλιοθήκη της React ανοιχτού κώδικα η οποία προσφέρει στοιχεία (components) τα οποία χρησιμοποιούνται στο σχεδιασμό ιστοσελίδων που εμφανίζονται στο διαδίκτυο. Ουσιαστικά, τα στοιχεία αυτά αποτελούν τον τρόπο εμφάνισης και σχεδιασμού του περιεχομένου της διεπαφής χρήστη. Διευκολύνουν τον προγραμματιστή να παράγει στοιχεία όπως κουμπιά, πίνακες, πλαίσια κειμένου τα οποία ακολουθούν το Material Design της Google. Η Material UI κυκλοφόρησε για πρώτη φορά το 2014 και χρησιμοποιείται αποκλειστικά για εφαρμογές οι οποίες αναπτύσσονται μέσω της React.[30]

3.2.4.4 Επίλογος

Παραπάνω αναλύθηκαν εν συντομία ο ορισμός και η δράση της React για την δημιουργία διεπαφών χρήστη από την πλευρά προγραμματισμού του πελάτη (Front-End). Η ανάλυση εστιάστηκε στα κύρια χαρακτηριστικά της React ως προς τον τρόπο λειτουργίας και δράσης της για την ανάπτυξη διεπαφών χρήστη και στις επιμέρους βιβλιοθήκες που εφαρμόστηκαν για την εκπόνηση της παρούσας διπλωματικής εργασίας.

3.3 Τεχνολογίες Back-End

Η κωδικοποίηση από την πλευρά του διακομιστή (Back-End) αναφέρεται στον τρόπο επικοινωνίας της εφαρμογής με την βάση δεδομένων και τον ίδιο τον διακομιστή (server). Πιο συγκεκριμένα, το Back-End είναι ένα λογισμικό το οποίο περιλαμβάνει κομμάτια κώδικα τα οποία επικοινωνούν άμεσα με την βάση δεδομένων για την αποθήκευση και τροποποίηση των δεδομένων της. Επιπρόσθετα, παράγουν προσαρμοσμένες απαντήσεις σε κάθε έτοιμα που δημιουργείται από τον χρήστη σε έναν ιστότοπο. Πολλά κομμάτια κώδικα εκτελούνται από τον διακομιστή κάθε φορά που γίνεται πρόσβαση από ένα χρήστη στη σελίδα ενός ιστότοπου. Οι σελίδες αυτές ονομάζονται δυναμικές διότι διαθέτουν την δυνατότητα να δημιουργούν και να τροποποιούν συνεχώς το περιεχόμενο τους. Ένας back-end διακομιστής μπορεί να χρησιμοποιηθεί για πολλές χρήσιμες λειτουργίες.[31]

Μερικές από αυτές είναι οι παρακάτω :

- Αποθηκεύει και διαμερίζει τα δεδομένα.
- Επικοινωνεί και ανταλλάζει δεδομένα μέσω αιτημάτων με την κύρια εφαρμογή.

- Παρέχει χρήσιμα στατιστικά σχετικά με τα αποθηκευμένα δεδομένα.

Η κωδικοποίηση από την πλευρά του διακομιστή (Back-End) αποτελείται από τρία κύρια μέρη :

- Τον διακομιστή , δηλαδή τον υπολογιστή που δέχεται αιτήματα .
- Την βάση δεδομένων στην οποία αποθηκεύονται και συγκροτούνται τα δεδομένα .
- Την εφαρμογή. Δηλαδή, το λογισμικό το οποίο εκτελείται μέσα στον διακομιστή και είναι υπεύθυνο για την παραλαβή και απάντηση των εισερχόμενων αιτημάτων. Το λογισμικό αυτό αποσκοπεί στην ανάκτηση και αποστολή δεδομένων από την βάση δεδομένων όταν το έτοιμα θεωρείται ορθό και στην εμφάνιση μηνύματος λάθους στην περίπτωση όπου το έτοιμα δεν θεωρείται αποδεκτό.

Για την υλοποίηση του Back-End προγραμματισμού επιλέχθηκε η δημιουργία ενός REST API δηλαδή ενός διακομιστή ιστού με τη χρήση της αρχιτεκτονικής RE.S.T. (Μεταφορά αναπαραστατικής κατάστασης). Για την δημιουργία της εφαρμογής διακομιστή χρησιμοποιήθηκε η πλατφόρμα Node js και για την ανάπτυξη της βάσης δεδομένων, η MongoDB. Οι παραπάνω τεχνολογίες αποτελούν το σκελετό της στοίβας MERN ως προς τον προγραμματισμό του Back-End, μέσω της οποίας αναπτύχθηκε η παρούσα διαδικτυακή εφαρμογή.

3.3.1 Rest API (Representational State Transfer Application Programming Interface)

3.3.1.1 API (Application Programming Interface)

Αντικείμενο της Διεπαφή Προγραμματισμού Εφαρμογών (API) χαρακτηρίζεται η δημιουργία μίας διεπαφής σε ένα λειτουργικό σύστημα, η οποία περιέχει προγραμματιστικές λειτουργίες με την μορφή μίας βιβλιοθήκης ή μίας εφαρμογής και είναι σχεδιασμένη στο να δέχεται και να απαντάει σε αιτήματα τα οποία πραγματοποιούνται από άλλα προγράμματα. Ο βασικός λόγος δημιουργίας ενός API να καθορίζει και να συντάσσει το σύνολο των προγραμματιστικών λειτουργιών που παρέχει η βιβλιοθήκη ή η εφαρμογή του, σε άλλα προγράμματα, δημιουργώντας μια επικοινωνία με την μορφή αιτημάτων προς απάντηση.[32]

3.3.1.2 Υπηρεσίες Ιστού (Web Services)

Για την αξιοποίηση ενός API απαιτείται η δυνατότητα επικοινωνίας μεταξύ της εφαρμογής του διακομιστή με άλλες εφαρμογές οι οποίες αναπτύσσονται από την πλευρά του πελάτη (Front-End). Η επικοινωνία αυτή μεταξύ της εφαρμογής διακομιστή και της εφαρμογής πελάτη επιτυγχάνεται μέσω των υπηρεσιών ιστού. Υπάρχουν διάφοροι τύποι υπηρεσιών ιστού όπως τα XML-RPC, UDDI, SOAP και REST και αποτελούν μια μονάδα αρχιτεκτονικής για τον Παγκόσμιο Ιστό σχεδιασμένη με στόχο την εκτέλεση ενός συγκεκριμένου συνόλου εργασιών. Για την υλοποίηση της εργασίας χρησιμοποιήθηκε ο τύπος υπηρεσίας ιστού REST (Representational State Transfer).[33]

3.3.1.3 REST

Το REST (Representational State Transfer) πρωτοεμφανίστηκε το 2000 από τον Roy Fielding στην ακαδημαϊκή του διπλωματική εργασία με τίτλο «Architectural Styles and the Design of Network-based Software Architectures». Ο ορισμός του αναφέρεται στην αρχιτεκτονική που διαθέτει ώστε να κατευθύνει το σχεδιασμό ανάπτυξης διαδικτυακών υπηρεσιών για τον Παγκόσμιο Ιστό. Η αρχιτεκτονική αυτή στηρίζεται στην παροχή ενός συνόλου περιορισμών για το πως πρέπει να συμπεριφέρεται ο Ιστός κατά την δημιουργία αιτημάτων προς απάντηση από την εφαρμογή πλευράς χρήστη προς την εφαρμογή του διακομιστή. Πιο συγκεκριμένα, δίνεται έμφαση στην ανεξάρτητη

ανάπτυξη και επεκτασιμότητα των στοιχειών και στην ανάπτυξη μίας πολύ-επίπεδης αρχιτεκτονικής που αποσκοπεί στην ενθυλάκωση παλαιών επαναχρησιμοποιήσιμων συστημάτων, στην βελτίωση της ασφάλειας και στην αξιοποίηση της μνήμης cache για την αποθήκευση στοιχείων.

Τα Web APIs τα οποία υπακούν στους περιορισμούς της αρχιτεκτονικής REST αναφέρονται ανεπίσημα και ως RESTful. Τα RESTful Web APIs εφαρμόζουν συνήθως τις μεθόδους HTTP για να αποκτήσουν πρόσβαση σε διάφορους πόρους. Οι πόροι αυτοί αντιπροσωπεύουν κωδικοποιημένα URLs τα οποία μέσω παραμέτρων παράγουν JSON ή XML στοιχεία πραγματοποιώντας έτσι επικοινωνία με τον εκάστοτε διακομιστή και παροχή των δεδομένων. Πιο συγκεκριμένα, η δημιουργία μίας υπηρεσίας RESTful Web, είναι σχεδιασμένη στο να παράγει απαντήσεις στα αιτήματα που γίνονται μέσω συγκεκριμένων URI από εφαρμογές χρήστη. Οι απαντήσεις αντικατοπτρίζουν ένα ωφέλιμο φορτίο συνήθως σε μορφή HTML,XML ή JSON. Το πιο κοινό και ευρέως εφαρμοσμένο πρωτόκολλο για την δημιουργία ερωτημάτων και απαντήσεων είναι το HTTP. Το HTTP παρέχει ειδικά προγραμματισμένες μεθόδους όπως το GET για την ανάκτηση, το POST για την δημιουργία ή αποστολή, το PUT για την επεξεργασία και το DELETE για την διαγραφή των πόρων.[34], [35]

Στόχος της αρχιτεκτονικής του REST είναι η βελτίωση της απόδοσης, της επεκτασιμότητας αλλά και απλότητας χρήσης, της δυνατότητας τροποποίησης και αξιοπιστίας της διαδικτυακής εφαρμογής. Τα συστήματα αυτά προσφέρουν επίσης την δυνατότητα να διαχειρίζονται και να ενημερώνονται ανεξάρτητα από το σύνολο της παραγόμενης διαδικτυακής εφαρμογής, ακόμη και όταν εκτελείται.

3.3.2 Node.js

3.2: Λογότυπο Node.js

Το Node.js χαρακτηρίζεται ως μια πλατφόρμα ανάπτυξης λογισμικού ανοιχτού κώδικα. Χρησιμοποιεί την γλώσσα προγραμματισμού JavaScript και αξιοποιείται από τους προγραμματιστές για την δημιουργία διαδικτυακών εφαρμογών από την πλευρά του διακομιστή. Πιο συγκεκριμένα, η πλατφόρμα της Node επιτρέπει την ανάπτυξη σεναρίων προγραμματισμού από την πλευρά του διακομιστή (server-side scripting) αλλά και εργαλείων γραμμής εντολών (command line tools) με σκοπό την παραγωγή δυναμικού περιεχομένου μιας ιστοσελίδας. Μέσω της Node, οι προγραμματιστές μπορούν να έχουν πρόσβαση και να επεξεργάζονται αυτό το περιεχόμενο πριν αλλά και κατά την διάρκεια της εκτέλεσης της σελίδας αυτής στο πρόγραμμα περιήγησης του χρήστη.

Ένα αξιοσημείωτο στοιχείο το οποίο κάνει την Node js ξεχωριστή είναι ότι έχει κατασκευαστεί στον V8 JavaScript engine του Chrome, έναν open source engine ο οποίος μεταγλωττίζει τον κώδικα JavaScript σε γλώσσα μηχανής. Αυτό καθιστά την Node js πολύ γρήγορη στο χρόνο που απαιτείται για την μεταγλώττιση της, αφού εκτελείται στον περιηγητή μαζί με την JavaScript στο ίδιο engine.

Το κύριο χαρακτηριστικό της αρχιτεκτονικής και ένα από τα μεγαλύτερα πλεονεκτήματα της Node είναι η δυνατότητα της να εκτελεί σενάρια κώδικα (scripts) βάσει γεγονότων (event-driven) ασύγχρονα. Η παροχή αυτής της ασύγχρονης επικοινωνίας προσφέρεται από την γλώσσα προγραμματισμού

JavaScript μέσω των ονομαζόμενων λειτουργιών callback functions. Ειδικότερα, τα callback functions εκτελούνται μόνο όταν εκπληρωθεί κάποιο γεγονός. Όσο ο επεξεργαστής περιμένει να εκπληρωθεί κάποιο γεγονός (πχ διάβασμα αρχείου), παραμένει ελεύθερος χωρίς να παγώνει. Μόλις εκπληρωθεί, καλείται η callback function η οποία κρατάει τον επεξεργαστή απασχολημένο μέχρι να εκτελεστεί η λειτουργία της. Για το λόγο αυτό, η Node μπορεί να διαχειρίζεται πολλές ταυτόχρονες συνδέσεις με την χρήση ενός μόνο επεξεργαστή με ελάχιστο χρόνο αναμονής και μικρή απαίτηση μνήμης.

Η Node.js αναπτύχθηκε το 2009 στην ακαδημαϊκή διατριβή του Ryan Dahl. Έπειτα, χρηματοδοτήθηκε από τον Joyent και συνεχίζει να εξελίσσεται συνεχώς. Σήμερα, η διατήρηση και συντήρηση της καθοδηγούνταν από τον Dahl και από μια τεράστια κοινότητα η οποία έχει δημιουργηθεί και την υποστηρίζει παρέχοντας πάνω από 500.000 πακέτα ανοιχτού κώδικα όπου κάθε προγραμματιστής έχει ελεύθερη πρόσβαση σε αυτά με την χρήση του διαχειριστή πακέτων NPM.[36]

Τα σημαντικότερα πακέτα που αξιοποιήθηκαν για την εκπόνηση της παρούσας εργασίας ως προς τον προγραμματισμό της εφαρμογής από την πλευρά του διακομιστή είναι τα Express js, Mongoose, Bcrypt, JsonWebToken, apiCache και αναλύονται παρακάτω.

3.3.2.1 Express.js

Η δημιουργία ενός λογισμικού από την πλευρά του διακομιστή (Back-End) για την υλοποίηση ενός REST API είναι μια δύσκολη και απαιτητική διαδικασία. Παρόλο που μπορεί να αναπτυχθεί απευθείας μόνο με την χρήση της Node.js, οι προγραμματιστές σε όλο τον κόσμο χρησιμοποιούν ένα πρόσθετο πακέτο το οποίο διευκολύνει την δημιουργία, τη συντήρηση και την επέκταση μίας τέτοιας εφαρμογής. Το πακέτο αυτό, ονομάζεται Express.js και ορίζεται ως ένα Πλαίσιο Εφαρμογών Ιστού, ανοιχτού κώδικα το οποίο επιτρέπει την ανάπτυξη ενδιάμεσου λογισμικού και δρομολόγησης για μια εφαρμογή διακομιστή σε πολύ λιγότερο χρόνο και με πολύ λιγότερο εκτενή κώδικα. Η εγκατάσταση του γίνεται μέσω του NPM και παρέχεται δωρεάν βάσει της άδειας του MIT (MIT License - Massachusetts Institute of Technology).

Το Express είναι εξ ολοκλήρου γραμμένο σε γλώσσα JavaScript. Ο τρόπος λειτουργίας του βασίζεται σε κλήσεις συναρτήσεων middleware, δηλαδή λειτουργίες-σενάρια κώδικα τα οποία μπορούν να έχουν πρόσβαση στο αντικείμενο αιτήματος (request) που δημιουργείται από τους χρήστες, στο αντικείμενο απόκρισης (response) το οποίο παράγεται από την εφαρμογή διακομιστή ως απάντηση προς το αίτημα και στην επόμενη συνάρτηση στον κύκλο αίτησης-απόκρισης της εφαρμογής. Επιπρόσθετα, το Express διαθέτει πολλές βοηθητικές εργοστασιακές (Built-in) μεθόδους για την διαχείριση των HTTP αιτημάτων-απαντήσεων από και προς τον διακομιστή. Όλες αυτές οι δυνατότητες καθιστούν την χρήση του Express πολύ σημαντική για την δημιουργία, τη συντήρηση και την επέκταση μίας εφαρμογής διακομιστή.[37]

3.3.2.2 Mongoose

Το Mongoose είναι μια βιβλιοθήκη ODM (Object Data Modeling) η οποία χρησιμοποιείται στην πλατφόρμα Node.js με στόχο την διευκόλυνση της επικοινωνίας και την διαχείριση των δεδομένων από την βάση δεδομένων MongoDB. Η αρχιτεκτονική του Mongoose ως προς την ανάπτυξη λογισμικού (Back-End) για την διαχείριση των δεδομένων βασίζεται σε δύο αντικείμενα τα οποία διαθέτει, μοντέλα και σχήματα (Models and Schemas). Αρχικά, τα σχήματα ορίζουν την δομή και τα περιεχόμενα τα οποία θα αποθηκευτούν στην βάση δεδομένων. Πιο συγκεκριμένα, κάθε σχήμα αντιπροσωπεύει μια συλλογή εγγραφών (δεδομένων) της MongoDB όπου κάθε έγγραφο διαθέτει την δομή που έχει δηλωθεί στο σχήμα αυτό. Τα μοντέλα περιτυλίγουν τα σχήματα και αντιπροσωπεύουν την διεπαφή την οποία

απαιτείται για την διαχείριση των εγγραφών που βρίσκονται μέσα στην βάση δεδομένων. Η εγκατάσταση του πακέτου Mongoose γίνεται μέσω του NPM και παρέχεται δωρεάν.[38]

3.3.2.3 Bcrypt

Το Bcrypt είναι μια βιβλιοθήκη η οποία αξιοποιείται από την πλατφόρμα Node.js ή οποία χρησιμοποιείται για τον κατακερματισμό (hashing) των κωδικών των χρηστών που δημιουργούν λογαριασμό σε κάποια εφαρμογή. Προσφέρει ασφάλεια από επιθέσεις που έχουν σχέση με την ανάκτηση των κωδικών των χρηστών της εφαρμογής. Ειδικότερα, το Bcrypt κάνει χρήση ενός ειδικά σχεδιασμένου αλγορίθμου ο οποίος ονομάζεται Bcrypt και αποσκοπεί στην κρυπτογράφηση των κωδικών ώστε να αποθηκεύονται στην βάση δεδομένων κρυπτογραφημένοι και να μην μεταφέρονται αυτούσιοι μεταξύ της επικοινωνίας πελάτη-διακομιστή κατά την ανταλλαγή αιτημάτων-απαντήσεων. Για την κρυπτογράφηση των κωδικών το Bcrypt προσφέρει μια εργοστασιακή συνάρτηση η οποία ονομάζεται salt και αναφέρεται στην προσθήκη μιας σειράς χαρακτήρων την οποία υποβάλλει ο προγραμματιστής στον κωδικό του χρήστη πριν συμβεί η κρυπτογράφηση του, ώστε να επιτευχθεί ακόμη μεγαλύτερο επίπεδο ασφάλειας. Επιπλέον διαθέτει την δυνατότητα της αποκρυπτογράφησης των κωδικών ώστε να είναι σε θέση να ελέγξει αν ο κωδικός τον οποίο υποβάλει ο χρήστης είναι ο σωστός και να πραγματοποιηθεί η σύνδεση.[39]

3.3.2.4 JsonWebToken

Το JSON Web Token (JWT) ορίζεται ως ένα ανοιχτό πρότυπο (RFC 7519) το οποίο εφαρμόζει έναν συμπαγή και αυτόνομο τρόπο για την ασφάλεια μετάδοσης πληροφοριών μεταξύ των τμημάτων της διεπαφής χρήστη (Front-End) και της εφαρμογής διακομιστή (Back-End) μιας διαδικτυακής εφαρμογής με την χρήση ενός JSON αντικειμένου. Με απλά λόγια, ο στόχος της χρήσης των JWT είναι η προστασία των περιεχομένων μιας διαδικτυακής εφαρμογής από μη επαληθευμένους χρήστες αλλά και η διατήρηση των ήδη επαληθευμένων χρηστών στην εφαρμογή. Τα αντικείμενα JWT χρησιμοποιούν ένα μυστικό ζεύγος δημόσιου / ιδιωτικού κλειδιού μέσω του αλγόριθμου HMAC για την παραγωγή μίας ψηφιακής υπογραφής η οποία δηλώνει την αξιοπιστία και αυθεντικότητα του αντικειμένου.

Ένα JSON Web Token (JWT) αποτελείται από τρία τμήματα. Τα τμήματα αυτά διαχωρίζονται με τελείες (.) παίρνοντας έτσι την μορφή (xxxxx.yyyyy.zzzzz).

1) Header (Κεφαλίδα)
2) Payload (Φορτίο)
3) Signature (Υπογραφή)

Ο Header (κεφαλίδα) περιέχει μεταδεδομένα σχετικά με το διακριτικό (token) και αποτελείται συνήθως από δύο μέρη: τον τύπο του διακριτικού(JWT), και τον αλγόριθμο υπογραφής (HMAC ή SHA256 ή το RSA).

Το payload (φορτίο) διαθέτει τα δεδομένα που μεταδίδονται όπως η διεύθυνση ενός χρήστη. Είναι σημαντικό να μην υπάρχουν στο φορτίο ευαίσθητου τύπου δεδομένα γιατί είναι εύκολο να ανακτηθούν αν συμβεί κάποια επίθεση στο λογισμικό του διακομιστή.

Η υπογραφή είναι το στοιχείο που χρησιμοποιεί το λογισμικό του διακομιστή για να επαληθεύσει ότι το διακριτικό είναι έγκυρο και ότι δεν έχει παραβιαστεί. Δημιουργείται μέσω κατακερματισμού της κεφαλίδας και του φορτίου με μία μυστική σειρά χαρακτήρων την οποία γνωρίζει μόνο η εφαρμογή του διακομιστή εξασφαλίζοντας έτσι την ασφάλεια της μεταδιδόμενης πληροφορίας μεταξύ των τμημάτων Front-End και Back-End της διαδικτυακής εφαρμογής.[40]

3.3.2.5 ApiCache

Για την εκπόνηση της εργασίας κρίθηκε πολύ σημαντική η εφαρμογή μιας προσωρινής μνήμης για την αποθήκευση των δεδομένων τα οποία δημιουργούνται μέσω αποκρίσεων από την πλευρά του διακομιστή προς τα αιτήματα που πραγματοποιούνται από την πλευρά του πελάτη. Για την υλοποίηση της ενέργειας αυτής, χρησιμοποιήθηκε το πακέτο ApiCache μέσω του NPM, το οποίο αξιοποιεί την μνήμη cache από την πλευρά του διακομιστή και προσφέρει ένα πολύ γρήγορο και εύκολο τρόπο χρήσης της. Πιο συγκεκριμένα, μέσω του ApiCache οι απαντήσεις οι οποίες δημιουργούνται από την πλευρά του διακομιστή, αποθηκεύονται στην μνήμη του ως αντίγραφα για μια συγκεκριμένη χρονική περίοδο. Με τον τρόπο αυτό, αν ζητηθεί ξανά από τον διακομιστή να παράγει δεδομένα προς απάντηση στο ίδιο αίτημα που είχε εισέλθει εντός της χρονικής περιόδου, θα επιστρέψει τα δεδομένα τα οποία έχουν αποθηκευτεί στην μνήμη του χωρίς να δημιουργήσει νέα απάντηση προς αυτό το αίτημα. Η αξιοποίηση της μνήμης cache συνεισφέρει στην μείωση του φόρτου στην βάση δεδομένων και στην βελτίωση της απόδοσης της εφαρμογής ως προς τον χρόνο φόρτωσης των δεδομένων.[41]

3.3.3 Βάση Δεδομένων

Οι βάσεις δεδομένων αντιπροσωπεύουν μια οργανωμένη συλλογή δεδομένων. Χρησιμοποιούνται συστηματικά από τους προγραμματιστές για την διαχείριση μεγάλων συνόλων δεδομένων κατά την υλοποίηση διαδικτυακών εφαρμογών. Με την χρήση των βάσεων δεδομένων, οι προγραμματιστές μπορούν να αποθηκεύσουν, διατηρήσουν, ταξινομήσουν και να αναζητήσουν δεδομένα με μεγάλη ταχύτητα και αποτελεσματικότητα. Διαχωρίζονται σε δύο κύριες κατηγορίες, τις Σχεσιακές και Μη σχεσιακές βάσεις δεδομένων. Οι σχεσιακές βάσεις δεδομένων υποστηρίζουν την αποθήκευση των δεδομένων σε πίνακες. Κάθε γραμμή του πίνακα αντιστοιχεί σε μια εγγραφή η οποία ξεχωρίζει από τις άλλες διαθέτοντας με μια μοναδική τιμή που ονομάζεται πρωτεύων κλειδί. Ονομάζεται σχεσιακή διότι μεταξύ των πινάκων υπάρχει η δυνατότητα δημιουργίας σχέσεων οι οποίες αναπτύσσονται βάσει των κλειδιών αυτών. Από την άλλη οι Μη-Σχεσιακές βάσεις δεδομένων αποτελούνται από μία δομή εγγράφων. Κάθε εγγραφή αποθηκεύεται ως ξεχωριστό έγγραφο το οποίο διαχωρίζεται από μία μοναδική οντότητα δεδομένων (id).[42]

```
{
   “name”:”Konstantinos”,
   “hobby”:”football”
}
```

3.3.3.1 MongoDB

3.3: Λογότυπο MongoDB

Το MongoDB ήρθε στο φως στα μέσα της δεκαετίας του 2000 αποτελώντας ένα δωρεάν, ανοιχτού λογισμικού, συστήματος διαχείρισης βάσεων δεδομένων. Κατατάσσεται στην κατηγορία των Μη-Σχεσιακών βάσεων δεδομένων. Χρησιμοποιεί συλλογές για την αποθήκευση εγγράφων. Τα έγγραφα αποτελούν την δομή με την οποία αποθηκεύονται τα δεδομένα στο MongoDB. Απαρτίζονται με την μορφή ζευγών κλειδιού-τιμής η οποία συμβαδίζει με τον τρόπο με τον οποίο οι προγραμματιστές κατασκευάζουν τις κλάσεις και τα αντικείμενα τους στις αντίστοιχες γλώσσες προγραμματισμού, προσφέροντας έτσι μεγαλύτερη κατανόηση στον τρόπο διαχείρισης τους. Επιπλέον, κάθε έγγραφο ανάλογα με τις ανάγκες του προγραμματιστή μπορεί να διαθέτει διαφορετικό αριθμό πεδίων. Ο ρόλος των συλλογών είναι να αποθηκεύουν και να διατηρούν οργανωμένα τα σύνολα εγγράφων και λειτουργιών. (andreas) Επιπλέον, τα δεδομένα τα οποία εισέρχονται στην βάση αποθηκεύονται σε μορφή BSON (binary JSON). Η μορφή αυτή διευκολύνει πολύ την χρήση της με την πλατφόρμα Node.js αφού τα δεδομένα μπορούν να χρησιμοποιούνται απευθείας, χωρίς δηλαδή να απαιτείται μετατροπή των δεδομένων σε κάποια ειδική μορφή την οποία υποστηρίζει η βάση.[43]

Στην εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας επιλέχθηκε η MongoDB κυρίως λόγω δημοτικότητας και της άψογης δυνατότητας συνδυασμού της με το Node.js. Υπάρχουν πολλές βιβλιοθήκες ανεπτυγμένες σε περιβάλλον Node.js για χρήση με MongoDb με πολλές ισχυρές δυνατότητες. Στην παρούσα εφαρμογή επιλέχθηκε η χρήση της Mongoose.

Σημαντικά πλεονεκτήματα της χρήσης της MongoDB:

- Είναι Document Oriented. Η δυνατότητα της MongoDB ως Μη-Σχεσιακή βάση δεδομένων να αποθηκεύει τα δεδομένα σε έγγραφα και να είναι σε θέση να τα αξιοποιήσει απευθείας μέσω της απλής της σύνταξης σε ζεύγη κλειδιού-τιμής, την καθιστά εξαιρετικά ευέλικτη και προσαρμόσιμη στις ανάγκες του επιχειρηματικού κόσμου.
- Ad hoc queries. Τα ερωτήματα αυτά υποστηρίζουν την αναζήτηση ανά πεδίο, range queries (ερωτήματα εύρους) και αναζητήσεις με regular expressions (κανονικές εκφράσεις).
- Indexing (ευρετηρίαση). Τα indexes (ευρετήρια) είναι οντότητες οι οποίες προσφέρονται στο MongoDB για τη βελτίωση της απόδοσης των αναζητήσεων που πραγματοποιούνται. Κάθε πεδίο μέσα σε ένα έγγραφο μπορεί να ευρετηριαστεί και κατά την αναζήτηση του στη MongoDB να αποδοθεί με πολύ γρήγορο και αποτελεσματικό τρόπο.
- Replication (αντιγραφή). Η βάση δεδομένων MongoDB προσφέρει την δυνατότητα χρησιμοποίησης σετ αντιγράφων. Τα αντίγραφα αυτά διαχωρίζονται σε 2 κατηγορίες, το πρωτεύον και δευτερεύον αντίγραφο. Το πρωτεύον αντίγραφο αντιπροσωπεύει τον διακομιστή που αλληλεπιδρά με την εφαρμογή από την πλευρά του πελάτη (Front-End) για την εκτέλεση των

- λειτουργιών ανάγνωσης και εγγραφής δεδομένων. Τα δευτερεύοντα αντίγραφα χρησιμοποιούνται στο να κρατούν ένα αντίγραφο των δεδομένων του πρωτεύοντος με την αξιοποίηση μίας εργοστασιακής μεθόδου της MongoDB, η οποία ονομάζεται replication. Η λειτουργία του Replication βασίζεται στην εναλλαγή του πρωτεύοντος αντίγραφου όταν αποτύχει με το δευτερεύον. Έτσι το δευτερεύον γίνεται αυτό ο κύριος διακομιστής προσδίδοντας ευελιξία και πολύ υψηλή απόδοση ως προς την λειτουργία του διακομιστή.
- Load Balancing (Εξισορρόπηση φορτίου). Η βάση της MongoDB χρησιμοποιεί την έννοια της θραύσης (sharding) για οριζόντια κλιμάκωση. Με τον τρόπο αυτό τα δεδομένα διαχωρίζονται σε MongoDB instances. Με την δυνατότητα αυτή, η MongoDB μπορεί να εκτελεστεί σε πολλούς διακομιστές ταυτόχρονα, είτε εξισορροπώντας το φορτίο ή αντιγράφοντας δεδομένα ή με την χρήση και των δύο ώστε σε κάποια περίπτωση ζημιάς υλικού, το σύστημα να συνεχίσει να λειτουργεί ορθά.

3.3.4 Postman

3.4: Λογότυπο Postman

Το Postman λειτουργεί ως μία εφαρμογή πελάτη HTTP (client) η οποία διαθέτοντας ένα γραφικό περιβάλλον διεπαφής χρήστη, επιτρέπει την δοκιμή αιτημάτων HTTP. Το Postman παρέχει πολλές μεθόδους αλληλεπίδρασης με τελικά σημεία (endpoints). Αξιοποιήθηκε κατά την υλοποίηση της παρούσας εφαρμογής για την δοκιμή των διαδρομών (routes) που αναπτύχθηκαν μέσω της εφαρμογής του διακομιστή (Back-End). Παρακάτω, παρουσιάζονται μερικά από τα πιο χρησιμοποιούμενα αιτήματα HTTP, συμπεριλαμβανομένων των λειτουργιών τους:[44]

- GET: Δέχεται πληροφορίες
- POST: Δημιουργεί πληροφορίες
- PUT: Αντικαθιστά πληροφορίες
- PATCH: Τροποποιεί πληροφορίες
- DELETE: Διαγράφει πληροφορίες

Ως προς τον έλεγχο των διαδρομών (routes) του API, το Postman παρέχει διαφορετικούς κωδικούς κατάστασης απόκρισης. Παρακάτω περιγράφονται οι πιο συνηθισμένοι κωδικοί απόκρισης:

- 100 Series: Αντιπροσωπεύει τις προσωρινές ή αλλιώς ενημερωτικές απαντήσεις. Υποδηλώνει ότι το αρχικό μέρος μιας αίτησης έχει παραληφθεί.
- 200 Series: Είναι οι επιτυχείς απαντήσεις, δηλαδή αυτές που ο πελάτης στέλνει το αίτημα και ο διακομιστής το λαμβάνει και το αποδίδει πίσω σε αυτόν επιτυχημένα.
- 300 Series: Απαντήσεις που αναφέρονται στην ανακατεύθυνση URL
- 400 Series: Απαντήσεις σφάλματος προς την πλευρά του πελάτη
- 500 Series: Απαντήσεις σφάλματος προς την πλευρά του διακομιστή

3.3.5 Swagger

3.5: Λογότυπο Swagger

Το Swagger είναι πλαίσιο που προσφέρει την παρουσίαση και περιγραφή ενός REST API, από τη σχεδίαση έως τη δοκιμή και την ανάπτυξη του. Αποτελείται από ένα αρχείο σε μορφή JSON το οποίο ονομάζεται swagger.json και διαθέτει τις διευθύνσεις URL των τελικών σημείων (endpoints), τις περιγραφές, τις παραμέτρους και τις δομές απόκρισης για ολόκληρο το API. Μέσω του Swagger οι χρήστες μπορούν να χρησιμοποιήσουν τις υπηρεσίες ιστού RESTful που αναπαράγονται από το API και να δοκιμάσουν τον τρόπο λειτουργίας και τις δυνατότητές του. Το Swagger έχει αποκτήσει πολύ μεγάλη φήμη τα τελευταία χρόνια, διατίθεται από πού μεγάλες εταιρείες όπως η Google, Microsoft, Netflix και πολλές άλλες.[45], [46]

3.4 Επίλογος

Στο παραπάνω κεφάλαιο αναλύθηκαν οι τεχνολογίες που εφαρμόστηκαν για την υλοποίηση της παρούσας εφαρμογής. Η ανάλυση πραγματοποιήθηκε σε τρία στάδια. Αρχικά, αναλύθηκαν οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση του προγραμματισμού από την πλευρά του πελάτη (Front-End). Έπειτα, αναλύθηκαν οι τεχνολογίες που χρησιμοποιήθηκαν από την πλευρά του διακομιστή (Back-End) και τέλος αναλύθηκε η βάση δεδομένων που αξιοποιήθηκε για την αποθήκευση και συντήρηση των δεδομένων.

Κεφάλαιο 4ο: Αρχιτεκτονική Υλοποίησης της Εφαρμογής

4.1 Εισαγωγή

Η παρούσα διπλωματική εργασία είναι μία διαδικτυακή εφαρμογή η οποία ακολουθεί τα πρότυπα της στοίβας τεχνολογιών MERN. Όπως έχει ήδη προαναφερθεί, η στοίβα MERN αποτελείται από τις τεχνολογίες MongoDB, Express.js, Node.js και React. Ως προς την υλοποίηση της εφαρμογής, η MongoDB ως μία Μη-Σχεσιακή βάση δεδομένων εφαρμόστηκε για την αποθήκευση, συντήρηση και οργάνωση των δεδομένων. Η Express.js ως ένα πλαίσιο της πλατφόρμας Node.js χρησιμοποιήθηκε για την ανάπτυξη του REST API λειτουργώντας ως ενδιάμεσο λογισμικό δρομολόγησης για την εφαρμογή διακομιστή (Back-End) και η React ως βιβλιοθήκη ανοιχτού κώδικα JavaScript από την πλευρά του πελάτη (Front-End) χρησιμοποιήθηκε για την ανάπτυξη της διεπαφής χρήστη.

Το κεφάλαιο αυτό αναφέρεται στην ανάλυση της μεθοδολογίας που εφαρμόστηκε για την υλοποίηση της παρούσας διπλωματικής εργασίας. Λόγω της κατασκευής της ακολουθώντας τη δομή της στοίβας MERN, η ανάλυση παρουσιάζεται σε τρία τμήματα, στη βάση δεδομένων, το Back-End REST API και το Front-End, τα οποία αλληλεπιδρώντας το ένα με το άλλο αποτελούν την ολοκληρωμένη διαδικτυακή εφαρμογή. Στην αρχή παρουσιάζονται τα APIs τα οποία χρησιμοποιεί η παρούσα εφαρμογή για την δημιουργία και απόδοση του περιεχομένου της. Έπειτα, αναλύεται η δομή με την οποία δημιουργούνται τα δεδομένα της βάσης δεδομένων MongoDB. Τέλος, επεξηγείται ο τρόπος δημιουργίας και απόδοσης των δεδομένων μέσω του Back-End Rest API και ο τρόπος αξιοποίησης των δεδομένων από το Front-End.[47], [48]

4.2 APIs

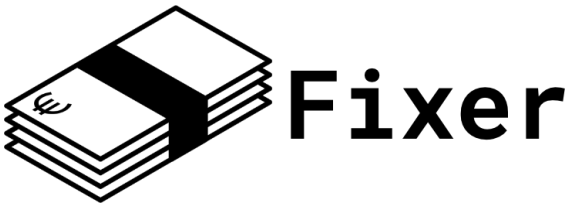
4.2.1 CoinGecko API

4.1: Λογότυπο CoinGecko Api

Το CoinGecko είναι μια διαδικτυακή εφαρμογή η οποία προσφέρει στους χρήστες μια θεμελιώδη ανάλυση ως προς την παρακολούθηση και αγορά των κρυπτονομισμάτων. Εκτός από την παρακολούθηση της τιμής και άλλων χρήσιμων στοιχείων για την αγορά κρυπτονομισμάτων, το CoinGecko διαθέτει μία ολόκληρη κοινότητα, μέσω της οποίας παρέχει στους χρήστες της χρήσιμα στοιχεία όπως ποια κρυπτονομίσματα αποτελούν τάση. Επιπλέον, μέσω της κοινότητας αυτής, το CoinGecko διαθέτει ένα API ανοιχτού κώδικα, το οποίο είναι οργανωμένο, χρησιμοποιείται εύκολα και ενημερώνεται συνεχώς για τις εξελίξεις ως προς την αγορά κρυπτονομισμάτων.

Στην παρούσα διπλωματική εργασία χρησιμοποιήθηκε η δωρεάν έκδοση του CoinGecko API η οποία επιτρέπει έως και 50 αιτήματα ανά λεπτό. Η επιλογή του συγκεκριμένου API έγινε λόγω της δωρεάν χρησιμοποίησης του, της πολύ καλής οργάνωσης και ευκολίας χρήσης των δεδομένων του.[49]

4.2.2 Fixer API



4.2: Λογότυπο Fixer Api

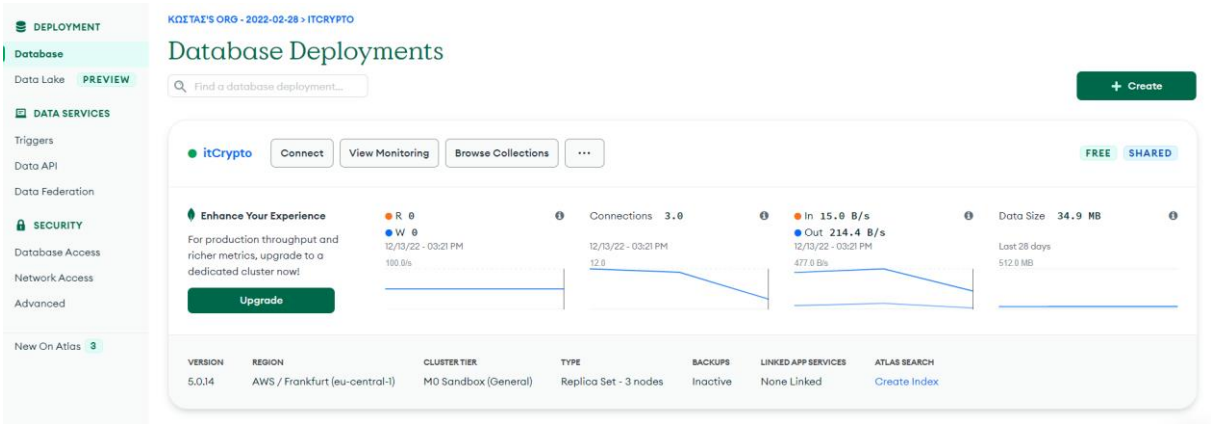
Το Fixer API χρησιμοποιείται για την παροχή δεδομένων συναλλαγματικής ισοτιμίας για πάνω από 170 παγκόσμια νομίσματα. Τα δεδομένα αυτά αποσπώνται από δεκαπέντε αξιόπιστες πηγές όπως τράπεζες και η ενημέρωση τους γίνεται ανά ένα λεπτό. Το Fixer API διαθέτει πολλών ειδών τελικά σημεία για την παροχή δεδομένων όπως η παρακολούθηση των συναλλαγματικών ισοτιμιών. Στην παρούσα διαδικτυακή εφαρμογή, το Fixer API αξιοποιήθηκε για την μετατροπή των τιμών των κρυπτονομισμάτων από ευρώ ως το προκαθορισμένο νόμισμα σε άλλα νομίσματα όπως το δολάριο, η λίρα κ.α. Πιο συγκεκριμένα, το Fixer API παρέχει στην εφαρμογή τις συναλλαγματικές ισοτιμίες με βάση το νόμισμα ευρώ για κάποια διαφορετικά παγκόσμια νομίσματα. Η συναλλαγματικές ισοτιμίες πολλαπλασιάζονται με την τιμή του ευρώ και έτσι παράγονται οι πραγματικές τιμές των νομισμάτων.[50]

4.3 Βάση Δεδομένων - MongoDB

Ένα από τα σημαντικότερα δομικά στοιχεία για την δημιουργία μίας διαδικτυακής εφαρμογής είναι η αξιοποίηση μίας βάσης δεδομένων. Μία σωστά δομημένη βάση δεδομένων καθορίζει σε πολύ υψηλό βαθμό την λειτουργία, την απόδοση και την επεκτασιμότητα της παραγόμενης εφαρμογής. Στη παρούσα εφαρμογή χρησιμοποιήθηκε η MongoDB και στο κεφάλαιο αυτό θα αναλυθούν ο τρόπος δημιουργίας της μέσω της διαδικτυακής υπηρεσίας MongoDB Atlas και τα μοντέλα και σχήματα (Models and Schemas) τα οποία δημιουργήθηκαν μέσω του πακέτου Mongoose. Για την υλοποίηση της εφαρμογής αναπτύχθηκαν τριών ειδών μοντέλα και σχήματα διαχωρισμένα σε αυτά των χρηστών, των παραστατικών νομισμάτων και των κρυπτονομισμάτων στα οποία συμπεριλαμβάνονται τα μοντέλα και σχήματα των ανταλλακτηρίων και μοντέρνων (trending) κρυπτονομισμάτων.

4.3.1 MongoDb Atlas

Για την δημιουργία της βάσης δεδομένων MongoDB χρησιμοποιήθηκε η δωρεάν έκδοση της διαδικτυακής υπηρεσίας MongoDB Atlas. Η MongoDB Atlas υλοποιήθηκε από την ομάδα της MongoDB και ορίζεται ως μια βάση δεδομένων που επιτρέπει την διαχείριση των δεδομένων της διαδικτυακά μέσω υπηρεσιών cloud της επιλογής (AWS, Azure και GCP). Στην παρούσα διαδικτυακή εφαρμογή, μέσω της υπηρεσίας MongoDB Atlas πραγματοποιήθηκε η δημιουργία, η σύνδεση με την εφαρμογή διακομιστή και ο χειρισμός της ανάπτυξης της βάσης δεδομένων στον πάροχο υπηρεσιών cloud, AWS.[51]



4.3: Σχήμα MongoDB Atlas

4.3.2 Models and Schemas (Μοντέλα - Σχήματα)

4.3.2.1 User Model

Το σχήμα των χρηστών, δηλαδή η δομή μέσω της οποίας δημιουργούνται τα δεδομένα στην βάση δεδομένων MongoDB αποτελείται από δύο πεδία, το "email" και το "password" (κωδικό). Και τα δύο αυτά πεδία είναι αλφαριθμητικού τύπου και είναι απαιτούμενα για την δημιουργία ενός χρήστη. Επιπλέον, το πεδίο "email" διαθέτει το στοιχείο μοναδικό (unique) το οποίο υποδηλώνει ότι το περιεχόμενο του πεδίου "email" είναι μοναδικό και δεν υπάρχει άλλο έγγραφο αποθηκευμένο στη βάση δεδομένων με το ίδιο περιεχόμενο.

```
const userSchema = new Schema({
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  }
});
module.exports = mongoose.model('User', userSchema);
```

4.3.2.2 Currencies Model

Το συγκεκριμένο μοντέλο διαθέτει τις συναλλαγματικές ισοτιμίες για την μετατροπή του νομίσματος του ευρώ σε οποιοδήποτε άλλο παραστατικό νόμισμα. Τα δεδομένα αυτά παρέχονται μέσω του Fixer API το οποίο έχει αναλυθεί παραπάνω. Το σχήμα των παραστατικών νομισμάτων αποτελείται από τα πεδία "_id: false" και "any: {}". Το πρώτο πεδίο, "_id: false", εφαρμόστηκε ώστε το πακέτο mongoose να μην καταχωρεί από προεπιλογή ένα πεδίο "_id" στο σχήμα. Επιπλέον, αν το σχήμα περιέχει το πεδίο "id" για την εγγραφή μίας μοναδικής τιμής, όταν επιδιώκεται αναζήτηση για ένα συγκεκριμένο έγγραφο, το σύστημα θεωρεί όμοια τα πεδία "id" και "_id" αλλά αναπτύσσει την αναζήτηση μόνο για

το πεδίο "_id", δημιουργώντας σφάλμα. Το δεύτερο πεδίο, "any: {}", χρησιμοποιήθηκε ώστε να επιτρέπεται στη βάση δεδομένων να εισάγονται απευθείας οι τιμές του Fixer API χωρίς να ορίζονται από κάποια συγκεκριμένη δομή. Το πεδίο αυτό ορίζεται ως "schema-less" δηλαδή χωρίς κάποια δομή σχήματος. Το πακέτο mongoose διαθέτει από προεπιλογή την λειτουργία strict (αυστηρή επιλογή) η οποία διασφαλίζει ότι οι τιμές που δεν είναι καθορισμένες βάσει της δομής του σχήματος, δεν αποθηκεύονται στο db. Για τον λόγο αυτό, χρησιμοποιείται η επιλογή { strict: false } ώστε τα δεδομένα τα οποία δημιουργούνται μέσω του πεδίου "any: {}" να αποθηκεύονται στην βάση δεδομένων.

```
const geckoSchema = new Schema(
  {
    _id: false,
    any: {},
  },
  { strict: false }
);
module.exports = mongoose.model("currencies", geckoSchema);
```

4.3.2.3 Crypto Model

Το μοντέλο των κρυπτονομισμάτων διαθέτει την δομή όλων των πληροφοριών που παρέχει η διαδικτυακή εφαρμογή για τα κρυπτονομίσματα. Τα δεδομένα αυτά παρέχονται μέσω του CoinGecko API το οποίο έχει αναλυθεί παραπάνω. Το σχήμα αυτό αποτελείται από τα πεδία "_id: false" και "any: {}" τα οποία αναλύθηκαν παραπάνω, στο μοντέλο των παραστατικών νομισμάτων, με την μόνη διαφορά ότι το πεδίο "any: {}" χρησιμοποιείται για την εισαγωγή των δεδομένων μέσω του CoinGecko API. Επιπρόσθετα, το σχήμα αποτελείται και από τα πεδία "id" και "user_ids". Το πεδίο "id" αντιπροσωπεύει το "id" του κρυπτονομίσματος το οποίο παρέχεται μέσω του CoinGecko API και χρησιμοποιείται για την επίτευξη μιας πιο αποδοτικής αναζήτησης σε χρόνο για το συγκεκριμένο κρυπτονόμισμα. Το πεδίο "user_ids" είναι ένας πίνακας ο οποίος διαθέτει τα "id" των χρηστών που έχουν αποθηκεύσει στην λίστα παρακολούθησης (watchlist) τους, το συγκεκριμένο κρυπτονόμισμα.

```
const geckoSchema = new Schema(
  {
    _id: false,
    id: {
      type: String,
      required: true,
    },
    user_ids: [
      {
        type: String,
      },
    ],
    any: {},
  },
  { strict: false }
);
module.exports = mongoose.model("Coins", geckoSchema);
```

4.3.2.4 Exchanges Model

Το συγκεκριμένο μοντέλο διαθέτει τις πληροφορίες για τα ανταλλακτήρια τα οποία επιτρέπουν την αγοραπωλησία κρυπτονομισμάτων. Τα δεδομένα αυτά παρέχονται μέσω του CoinGecko API το οποίο έχει αναλυθεί παραπάνω. Το σχήμα των παραστατικών νομισμάτων αποτελείται από τα πεδία "_id: false" και "any: {}" τα οποία έχουν αναλυθεί παραπάνω, στο μοντέλο των παραστατικών νομισμάτων, με την μόνη διαφορά ότι το πεδίο "any: {}" χρησιμοποιείται για την εισαγωγή των δεδομένων μέσω του CoinGecko API.

```
const geckoSchema = new Schema(
  {
    _id: false,
    any: {},
  },
  { strict: false }
);
module.exports = mongoose.model("exchanges", geckoSchema);
```

4.3.2.5 TrendingCoins Model

Το συγκεκριμένο μοντέλο διαθέτει τις πληροφορίες για τα κρυπτονομίσματα τα οποία χρησιμοποιούνται περισσότερο σε μια συγκεκριμένη χρονική περίοδο. Τα δεδομένα αυτά παρέχονται μέσω του CoinGecko API το οποίο έχει αναλυθεί παραπάνω. Το σχήμα των παραστατικών νομισμάτων αποτελείται από τα πεδία "_id: false" και "any: {}" τα οποία έχουν αναλυθεί παραπάνω, στο μοντέλο των παραστατικών νομισμάτων, με την μόνη διαφορά ότι το πεδίο "any: {}" χρησιμοποιείται για την

εισαγωγή των δεδομένων μέσω του CoinGecko API. Το πεδίο “name” αντιπροσωπεύει το όνομα του κρυπτονομίσματος και χρησιμοποιείται για την διευκόλυνση της παραπομπής του σε συγκεκριμένες μεθόδους οι οποίες θα αναλυθούν παρακάτω, στο σχεδιασμό του Back-End API.

```
const geckoSchema = new Schema(
  {
    _id: false,
    name: {
      type: String,
      required: true,
    },
    any: {},
  },
  { strict: false }
);
module.exports = mongoose.model("trendingCoins", geckoSchema);
```

4.4 Back-End Rest API

Το Back-End είναι ένα απαραίτητο δομικό στοιχείο για την ανάπτυξη μίας διαδικτυακής εφαρμογής. Ουσιαστικά, χρησιμοποιείται στο να λαμβάνει τα αιτήματα που δημιουργούνται από την πλευρά του πελάτη της εφαρμογής και να διαθέτει τη λογική και την ικανότητα ώστε να στείλει τα κατάλληλα δεδομένα ως απάντηση πίσω στην πλευρά του πελάτη. Επιπλέον, περιλαμβάνει τη σύνδεση και διαχείριση της βάσης δεδομένων που χρησιμοποιείται.

Στην παρούσα διπλωματική εργασία, το Back-End είναι ανεξάρτητο από το Front-End και λειτουργεί ως ένα Restful API. Εκτελείται μέσω ενός αρχείου JavaScript το οποίο ονομάζεται server.js και περιέχει όλες τις λειτουργίες υλοποίησης της εφαρμογής διακομιστή. Για την υλοποίηση του χρησιμοποιήθηκε το πλαίσιο Express.js της πλατφόρμας ανάπτυξης λογισμικού ανοιχτού κώδικα Node.js και επιτεύχθηκαν οι παρακάτω λειτουργίες:

1) Σύνδεση με την βάση δεδομένων MongoDB και διαχείριση του τρόπου εισαγωγής των δεδομένων της.
2) Εισαγωγή δεδομένων στη βάση από τα APIs, CoinGecko και Fixer, με την δημιουργία σεναρίων.
3) Ανάπτυξη λειτουργιών για την επαλήθευση χρηστών (user Authentication).
4) Δημιουργία Διαδρομών συστήματος (routes).
5) Δημιουργία διεπαφής Swagger για την αξιοποίηση του Back-End REST API.

Η ενότητα αυτή αναφέρεται στην ανάλυση της μεθοδολογίας που εφαρμόστηκε για την υλοποίηση του προγραμματισμού Back-End επεξηγώντας τον τρόπο υλοποίησης των παραπάνω λειτουργιών.

4.4.1 Σύνδεση με την βάση δεδομένων MongoDB

Για την υλοποίηση της σύνδεσης με την βάση δεδομένων MongoDB και πιο συγκεκριμένα με την διαδικτυακή υπηρεσία MongoDB Atlas εφαρμόστηκε το πλαίσιο Mongoose. Όπως έχει ήδη προαναφερθεί, το Mongoose είναι ένα πλαίσιο το οποίο τυλίγει την οντότητα MongoDB με ένα επιπλέον στρώμα το οποίο διαθέτει διάφορους μεθόδους προς αξιοποίηση. Μία από αυτές τις μεθόδους είναι το connect. Η μέθοδος αυτή λειτουργεί ασύγχρονα, απαιτεί ως παράμετρο ένα σύνολο αλφαριθμητικών χαρακτήρων, το οποίο παρέχεται με την δημιουργία της βάσης δεδομένων μέσω του MongoDB Atlas και πρέπει να είναι κρυφό ώστε να μην μπορούν άλλοι χρήστες να εισχωρήσουν και να επεξεργαστούν την βάση δεδομένων. Αν το σύνολο αλφαριθμητικών χαρακτήρων είναι σωστό, τότε διαβάζει το port και η εφαρμογή εκτελείται, αν όχι, εμφανίζει μήνυμα λάθους. Το σύνολο αλφαριθμητικών χαρακτήρων “MONGO_URI” και το “PORT” παρέχονται μέσω ενός αρχείου env για την διατήρηση των ευαίσθητων αυτών δεδομένων χωριστά από τον κώδικα. Παρακάτω συντάσσεται ο κώδικας ο οποίος εφαρμόστηκε για την υλοποίηση της σύνδεσης της εφαρμογής του διακομιστή με την βάση δεδομένων.

```
const mongoose = require("mongoose");
// connect to db
mongoose
  .connect(process.env.MONGO_URI)
  .then(() => {
    // listen to port
    app.listen(process.env.PORT, () => {
      console.log(
        "connected to db & listening for requests on port",
        process.env.PORT
      );
    });
  })
  .catch((err) => {
    console.log(err);
  });
```

4.4.2 Εισαγωγή δεδομένων αξιοποιώντας APIs

Η εισαγωγή των δεδομένων στην βάση δεδομένων MongoDB πραγματοποιήθηκε με την δημιουργία σεναρίων (scripts) τα οποία προγραμματίστηκαν στο να αποσπούν δεδομένα από τα APIs, CoinGecko και Fixer, ανά συγκεκριμένα χρονικά διαστήματα. Για την υλοποίηση της εφαρμογής διακομιστή της παρούσας διπλωματικής εργασίας εφαρμόστηκαν τα παρακάτω τέσσερα σενάρια:

4.4.2.1 Εισαγωγή των κρυπτονομισμάτων

Για την εισαγωγή των δεδομένων των κρυπτονομισμάτων χρησιμοποιήθηκε ένα τελικό σημείο (endpoint) του CoinGecko API σε μορφή URL, το https://api.coingecko.com/api/v3/coins/markets. Για την πλήρη αξιοποίηση του, χρησιμοποιήθηκαν διάφοροι παράμετροι ερωτήματος με τις πιο σημαντικές

αυτές της σελίδας εμφάνισης στοιχείων η οποία είναι απαραίτητη και του αριθμού εμφάνισης στοιχείων ανά σελίδα. Η ταξινόμηση των κρυπτονομισμάτων γίνεται ανεβαίνοντας στις σελίδες από τα κρυπτονομίσματα μεγαλύτερης τάξης σε αυτά της μικρότερης, δηλαδή οι πρώτες σελίδες περιέχουν τα κρυπτονομίσματα της μεγαλύτερης τάξης. Κατά την εκτέλεση του σεναρίου, η σελίδα εμφάνισης στοιχείων αλλάζει δυναμικά και ο αριθμός εμφάνισης στοιχείων είναι ο μεγαλύτερος δυνατός βάσει του CoinGecko API στα 250 στοιχεία ανά σελίδα.

Η αρχιτεκτονική η οποία χρησιμοποιήθηκε για την εισαγωγή των δεδομένων των κρυπτονομισμάτων στηρίζεται στην εισαγωγή και τροποποίηση των στοιχείων της μεγαλύτερης τάξης γρηγορότερα από αυτά των χαμηλότερων τάξεων ώστε να μην ξεπεραστεί ο αριθμός των 50 αιτημάτων ανά λεπτό που διαθέτει ως περιορισμό η δωρεάν έκδοση του CoinGecko API. Για τον λόγο αυτό, υλοποιήθηκαν τρεις μέθοδοι με τα ονόματα “fetch1” η οποία φορτώνει τα δεδομένα των πρώτων 10 σελίδων ανά 15 δευτερόλεπτα την κάθε μία, “fetch2” η οποία φορτώνει δεδομένα των επόμενων 20 σελίδων ανά 30 δευτερόλεπτα την κάθε μία και “fetch3” η οποία φορτώνει τα δεδομένα για τις υπόλοιπες σελίδες ανά 35 δευτερόλεπτα την κάθε μία. Οι μέθοδοι αυτοί εκτελούνται συνεχώς και ταυτόχρονα. Σε κάθε φόρτωση των δεδομένων ανά σελίδα χρησιμοποιείται η μέθοδος “updateMany” του πλαισίου mongoose για την εισαγωγή των δεδομένων στο μοντέλο “coins” το οποίο αναπτύχθηκε ώστε να περιέχει όλες τις πληροφορίες για τα κρυπτονομίσματα. Η μέθοδος “updateMany” σε κάθε φόρτωση χρησιμοποιείται για να αλλάζει τις παλαιότερες τιμές δεδομένων των κρυπτονομισμάτων με τις νεότερες απευθείας.

4.4.2.2 Εισαγωγή των πιο διαδεδομένων κρυπτονομισμάτων

Για την εισαγωγή των δεδομένων των πιο διαδεδομένων κρυπτονομισμάτων ανά μέρα χρησιμοποιήθηκε επίσης ένα τελικό σημείο (endpoint) του CoinGecko API, το https://api.coingecko.com/api/v3/search/trending. Το συγκεκριμένο URL δεν διαθέτει παράμετρους ερωτήματος. Κατά την εκτέλεση του, επιστρέφει τα δεδομένα 7 κρυπτονομισμάτων που έχουν αναζητηθεί τις περισσότερες φορές από τα μέλη της εφαρμογής του CoinGecko.

Η αρχιτεκτονική του συγκεκριμένου σεναρίου στηρίζεται στην αξιοποίηση του πακέτου Node-Schedule το οποίο επιτρέπει την εκτέλεση μεθόδων σε συγκεκριμένες ημερομηνίες με κατ' επιλογήν κανόνες επανάληψης. Το συγκεκριμένο σενάριο εκτελείται ανά 24 ώρες και 1 λεπτό. Σε κάθε φόρτωση των δεδομένων χρησιμοποιούνται δύο μέθοδοι του πλαισίου Mongoose. Αρχικά, εφαρμόζεται η μέθοδος “deleteMany” για την διαγραφή των υπαρχόντων στοιχείων και έπειτα η μέθοδος “insertMany” για την εισαγωγή των νέων στοιχείων που φορτώθηκαν. Οι μέθοδοι αυτοί εκτελούνται στο μοντέλο “trendingCoins” το οποίο αναπτύχθηκε ώστε να περιέχει τα δεδομένα των πιο διαδεδομένων κρυπτονομισμάτων. Ο λόγος για τον οποίο δεν χρησιμοποιήθηκε η μέθοδος “updateMany” είναι ότι μετά από κάθε νέα φόρτωση στοιχείων, τα προηγούμενα κρυπτονομίσματα τα οποία δηλαδή δεν είναι πια διαδεδομένα, θα παρέμεναν στην βάση αφού περιέχουν διαφορετικό αναγνωριστικό “id” και δεν θα μπορούσαν να τροποποιηθούν.

4.4.2.3 Εισαγωγή των ανταλλακτηρίων των κρυπτονομισμάτων

Για την εισαγωγή των δεδομένων των πιο διαδεδομένων κρυπτονομισμάτων ανά μέρα χρησιμοποιήθηκε επίσης ένα τελικό σημείο (endpoint) του CoinGecko API, το https://api.coingecko.com/api/v3/exchanges. Για την πλήρη αξιοποίηση του, χρησιμοποιήθηκαν δύο παράμετροι ερωτήματος, η σελίδας εμφάνισης στοιχείων και ο αριθμού εμφάνισης στοιχείων ανά σελίδα. Κατά την εκτέλεση του σεναρίου, η σελίδα εμφάνισης στοιχείων αλλάζει δυναμικά και ο αριθμός εμφάνισης στοιχείων είναι ο μεγαλύτερος δυνατός βάσει του CoinGecko API στα 250 στοιχεία ανά σελίδα.

Η αρχιτεκτονική του συγκεκριμένου σεναρίου στηρίζεται στην αξιοποίηση του πακέτου Node-Schedule το οποίο επιτρέπει την εκτέλεση μεθόδων σε συγκεκριμένες ημερομηνίες με κατ' επιλογήν κανόνες επανάληψης. Το συγκεκριμένο σενάριο εκτελείται ανά 12 ώρες και 1 λεπτό. Επιπλέον, κάθε φόρτωση δεδομένων ανά σελίδα εμφάνισης υλοποιείται ανά 600 δευτερόλεπτα (10 λεπτά) λόγω του ότι οι τιμές των ανταλλακτηρίων αλλάζουν πολύ πιο αργά σε σύγκριση με αυτές των κρυπτονομισμάτων και ώστε να υπάρχει σημαντικό περιθώριο ώστε να μην ξεπεραστεί ο αριθμός των 50 αιτημάτων ανά λεπτό που διαθέτει ως περιορισμό η δωρεάν έκδοση του CoinGecko API. Σε κάθε φόρτωση των δεδομένων ανά σελίδα χρησιμοποιείται η μέθοδος “updateMany” του πλαισίου mongoose για την εισαγωγή των δεδομένων στο μοντέλο “exchanges” το οποίο αναπτύχθηκε ώστε να περιέχει όλες τις πληροφορίες για τα ανταλλακτήρια κρυπτονομισμάτων. Η μέθοδος “updateMany” σε κάθε φόρτωση χρησιμοποιείται για να αλλάζει τις παλαιότερες τιμές δεδομένων των ανταλλακτηρίων των κρυπτονομισμάτων με τις νεότερες απευθείας.

4.4.2.4 Εισαγωγή των δεδομένων συναλλαγματικής ισοτιμίας

Τα παραπάνω σενάρια τα οποία δημιουργήθηκαν αξιοποιώντας δεδομένα μέσω του CoinGecko API επιστρέφουν τα δεδομένα σε μία μόνο συναλλαγματική ισοτιμία, αυτή του ευρώ (€). Για την εισαγωγή των δεδομένων σε περισσότερες συναλλαγματικές ισοτιμίες θα έπρεπε να δημιουργηθούν πολλές περισσότερες φορτώσεις δεδομένων από το CoinGecko API και το όριο των 50 αιτημάτων ανά λεπτό που διαθέτει η δωρεάν έκδοσή του θα είχε ξεπεραστεί.

Η επίλυση του προβλήματος αυτού επιτεύχθηκε με την εισαγωγή των δεδομένων συναλλαγματικής ισοτιμίας μέσω ενός τελικό σημείο (endpoint) της διεπαφής προγραμματισμού Fixer API, το https://api.apilayer.com/fixer/latest?base=EUR. Το συγκεκριμένο URL, για την αξιοποίηση του, απαιτεί την δημιουργία ενός κλειδιού γνωστό ως “API key”, το οποίο εμπεριέχεται στο URL ως παράμετρος ερωτήματος. Το κλειδί αυτό προσφέρεται δωρεάν μέσω της διαδικτυακής υπηρεσίας του Fixer.

Η αρχιτεκτονική του συγκεκριμένου σεναρίου στηρίζεται στην αξιοποίηση του πακέτου Node-Schedule το οποίο επιτρέπει την εκτέλεση μεθόδων σε συγκεκριμένες ημερομηνίες με κατ' επιλογήν κανόνες επανάληψης. Το συγκεκριμένο σενάριο εκτελείται ανά 2 ώρες και 1 λεπτό. Σε κάθε φόρτωση των δεδομένων χρησιμοποιούνται δύο μέθοδοι του πλαισίου Mongoose. Αρχικά, εφαρμόζεται η μέθοδος “updateMany” η οποία τροποποιεί την τιμή των υπαρχόντων στοιχείων. Αν φορτώσει κάποιο δεδομένο το οποίο δεν υπάρχει στην βάση, η μέθοδος “updateMany” το δημιουργεί. Ωστόσο, τα δεδομένα που φορτώνονται αποτελούν τις συναλλαγματικές αξίες 170 νομισμάτων τα οποία είναι πολλά και θα καθιστούσαν την εφαρμογή ιδιαίτερα αργή. Για αυτό τον λόγο εφαρμόστηκε η δεύτερη μέθοδος “find” μέσω της οποίας μεταφέρονται στην βάση δεδομένων μόνο οι συναλλαγματικές αξίες των παρακάτω νομισμάτων.

USD: Δολάριο των Ηνωμένων Πολιτειών

JPY: Γιεν Ιαπωνίας

GBP: Λίρα Αγγλίας

CAD: Δολάριο Καναδά

AUD: Δολάριο Αυστραλίας

4.4.3 Επαλήθευση χρηστών (user Authentication)

4.4.3.1 Εισαγωγή

Στην ενότητα αυτή παρουσιάζεται η αρχιτεκτονική η οποία χρησιμοποιήθηκε για την εγγραφή και σύνδεση των χρηστών στην εφαρμογή. Επίσης, παρουσιάζεται η λειτουργία requireAuth η οποία είναι υπεύθυνη για την προστασία των διαδρομών της εφαρμογής από μη επαληθευμένους χρήστες. Η εκτέλεση των διαδρομών της επαλήθευσης χρηστών γίνεται στο αρχείο server.js με την μορφή:

app.use("/api/user", userRoutes);

4.4.3.2 Εγγραφή χρήστη

Η εγγραφή ενός χρήστη στο σύστημα της εφαρμογής επιτυγχάνεται με την δημιουργία της παρακάτω διαδρομής (endpoint):

/api/user/signup

Η οποία συντάσσεται προγραμματιστικά στο αρχείο διαδρομών userRoutes ως:

router.post('/signup, signupUser);

Ουσιαστικά, η διαδρομή αυτή δημιουργείται από το δρομολογητή (Router) που προσφέρει η Express.js και χρησιμοποιείται για την διαχείριση του αιτήματος της εγγραφής ενός χρήστη μέσω της συνάρτησης "signupUser", η οποία καλείται όταν λαμβάνεται το αίτημα αυτό και αλληλεπιδρά με το μοντέλο "User" (UserModel) για την αποθήκευση ή απόρριψη του αιτήματος αυτού. Παρακάτω αναλύεται ο τρόπος αλληλεπίδρασης της συνάρτησης "signupUser" με το μοντέλο "UserModel" για την για την διαχείριση του αιτήματος της εγγραφής χρήστη με την βοήθεια τμημάτων κώδικα της εφαρμογής.

UserModel - signup function

Το πλαίσιο Mongoose προσφέρει διάφορες μεθόδους για την διαχείριση των δεδομένων που δομούνται σε ένα μοντέλο. Ωστόσο, προσφέρει και την δυνατότητα δημιουργίας μεθόδων για την υλοποίηση συγκεκριμένων ενεργειών με μεγαλύτερη ταχύτητα αφού εκτελούνται απευθείας μέσα στο μοντέλο. Στο "UserModel" γίνεται χρήση στατικής μεθόδου που προσφέρει το πλαίσιο Mongoose για τον καθορισμό της συνάρτησης signup. Η συνάρτηση αυτή δέχεται ως παραμέτρους το μέιλ (email) και τον κωδικό (password). Αρχικά, ελέγχει αν το μέιλ που δόθηκε είναι έγκυρο μέσω του πακέτου Validator. Έπειτα, μέσω της μεθόδου "findOne" βρίσκει αν το μέιλ υπάρχει ήδη στη βάση δεδομένων ώστε να μην μπορεί να ξανά χρησιμοποιηθεί. Τέλος, εφόσον το μέιλ δεν υπάρχει, μέσω του πακέτου Bcrypt κάνει κατακερματισμό (hashing) των κωδικό που έχει δώσει ο χρήστης και αποθηκεύει στη βάση δεδομένων το μέιλ του και τον κωδικό του ο οποίος έχει κατακερματιστεί. Παρακάτω παρουσιάζεται ο κώδικας που εφαρμόστηκε για την υλοποίηση της στατικής συνάρτησης "signup".

```
userSchema.statics.signup = async function(email, password) {
  // validation
  if (!email || !password) { //checks if i have a value for email or password or both
    throw Error('All fields must be filled')
  }
  if (!validator.isEmail(email)) {
    throw Error('Email not valid')
  }
  if (!validator.isStrongPassword(password)) {
    throw Error('Password not strong enough')
  }
  const exists = await this.findOne({ email }) //this equals to user
  if (exists) {
    throw Error('Email already in use')
  }
  const salt = await bcrypt.genSalt(10) //adds 10 chars to the String-Password before it
gets hashed (extra protection)
  const hash = await bcrypt.hash(password, salt)
  const user = await this.create({ email, password: hash })
  return user
}
```

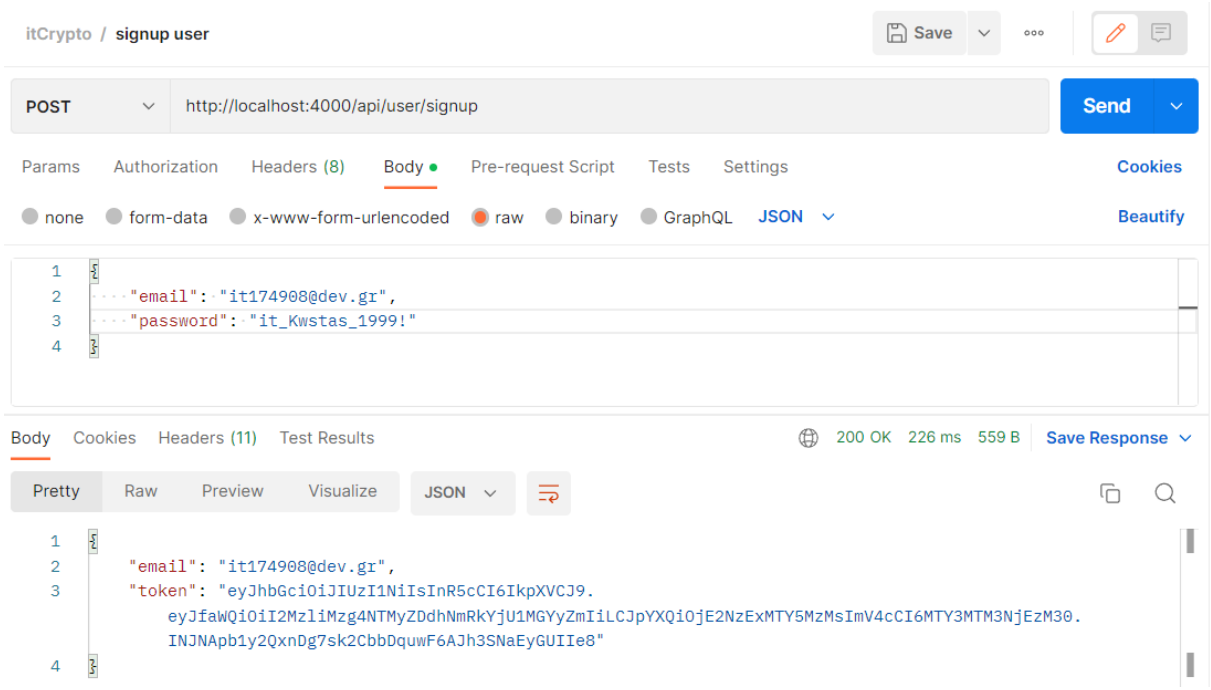
signupUser

Η συνάρτηση “signupUser” είναι μια ασύγχρονη λειτουργία η οποία καλείται όταν λαμβάνεται το αίτημα της εγγραφής ενός χρήστη από το σύστημα και χρησιμοποιείται για τον τρόπο διαχείρισης του αιτήματος αυτού. Περιέχει ως παραμέτρους το “req - request” (αίτημα) με τα αντικείμενα τα οποία εισέρχεται στο σύστημα και το “res - response” (απάντηση) που θα στείλει το σύστημα στο αίτημα αυτό. Ως προς την λειτουργία της συνάρτησης αυτής, αρχικά αποσπώνται τα αντικείμενα μέιλ και κωδικός από το αίτημα που αποστάλθηκε και στη συνέχεια εκτελείται η στατική συνάρτηση signup που δημιουργήθηκε στο “UserModel”. Τέλος, χρησιμοποιείται το πακέτο JWT (JsonWebToken) για την δημιουργία μίας ένδειξης (TOKEN) η οποία θα ισχύει για 3 ημέρες. Η δημιουργία της ένδειξης γίνεται μέσω της μεθόδου sign με παραμέτρους το αναγνωριστικό “_id” του χρήστη και ενός κρυφού συνόλου χαρακτήρων (SECRET) το οποίο γνωρίζει μόνο η εφαρμογή διακομιστή. Η ένδειξη επιστρέφει ως απάντηση του συστήματος μαζί με το μέιλ του χρήστη μέσω του αντικειμένου “response” που εμπεριέχεται ως παράμετρος δημιουργώντας έτσι έναν νέο επαληθευμένο χρήστη. Αν λήξει η προθεσμία των τριών ημερών, ο χρήστης θα πρέπει να ξανά συνδεθεί στο σύστημα. Παρακάτω παρουσιάζεται ο κώδικας που εφαρμόστηκε για την υλοποίηση της συνάρτησης “signupUser”.

```
const User = require('../models/user/userModel')
const jwt = require('jsonwebtoken')

const createToken = (_id) => {
  return jwt.sign({_id}, process.env.SECRET, { expiresIn: '3d' })
}
// signup a user
const signupUser = async (req, res) => {
  const {email, password} = req.body
  try {
    const user = await User.signup(email, password)
    // create a token
    const token = createToken(user._id)
    res.status(200).json({email, token})
  } catch (error) {
    res.status(400).json({error: error.message})
  }
}
```

Παράδειγμα εγγραφής ενός χρήστη στο σύστημα.



4.4.3.3 Σύνδεση χρήστη

Η σύνδεση ενός επαληθευμένου χρήστη στο σύστημα της εφαρμογής επιτυγχάνεται με την δημιουργία της παρακάτω διαδρομής (endpoint):

/api/user/login

Η οποία συντάσσεται προγραμματιστικά στο αρχείο διαδρομών userRoutes ως:

router.post('/login, loginUser);

Η διαδρομή αυτή χρησιμοποιείται για την διαχείριση του αιτήματος της σύνδεσης ενός επαληθευμένου χρήστη μέσω της συνάρτησης "loginUser", η οποία καλείται όταν λαμβάνεται το αίτημα αυτό και αλληλεπιδρά με το μοντέλο "UserModel" για την αποθήκευση ή απόρριψη του αιτήματος αυτού. Παρακάτω αναλύεται ο τρόπος αλληλεπίδρασης της συνάρτησης "loginUser" με το μοντέλο "UserModel" για την για την διαχείριση του αιτήματος της σύνδεσης χρήστη με την βοήθεια τμημάτων κώδικα της εφαρμογής.

UserModel - login function

Στο "UserModel" γίνεται επίσης χρήση στατικής μεθόδου που προσφέρει το πλαίσιο Mongoose για τον καθορισμό της συνάρτησης "login". Η συνάρτηση αυτή δέχεται ως παραμέτρους το μέιλ (email) και τον κωδικό (password). Αρχικά, επιβεβαιώνεται ότι οι παράμετροι μέιλ και κωδικός διαθέτουν τιμή. Τότε, μέσω της μεθόδου "findOne", ελέγχεται αν υπάρχει η τιμή του μέιλ αποθηκευμένη στην βάση δεδομένων. Έπειτα, εφόσον υπάρχει η τιμή του μέιλ, χρησιμοποιείται η μέθοδος "compare" του πακέτου Bcrypt για να συγκρίνει την ισότητα του κωδικού που πληκτρολόγησε ο χρήστης και του κωδικού που βρίσκεται αποθηκευμένος στη βάση δεδομένων και είναι κατακερματισμένος. Τέλος, αν το αποτέλεσμα της σύγκρισης είναι αληθής, το σύστημα επιστρέφει τον χρήστη. Σε κάθε περίπτωση από τις παραπάνω, αν κάποιος έλεγχος δώσει αναληθή αποτέλεσμα, το σύστημα επιστρέφει μήνυμα λάθους. Παρακάτω παρουσιάζεται ο κώδικας της στατικής συνάρτησης "login".

```
// static login method
userSchema.statics.login = async function(email, password) {
  if (!email || !password) {
    throw Error('All fields must be filled')
  }
  const user = await this.findOne({ email })
  if (!user) {
    throw Error('Incorrect email')
  }
  const match = await bcrypt.compare(password, user.password)
  // compares the passwords, the first is plain text and the other is hashed password
  // bcrypt can compare them and return if the passwords are the same
  if (!match) {
    throw Error('Incorrect password')
  }
  return user
}
```

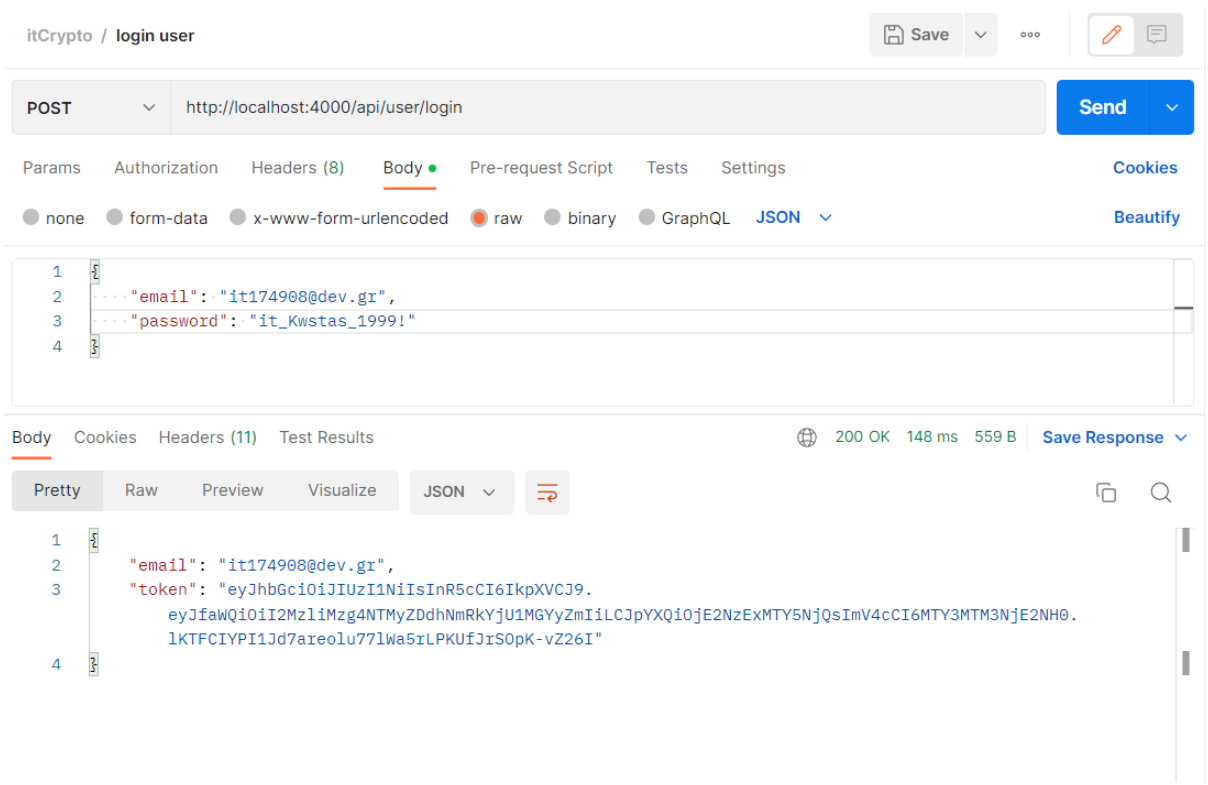
loginUser

Η συνάρτηση "loginUser" ακολουθεί ίδια μεθοδολογία με τη συνάρτηση "signupUser" που αναλύθηκε παραπάνω. Η μόνη διαφορά είναι ότι χρησιμοποιεί την λειτουργία "login" του "UserModel" για την διαχείριση των χρηστών που έχουν σκοπό να συνδεθούν στην εφαρμογή αντί της λειτουργίας "signup"

που υλοποιεί την εγγραφή ενός χρήστη. Παρακάτω παρουσιάζεται ο κώδικας της στατικής συνάρτησης “loginUser”.

```
const User = require('../models/user/userModel')
const jwt = require('jsonwebtoken')
const createToken = (_id) => {
  return jwt.sign({_id}, process.env.SECRET, { expiresIn: '3d' })
}
// login a user
const loginUser = async (req, res) => {
  const {email, password} = req.body
  try {
    const user = await User.login(email, password)
    // create a token
    const token = createToken(user._id)
    res.status(200).json({email, token})
  } catch (error) {
    res.status(400).json({error: error.message})
  }
}
```

Παράδειγμα σύνδεσης ενός χρήστη στο σύστημα.



4.4.3.4 RequireAuth - Middleware function

Η “requireAuth” είναι μία συνάρτηση Middleware, δηλαδή τρέχει ενδιάμεσα στο κύκλο αίτησης και απόκρισης της εφαρμογής, πριν την εκτέλεση της συνάρτησης που είναι υπεύθυνη για την διαχείριση της διαδρομής. Διαθέτει ως παραμέτρους το αντικείμενο αιτήματος (req), το αντικείμενο απόκρισης (res) και τη συνάρτηση η οποία θα εκτελεστεί μετά. Ο σκοπός της λειτουργίας αυτής είναι η προστασία των διαδρομών που καλύπτει από μη επαληθευμένους χρήστες. Ο τρόπος λειτουργίας της αναφέρεται στην παροχή της ένδειξης (TOKEN) από το αντικείμενο αιτήματος και στην αξιοποίηση της μεθόδου “verify” την οποία προσφέρει το πακέτο JsonWebToken για τον έλεγχο της εγκυρότητας του. Παρακάτω παρουσιάζεται ο κώδικας της συνάρτησης Middleware “requireAuth”.

```
const requireAuth = async (req, res, next) => {
  // verify user is authenticated
  const { authorization } = req.headers
  if (!authorization) {
    return res.status(401).json({error: 'Authorization token required'})
  }
  const token = authorization.split(' ')[1]
  try {
    const { _id } = jwt.verify(token, process.env.SECRET)

    req.user = await User.findOne({ _id }).select('_id')
    next()
  } catch (error) {
    console.log(error)
    res.status(401).json({error: 'Request is not authorized'})
  }
}
```

4.4.3.5 Επίλογος

Στην υποενότητα αυτή παρουσιάστηκε ο τρόπος υλοποίησης την επαλήθευσης των χρηστών του συστήματος.

4.4.4 Διαδρομές (Routes)

4.4.4.1 Εισαγωγή

Στην ενότητα αυτή αναλύεται αρχικά η αρχιτεκτονική η οποία χρησιμοποιήθηκε για την δημιουργία του περιεχομένου βάσει των διαδρομών του συστήματος και αφετέρου η παρουσίαση των αποκρίσεων των διαδρομών αυτών ως προς τα αιτήματα των χρηστών. Επιπλέον, για την καλύτερη κατανόηση του σχεδιασμού της εφαρμογής, αναλύονται συγκεκριμένα κομμάτια κώδικα τα οποία θεωρούνται σημαντικά.

Οι διαδρομές της ενότητας αυτής διαχωρίζονται σε 3 (τρεις) κατηγορίες:

- cryptoRoutes: Διαδρομές για την διαχείριση των δεδομένων των κρυπτονομισμάτων.

- watchlistRoutes: Διαδρομές για την διαχείριση των δεδομένων των κρυπτονομισμάτων της λίστας παρακολούθησης των χρηστών.
- exchangesRoutes: Διαδρομές για την διαχείριση των δεδομένων των ανταλλακτηρίων των κρυπτονομισμάτων.

Στον παρακάτω πίνακα παρουσιάζεται ο τρόπος σύνταξης των διαδρομών στο αρχείο server.js.

```
// routes
app.use("/api/coins", cryptoRoutes);
app.use("/api/exchanges", exchangesRoutes);
app.use("/api/watchlist", watchlistRoutes);
```

4.4.4.2 Διαδρομές κρυπτονομισμάτων (cryptoRoutes)

4.4.4.2.1 Εισαγωγή

Οι διαδρομές αυτές αποτελούν την διαχείριση των δεδομένων των κρυπτονομισμάτων. Εφαρμόζουν τις συναρτήσεις Middleware του πακέτου ApiCache και της συνάρτησης “requireAuth”. Το πακέτο ApiCache χρησιμοποιήθηκε για την προσωρινή αποθήκευση δεδομένων ώστε να μην εκτελούνται συνεχώς νέες αποκρίσεις στα αιτήματα που δημιουργούνται από τους χρήστες στον διακομιστή. Η συνάρτηση requireAuth χρησιμοποιήθηκε για την προστασία των διαδρομών αυτών από τους μη επαληθευμένους χρήστες.

```
let cache = apicache.middleware;
router.use(requireAuth);

router.get("/", cache('1 minute'), getCoins);
router.get("/trending", cache('2 minutes'), getTrendingCoins);
router.get("/search_coin", cache('2 minutes'), getSearchCoin);
router.get("/:id", cache('2 minutes'), getSingleCoin);
router.get("/:id/market_chart", cache('2 minutes'), getChart);
```

4.4.4.2.2 Αρχιτεκτονική εισαγωγής των δεδομένων των κρυπτονομισμάτων

Για την υλοποίηση των συναρτήσεων για την διαχείριση των δεδομένων των κρυπτονομισμάτων αξιοποιήθηκαν τα δεδομένα τα οποία προστέθηκαν και τροποποιούνται ανά τακτά χρονικά διαστήματα από το CoinGecko API. Ανάλογα την αίτηση του χρήστη, για παράδειγμα, για να πάρει δεδομένα για ένα ανταλλακτήριο, είτε για ένα συγκεκριμένο κρυπτονόμισμα, το σύστημα θα μπορούσε να χρησιμοποιήσει μόνο το δεδομένα που προέρχονται από το CoinGecko API. Ωστόσο, αν η αίτηση του χρήστη ήταν να πάρει δεδομένα για μία μεγάλη γκάμα κρυπτονομισμάτων με την δυνατότητα εναλλαγής σε διάφορες συναλλαγματικές ισοτιμίες, η εισαγωγή των δεδομένων από το CoinGecko API διαθέτει τον περιορισμό των 50 φορτώσεων ανά λεπτό και ότι κάθε φόρτωση δεδομένων θα γίνεται σε μία μόνο συναλλαγματική αξία. Έτσι, αν το σύστημα ήθελε να στηρίξει την προβολή της τιμής των κρυπτονομισμάτων σε διαφορετικές συναλλαγματικές ισοτιμίες, λόγω των παραπάνω περιορισμών, καθίσταται αδύνατο.

Για την επίλυση του προβλήματος αυτού, εφαρμόστηκε στο σχεδιασμό των διαδρομών (endpoints) η αξιοποίηση των δεδομένων συναλλαγματικής ισοτιμίας του Fixer API. Ουσιαστικά, ο τρόπος επίλυσης

του προβλήματος αναφέρεται στην τροποποίηση των δεδομένων που ζητάει ο χρήστης ανάλογα την προτίμηση του συναλλάγματος που τον ενδιαφέρει. Αρχικά, τα δεδομένα εισάγονται στην απόκριση του συστήματος με βάση το προκαθορισμένο νόμισμα το οποίο είναι το ευρώ και αν ο χρήστης έχει επιλέξει την εμφάνιση του με κάποιο άλλο νόμισμα, τότε τρέχει νέα μέθοδος η οποία κάνει την μετατροπή και επιστρέφει στην απόκριση του συστήματος τα δεδομένα σύμφωνα με το νόμισμα που ζήτησε ο χρήστης.

4.4.4.2.2.1 Παράδειγμα αρχιτεκτονικής με κομμάτια κώδικα

Για την επεξήγηση του τρόπου με τον οποίο υλοποιείται η παραπάνω αρχιτεκτονική των διαδρομών των κρυπτονομισμάτων θα αναλυθεί ο τρόπος λειτουργίας της διαδρομής του συστήματος για την παροχή μιας γκάμας κρυπτονομισμάτων που υποστηρίζει τη σελιδοποίηση και την εναλλαγή των τιμών των δεδομένων της σε διαφορετικά νομίσματα. Η διαδρομή η οποία θα αναλυθεί είναι η παρακάτω:

/api/coins?currency=usd&page=0

Η διαδρομή αυτή συντάσσεται προγραμματιστικά στο αρχείο διαδρομών cryptoRoutes ως:

router.get("/", cache('1 minute'), getCoins);

Η διαδρομή αυτή χρησιμοποιείται για την διαχείριση του αιτήματος της παροχής μίας γκάμας κρυπτονομισμάτων με τη δυνατότητα σελιδοποίησης και εναλλαγής των τιμών τους σε διαφορετικές νομισματικές ισοτιμίες μέσω της συνάρτησης “getCoins”, η οποία καλείται όταν λαμβάνεται το αίτημα αυτό και αλληλεπιδρά με το μοντέλο “Coins” (CoinsModel) για την αποθήκευση ή απόρριψη του αιτήματος αυτού. Παρακάτω αναλύεται ο τρόπος αλληλεπίδρασης της συνάρτησης “getCoins” με το μοντέλο “CoinsModel” για την για την διαχείριση του αιτήματος της σύνδεσης χρήστη με την βοήθεια τμημάτων κώδικα της εφαρμογής.

CoinsModel

Το παραπάνω κομμάτι κώδικα εμπεριέχεται μέσα στο CoinsModel και αποτελεί μια στατική μέθοδο η οποία ονομάζεται “UpdateCoins” και έχει ως στόχο την τροποποίηση των κρυπτονομισμάτων στο συνάλλαγμα που επιθυμεί - κάνει αίτημα ο χρήστης. Για την εφαρμογή του, δέχεται ως παραμέτρους τη σελίδα εμφάνισης (page) και το νόμισμα (currency) στο οποίο θέλει ο χρήστης να εμφανιστούν τα δεδομένα. Αρχικά, αποθηκεύονται στην μεταβλητή “exchange_rates” όλες οι συναλλαγματικές ισοτιμίες που έχουν εισαχθεί από το Fixer API. Έπειτα, μέσω της μεθόδου “updateMany” του πακέτου Mongoose γίνεται η τροποποίηση των κρυπτονομισμάτων και τέλος μέσω της μεθόδου “find” επιστρέφονται τα κρυπτονομίσματα στη συναλλαγματική αξία που ζήτησε ο χρήστης. Παρακάτω παρουσιάζεται σε μορφή κώδικα η στατική μέθοδος “UpdateCoins”.

```
geckoSchema.statics.UpdateCoins = async function (currency, page) {
  // 1) find all the exchange rates and save them in a variable
  const fetch_rates = await fetch_exchange_rates();
  const exchange_rates = fetch_rates[0].data;
  // 2) update the coins with the selected currency using the coins Model
  for (const [key, value] of Object.entries(exchange_rates)) {
    if (currency.to Uppercase() === key) {
      const update = await this.updateMany(
        { "data.market_cap_rank": { $ne: null } },
        [
          {
            $set: {
              [`${currency}.id`]: "$data.id",
              [`${currency}.symbol`]: "$data.symbol",
              "not shown": "to keep this page as short as possible"
            },
          },
        ],
        { upsert: true, order: true, multi: true }
      );
    }
  }
  // 3) find method to return all the updated trending coins with the selected currency
  const coinsPerPage = 30;
  const geckoData = await this.find({
    [`${currency}.market_cap_rank`]: { $ne: null },
  })
    .sort({ [`${currency}.market_cap_rank`]: 1 })
    .skip(page * coinsPerPage)
    .limit(coinsPerPage)
    .select(`${currency}`);

  return geckoData;
};
```

getCoins

Η συνάρτηση “getCoins” είναι μια ασύγχρονη λειτουργία η οποία καλείται όταν λαμβάνεται το αίτημα της εμφάνισης μια γκάμας κρυπτονομισμάτων από το σύστημα και χρησιμοποιείται για τον τρόπο διαχείρισης του αιτήματος αυτού. Περιέχει ως παραμέτρους το “req - request” (αίτημα) με τα αντικείμενα τα οποία εισέρχεται στο σύστημα και το “res - response” (απάντηση) που θα στείλει το σύστημα στο αίτημα αυτό. Ως προς την λειτουργία της συνάρτησης αυτής, αρχικά αν δεν δεχτεί στο αντικείμενο αιτήματος τη σελίδα εμφάνισης, την μετατρέπει στη σελίδα μηδέν η οποία θεωρείται η πρώτη. Επιπλέον, αν δεν δεχτεί το νόμισμα, τότε το μετατρέπει στην τιμή “data” η οποία είναι η προκαθορισμένη για την εμφάνιση των κρυπτονομισμάτων στην τιμή του ευρώ. Τέλος, χρησιμοποιεί τη στατική συνάρτηση του “CoinsModel” για την εμφάνιση των δεδομένων. Παρακάτω παρουσιάζεται σε μορφή κώδικα η μέθοδος “getCoins”.

```
const getCoins = async (request, response) => {
  //page=0 -> returns first 20 coins
  const page = request.query.page || 0;
  // "data" = eur -> saved directly from the coingecko fetch script
  // so data is eur, and if there's a currency -> changes
  // let -> so the value can be changed
  let currency = request.query.currency || "data";
  if (currency === "eur") currency = "data";
  try {
    const geckoData = await Coins.UpdateCoins(currency, page);
    response.status(200).json(geckoData);
  } catch (error) {
    response.status(400).json({ error: error.message });
  }
};
```

4.4.4.2.3 Διαδρομές

/api/coins/?currency=&page=

Η διαδρομή αυτή αποτελεί μέθοδο GET. Επιστρέφει το σύνολο 30 κρυπτονομισμάτων. Για την εκτέλεση της, μπορεί να δεχτεί ως παραμέτρους ερωτήματος το νόμισμα (currency) με το οποίο θέλει ο χρήστης να εμφανιστούν τα δεδομένα και η σελίδα εμφάνισης (page) για την εμφάνιση ενός συγκεκριμένου συνόλου δεδομένων. Κάθε σελίδα διαθέτει το ίδιο σύνολο αλλά διαφορετικά δεδομένα. Η διαδρομή αυτή έχει αναλυθεί παραπάνω ως παράδειγμα στην ενότητα - Αρχιτεκτονική διαχείρισης των δεδομένων των κρυπτονομισμάτων.

Παράδειγμα απόκρισης της διαδρομής:

/api/coins/?currency=usd&page=0

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.

```
{
        "_id": "6328d5f102b188932b5b0540",
        "usd": {
            "id": "bitcoin",
            "market_cap_rank": 1,
            "current_price": 17364.7685346,
            "market_cap": 334021286717.2178,
            "total_volume": 21900769739.153355,
            "high_24h": 17730.9410592,
            "not showing more": "to keep this page as short as possible"
        }
    },
    {
        "_id": "6328d5f102b188932b5b053f",
        "usd": {
            "id": "ethereum",
            "market_cap_rank": 2,
            "current_price": 1267.1044725,
            "market_cap": 152715779700.69235,
            "total_volume": 5848181498.93319,
            "high_24h": 1303.72384475,
             "not showing more": "to keep this page as short as possible"
        }
    },
    },…
```


/api/coins/trending?currency=

Η διαδρομή αυτή αποτελεί μέθοδο GET. Επιστρέφει το σύνολο 7 κρυπτονομισμάτων τα οποία θεωρούνται ως τα πιο διαδεδομένα στην κοινότητα της εφαρμογής CoinGecko. Για την εκτέλεση της, μπορεί να δεχτεί ως παράμετρο ερωτήματος το νόμισμα (currency) με το οποίο θέλει ο χρήστης να εμφανιστούν τα δεδομένα.

Παράδειγμα απόκρισης της διαδρομής:

/api/coins/trending?currency=jpy

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.

```
 {
        "_id": "632988b402b188932b8cf82b",
        "jpy": {
            "id": "aptos",
            "symbol": "apt",
            "name": "Aptos",
             "not showing more": "to keep this page as short as possible"
        }
    },
    {
        "_id": "6328d5f102b188932b5b0591",
        "jpy": {
            "id": "dogecoin",
             "not showing more": "to keep this page as short as possible"
        }
    },
    {
        "_id": "6328d5f202b188932b5b082e",
        "jpy": {
            "id": "evmos",
             "not showing more": "to keep this page as short as possible"
        }
    },...
}
```


/api/coins/search_coin?query=&currency=

Η διαδρομή αυτή αποτελεί μέθοδο GET. Χρησιμοποιείται για την αναζήτηση δεδομένων βάσει ενός ερωτήματος (query) ενός συγκεκριμένου αναγνωριστικού (id) κρυπτονομίσματος ή περισσότερων που διαθέτουν στο όνομα τους το συγκεκριμένο αναγνωριστικό. Για την εκτέλεση της, κρίνεται απαραίτητη η εφαρμογή της παραμέτρου ερωτήματος, ερώτημα (query), μέσω της οποίας πραγματοποιείται η αναζήτηση στα στοιχεία της βάσης δεδομένων. Επιπλέον, η συγκεκριμένη διαδρομή μπορεί να δεχτεί ως παράμετρο ερωτήματος το νόμισμα (currency) με το οποίο θέλει ο χρήστης να εμφανιστούν τα δεδομένα.

Παράδειγμα απόκρισης της διαδρομής:

/api/coins/search_coin?query=ethereum&currency=gbp

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.

```
{
        "_id": "6328d5f102b188932b5b053f",
        "gbp": {
            "id": "ethereum",
            "market_cap_rank": 2,
            "current_price": 1037.42426725,
            "not showing more": "to keep this page as short as possible"
        }
    },
    {
        "_id": "6328d5f102b188932b5b062a",
        "gbp": {
            "id": "ethereum-classic",
            "not showing more": "to keep this page as short as possible"
        }
    },
    {
        "_id": "6328d67c02b188932b5c6581",
        "gbp": {
            "id": "ethereum-pow-iou",
           "not showing more": "to keep this page as short as possible"
        }
    },
```


/api/coins/:id

Η διαδρομή αυτή αποτελεί μέθοδο GET. Επιστρέφει δεδομένα μόνο για το κρυπτονόμισμα του οποίου το αναγνωριστικό περιέχεται στην διαδρομή αντιπροσωπεύοντας την θέση “:id”. Το “:id” αποτελεί ένα μοναδικό αναγνωριστικό κρυπτονομίσματος και λειτουργεί ως παράμετρος . Η διαδρομή αυτή δεν απαιτεί την ύπαρξη της παραμέτρου ερωτήματος νόμισμα (currency) διότι τα δεδομένα που παράγονται από το CoinGecko API διαθέτουν την επιλογή της εμφάνισης τους με την μορφή πολλών διαφορετικών νομισμάτων.

Παράδειγμα απόκρισης της διαδρομής:

/api/coins/ethereum

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.

```
{
    "id": "ethereum",
    "symbol": "eth",
    "name": "Ethereum",
    "asset_platform_id": null,
    "not showing more": "to keep this page as short as possible"
}
```


/api/coins/:id/market_chart?days=&currency=

Η διαδρομή αυτή αποτελεί μέθοδο GET. Χρησιμοποιείται για την δημιουργία γραφημάτων για ένα κρυπτονόμισμα. Επιστρέφει δεδομένα μόνο για το κρυπτονόμισμα του οποίου το αναγνωριστικό περιέχεται στην διαδρομή αντιπροσωπεύοντας την θέση ":id". Το ":id" αποτελεί ένα μοναδικό αναγνωριστικό κρυπτονομίσματος και λειτουργεί ως παράμετρος . Η διαδρομή αυτή απαιτεί την ύπαρξη της παραμέτρου ερωτήματος νόμισμα (currency) και ημέρες (days) για την δημιουργία γραφήματος με διαφορετικά νομίσματα και σε συγκεκριμένο χρονικό περιθώριο.

Παράδειγμα απόκρισης της διαδρομής:

/api/coins/:id/market_chart?days=1&currency=eur

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.


```
{
    "prices": [
        [
            1671059128193,
            16684.157453877113
        ],
        "not showing more": "to keep this page as short as possible"
}
```


4.4.4.2.4 Επίλογος

Η υποενότητα αυτή αναφέρεται στον τρόπο λειτουργίας και στην παρουσίαση των διαδρομών διαχείρισης κρυπτονομισμάτων. Αρχικά, αναλύθηκε η αρχιτεκτονική του συστήματος για την υλοποίηση των διαδρομών αυτών και έπειτα η παρουσίασή τους μαζί με τα αποτελέσματα που προσφέρουν όταν ενεργοποιηθούν.

4.4.4.3 Διαδρομές ανταλλακτηρίων (exchangesRoutes)

4.4.4.3.1 Εισαγωγή

Οι διαδρομές αυτές αποτελούν την διαχείριση των δεδομένων των ανταλλακτηρίων των κρυπτονομισμάτων. Εφαρμόζουν τις συναρτήσεις Middleware του πακέτου ApiCache και της συνάρτησης "requireAuth". Η προσωρινή μνήμη cache διαρκεί 5 λεπτά διότι οι τιμές των ανταλλακτηρίων αλλάζουν με πιο αργή ταχύτητα από τις τιμές των κρυπτονομισμάτων. Η συνάρτηση

requireAuth χρησιμοποιήθηκε για την προστασία των διαδρομών αυτών από τους μη επαληθευμένους χρήστες.

```
let cache = apicache.middleware;


router.use(requireAuth);


router.get("/" ,cache('5 minutes'), getExchanges);
router.get("/search_exchange" ,cache('5 minutes'), getSearchExchange);
router.get("/:id" ,cache('5 minutes'), getSingleExchange);
```

4.4.4.3.2 Διαδρομές

/api/exchanges?page=

Η διαδρομή αυτή αποτελεί μέθοδο GET. Επιστρέφει το σύνολο 20 ανταλλακτηρίων κρυπτονομισμάτων. Για την εκτέλεση της, μπορεί να δεχτεί ως παράμετρο ερωτήματος τη σελίδα εμφάνισης (page) για την εμφάνιση ενός συγκεκριμένου συνόλου δεδομένων. Κάθε σελίδα διαθέτει το ίδιο σύνολο αλλά διαφορετικά δεδομένα.

Παράδειγμα απόκρισης της διαδρομής:

/api/exchanges?page=0

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.

```
{
        "_id": "631cdd9a3ab3ec1e9fc0946d",
        "data": {
            "id": "binance",
            "name": "Binance",
            "not showing more": "to keep this page as short as possible"
         }
}
```

/api/exchanges/:id

Η διαδρομή αυτή αποτελεί μέθοδο GET. Επιστρέφει δεδομένα μόνο για το ανταλλακτήριο κρυπτονομισμάτων του οποίου το αναγνωριστικό περιέχεται στην διαδρομή αντιπροσωπεύοντας την θέση “:id”. Το “:id” αποτελεί ένα μοναδικό αναγνωριστικό ανταλλακτηρίου και λειτουργεί ως παράμετρος .

Παράδειγμα απόκρισης της διαδρομής:

/api/exchanges/kraken

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.

```
{
    "name": "Kraken",
    "year_established": 2011,
    "country": "United States",
    "not showing more": "to keep this page as short as possible"
}
```


/api/exchanges/search_exchange?query=

Η διαδρομή αυτή αποτελεί μέθοδο GET. Χρησιμοποιείται για την αναζήτηση δεδομένων βάσει ενός ερωτήματος (query) ενός συγκεκριμένου αναγνωριστικού (id) ανταλλακτηρίου κρυπτονομισμάτων ή περισσότερων που διαθέτουν στο όνομα τους το συγκεκριμένο αναγνωριστικό. Για την εκτέλεση της, κρίνεται απαραίτητη η εφαρμογή της παραμέτρου ερωτήματος, ερώτημα (query), μέσω της οποίας πραγματοποιείται η αναζήτηση στα στοιχεία της βάσης δεδομένων.

Παράδειγμα απόκρισης της διαδρομής:

/api/exchanges/search_exchange?query=binance

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.


```
{
        "_id": "631cdd993ab3ec1e9fc08ff3",
        "data": {
            "id": "binance_us",
            "name": "Binance US",
            "not showing more": "to keep this page as short as possible"
        }
    },
    {
        "_id": "631cdd9a3ab3ec1e9fc0946d",
        "data": {
            "id": "binance",
            "name": "Binance",
            "not showing more": "to keep this page as short as possible"
    }...
}
```


4.4.4.3.3 Επίλογος

Η υποενότητα αυτή αναφέρεται στον τρόπο λειτουργίας και στην παρουσίαση των διαδρομών διαχείρισης των ανταλλακτηρίων των κρυπτονομισμάτων μαζί με τα αποτελέσματα που προσφέρουν όταν ενεργοποιηθούν.

4.4.4.4 Διαδρομές λίστας παρακολούθησης (watchlistRoutes)

4.4.4.4.1 Εισαγωγή

Οι διαδρομές αυτές αποτελούν την διαχείριση των δεδομένων των κρυπτονομισμάτων της λίστας παρακολούθησης των χρηστών. Εφαρμόζουν τη συνάρτηση “requireAuth” η οποία λειτουργεί ως Middleware και χρησιμοποιήθηκε για την προστασία των διαδρομών αυτών από τους μη επαληθευμένους χρήστες.

```
router.use(requireAuth);


router.post("/post_watchlist", postWatchlist);
router.get("/get_watchlist", getWatchlist);
router.delete("/delete_watchlist/:id", deleteFromWatchlist);
```

4.4.4.4.2 Αρχιτεκτονική διαχείρισης των δεδομένων της λίστας παρακολούθησης

Ο στόχος της ανάπτυξης των παραπάνω διαδρομών είναι η παροχή μίας ατομικής λίστας παρακολούθησης όπου οι χρήστες μπορούν να αποθηκεύσουν τα κρυπτονομίσματα που τους ενδιαφέρουν ώστε να έχουν άμεσα πρόσβαση στα δεδομένα τους, ιδίως όταν τα κρυπτονομίσματα αυτά είναι νέα στο χώρο και δεν διαθέτουν ακόμη κάποια τάξη (rank), οπότε καθίσταται δύσκολη η αναζήτηση τους σε κάποιον πίνακα κρυπτονομισμάτων.

Για την υλοποίηση της ατομικής λίστας παρακολούθησης, αρχικά δημιουργήθηκε ένα πεδίο πίνακα στο μοντέλο που περιέχει την δομή των δεδομένων των “CoinsModel”, το “user_ids”. Το πεδίο αυτό δέχεται τα αναγνωριστικά των χρηστών “_id” που έχουν προσθέσει στη λίστα παρακολούθησης τους το συγκεκριμένο κρυπτονόμισμα.

```
user_ids: [
      {
        type: String,
      },
    ],
```

Ο τρόπος εισαγωγής των αναγνωριστικών γίνεται με την αξιοποίηση της παρακάτω μεθόδου η οποία εκτελείται μέσω της διαδρομής:

/api/watchlist/post_watchlist

Αρχικά, μέσω του αντικειμένου αιτήματος (request) παρέχεται το αναγνωριστικό “id” του κρυπτονομίσματος που πρέπει να εισαχθεί στη λίστα παρακολούθησης και το αναγνωριστικό “_id” του χρήστη. Στη συνέχεια, στη μεταβλητή exists δηλώνεται αν υπάρχει το αναγνωριστικό “_id” του χρήστη στο πεδίο πίνακα “user_ids”. Αν δεν υπάρχει, τότε το αναγνωριστικό αποθηκεύεται στο πίνακα. Αν υπάρχει σημαίνει πως το συγκεκριμένο κρυπτονόμισμα υπάρχει ήδη στη λίστα παρακολούθησης του χρήστη οπότε το σύστημα δεν αποθηκεύει κάποιο στοιχείο στη βάση δεδομένων.

Η εμφάνιση των κρυπτονομισμάτων στη λίστα παρακολούθησης γίνεται αρχικά μέσω της λήψης του αναγνωριστικού “_id” του χρήστη από το αντικείμενο ερωτήματος. Έπειτα, ελέγχεται αν υπάρχει παράμετρος ερωτήματος για την μετατροπή των κρυπτονομισμάτων στο νόμισμα που επιθυμεί ο χρήστης. Τέλος, δημιουργείται έλεγχος αν το αναγνωριστικό του χρήστη “_id” υπάρχει στο πεδίο

πίνακα “user_ids” σε όλα τα κρυπτονομίσματα και εμφανίζονται αυτά στα οποία η απόκριση του ελέγχου ήταν αληθής.

```
const postWatchlist = async (request, response) => {
  const { id } = request.body;
  const { _id } = request.user;
  const exists = await Coins.find({
    id: id,
    user_ids: { $eq: _id },
  }).select("_id");
  // find the coin by its id and check if the _id exists in user_ids array
  // if yes -> exists takes the coin _id data of the user_ids array
  // if not -> exists is an empty array
  if (exists.length === 0) {
    await Coins.findOneAndUpdate(
      { id: id },
      {
        $push: {
          user_ids: _id,
        },
      },
      { upsert: true, order: true }
    )
      .then((data) => response.status(200).json(data))
      .catch(() => {
        response.status(500).json({ error: "Could not fetch the documents" });
      });
  } else {
    response.status(201).json(`${id} is already in users watchlist`);
  }
};
```

4.4.4.4.3 Διαδρομές

/api/watchlist/get_watchlist/?currency=

Η διαδρομή αυτή αποτελεί μέθοδο GET. Επιστρέφει το σύνολο των κρυπτονομισμάτων τα οποία έχουν αποθηκευτεί από τους χρήστες στη λίστα παρακολούθησης τους. Για την εκτέλεση της, μπορεί να δεχτεί ως παράμετρο ερωτήματος το νόμισμα (currency) με το οποίο θέλει ο χρήστης να εμφανιστούν τα δεδομένα.

Παράδειγμα απόκρισης της διαδρομής:

/api/watchlist/get_watchlist/?currency=eur

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.

```
{
        "_id": "6328d5f102b188932b5b0540",
        "data": {
            "id": "bitcoin",
            "symbol": "btc",
            "not showing more": "to keep this page as short as possible"
        }
    },
    {
        "_id": "6328d5f102b188932b5b053f",
        "data": {
            "id": "ethereum",
            "symbol": "eth",
           "not showing more": "to keep this page as short as possible"
         }…
}
```


/api/watchlist/post_watchlist

Η διαδρομή αυτή αποτελεί μέθοδο POST. Προσθέτει ένα συγκεκριμένο κρυπτονόμισμα το οποίο δηλώνεται στο σώμα της διαδρομής (body) στη λίστα παρακολούθησης του χρήστη. Η συγκεκριμένη διαδρομή έχει αναλυθεί παραπάνω στην ενότητα - Αρχιτεκτονική διαχείρισης των δεδομένων των κρυπτονομισμάτων της λίστας παρακολούθησης των χρηστών.

Παράδειγμα απόκρισης της διαδρομής. Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.


```
{
    "_id": "6328d5f102b188932b5b0540",
    "id": "bitcoin",
    "not showing more": "to keep this page as short as possible"
}
```


/api/watchlist/delete_watchlist/:id

Η διαδρομή αυτή αποτελεί μέθοδο DELETE. Διαγράφει ένα συγκεκριμένο κρυπτονόμισμα από τη λίστα παρακολούθησης του χρήστη.

Παράδειγμα απόκρισης της διαδρομής:

/api/watchlist/delete_watchlist/ethereum

Εμφάνιση ελάχιστων αποτελεσμάτων για την διατήρηση της σελίδα όσο το δυνατόν πιο σύντομη.

```
{
    "_id": "6328d5f102b188932b5b053f",
    "id": "ethereum",
    "not showing more": "to keep this page as short as possible"
}
```


4.4.4.4.4 Επίλογος

Η υποενότητα αυτή αναφέρεται στον τρόπο λειτουργίας και στην παρουσίαση των διαδρομών διαχείρισης των δεδομένων της λίστας παρακολούθησης των χρηστών. Αρχικά, αναλύθηκε η αρχιτεκτονική του συστήματος για την υλοποίηση των διαδρομών αυτών και έπειτα η παρουσίασή τους μαζί με τα αποτελέσματα που προσφέρουν όταν ενεργοποιηθούν.

4.4.5 Διεπαφή Swagger

Για την δημιουργία της διεπαφής swagger υλοποιήθηκε ένα αρχείο σε μορφη JSON το οποίο ονομάζεται swagger.json και διαθέτει τις διευθύνσεις URL των τελικών σημείων (endpoints), δηλαδή τις περιγραφές, τις παραμέτρους και τις δομές απόκρισης για ολόκληρο το API. Μέσω του Swagger οι χρήστες μπορούν να χρησιμοποιήσουν τις υπηρεσίες ιστού RESTful που αναπαράγονται από το API και να δοκιμάσουν τον τρόπο λειτουργίας και τις δυνατότητές του αφού πρώτα δημιουργήσουν λογαριασμό και επαληθευτούν στο σύστημα. Η εμφάνιση της διεπαφής πραγματοποιείται μέσω της διαδρομής:

/api-docs/

4.5 Front - End

4.5.1 Εισαγωγή

Όπως έχει ήδη προαναφερθεί το Front-End δηλαδή η κωδικοποίηση από την πλευρά του πελάτη είναι το λογισμικό με το οποίο υλοποιείται η διεπαφή χρήστη μίας διαδικτυακής εφαρμογής. Ουσιαστικά, το Front-End αναλαμβάνει να αποδώσει την εικόνα του ιστότοπου, δηλαδή την μορφή εμφάνισης των δεδομένων που παρέχονται μέσω διαδρομών (routes) από την εφαρμογή διακομιστή και τις λειτουργίες με τις οποίες ένας χρήστης μπορεί να δημιουργεί αιτήματα προς την εφαρμογή διακομιστή για να ζητήσει, να δημιουργήσει, να τροποποιήσει και να διαγράψει δεδομένα από την βάση δεδομένων της εφαρμογής.

Για την υλοποίηση του Front-End προγραμματισμού εφαρμόστηκε η βιβλιοθήκη ανοιχτού κώδικα JavaScript, React. Η ενότητα αυτή αναφέρεται στην ανάλυση της αρχιτεκτονικής που εφαρμόστηκε για την υλοποίηση του Front-End προγραμματισμού. Πιο συγκεκριμένα, θα αναλυθεί ο τρόπος χρήσης της βιβλιοθήκης React Query ως μέθοδος ανάκτησης δεδομένων από την βάση δεδομένων μέσω διαδρομών και ο τρόπος διαχείρισης των δεδομένων αυτών μέσω του πλαισίου Context.

4.5.2 Το πλαίσιο Context

4.5.2.1 Εισαγωγή

Όπως έχει ήδη προαναφερθεί, το Context ορίζεται ως ένα πλαίσιο διαχείρισης δεδομένων το οποίο προσφέρει η React και επιτρέπει την παροχή καθολικών δεδομένων σε θυγατρικά components λύνοντας το πρόβλημα της συνεχόμενης μετάδοσης των props μεταξύ των components.

Για την υλοποίηση της παρούσας διαδικτυακής εφαρμογής κρίθηκε απαραίτητο η δημιουργία δύο Context. Το πρώτο Context ονομάζεται AuthContext και αναφέρεται στη διαχείριση των χρηστών που εγγράφονται ή συνδέονται στην εφαρμογή. Το δεύτερο ονομάζεται CurrencyContext και αναφέρεται στην παροχή μίας καθολικής μεταβλητής "currency" (νόμισμα) για την προβολή των δεδομένων στη συναλλαγματική αξία που ενδιαφέρει τον χρήστη.

Στην ενότητα αυτή αναλύεται η αρχιτεκτονική του AuthContext και του CurrencyContext με την βοήθεια σχετικών κομματιών κώδικα. Παρακάτω εμφανίζεται με μορφή κώδικα ο τρόπος παροχής των δύο αυτών Context στο περιβάλλον της React στο αρχείο "index.js" καλύπτοντας το στοιχείο (component) "app.js" το οποίο χειρίζεται την εκκίνηση της εφαρμογής και τη δρομολόγηση όλων των στοιχείων (components) που δημιουργήθηκαν. Το AuthContext καλύπτει το CurrencyContext αφού για να επιλέξει κάποιος χρήστης το νόμισμα με το οποίο θα εμφανίζονται τα δεδομένα πρέπει πρώτα να έχει συνδεθεί στο σύστημα και να είναι επαληθευμένος.

```
const root = ReactDOM.createRoot(document.getElementById("root"));
const client = new QueryClient();
root.render(
  <React.StrictMode>
    <AuthContextProvider>
      <CurrencyContextProvider>
          <QueryClientProvider client={client}>
            <App />
          </QueryClientProvider>
      </CurrencyContextProvider>
    </AuthContextProvider>
  </React.StrictMode>
);
```

4.5.2.2 AuthContext - Επαλήθευση χρηστών

4.5.2.2.1 Αρχιτεκτονική

Το Context επαλήθευσης χρηστών (AuthContext) χρησιμοποιείται για την διαχείριση των λειτουργιών των χρηστών της εφαρμογής. Διαθέτει τις λειτουργίες σύνδεσης, εγγραφής και αποσύνδεσης χρήστη αλλά και την καθολική μεταβλητή "user" η οποία διαθέτει το "email" και το "token" του επαληθευμένου χρήστη που εγγράφεται ή συνδέεται στην εφαρμογή. Η καθολική μεταβλητή "user" είναι πολύ σημαντική. Αρχικά, μόλις επιτευχθεί σύνδεση ή εγγραφή ενός χρήστη, η μεταβλητή "user" αποθηκεύεται στο Local Storage του περιηγητή που χρησιμοποιεί ο χρήστης και κρατάει τις τιμές του (email,token) για το διάστημα τριών ημερών ώστε να μην απαιτείται να συνδέεται κάθε φορά ξανά στην εφαρμογή. Επιπλέον, η ύπαρξη του πεδίου "token" ως πεδίο της καθολικής μεταβλητής "user" είναι ιδιαίτερα σημαντική για δύο λόγους. Ο πρώτος λόγος αναφέρεται στην προστασία των ιστοσελίδων του ιστότοπου αφού η εμφάνιση τους επιτυγχάνεται μόνο αν η μεταβλητή "user" περιέχει τιμή στα πεδία της και ειδικότερα στο πεδίο "token" το οποίο είναι απαραίτητο για την δημιουργία αιτημάτων προς τον διακομιστή αφού κάθε διαδρομή απαιτεί το αναγνωριστικό αυτό "token" ενός επαληθευμένου χρήστη για να εκπληρωθεί. Ο δεύτερος λόγος αναφέρεται στην ευκολία παροχής και αξιοποίησης του πεδίου αυτού σύστημα, διότι είναι πεδίο της καθολικής μεταβλητής "user" και μέσω του AuthContext μπορεί να χρησιμοποιηθεί απευθείας σε οποιοδήποτε στοιχείο (component).

4.5.2.2.2 Παραδείγματα κώδικα

1) Παρουσίαση του τρόπου παροχής του AuthContext στην εφαρμογή μέσω του AuthContextProvider και της αυτόματης δοκιμής σύνδεσης χρήστη μέσω της μεταβλητής “user” η οποία παίρνει τις τιμές στα πεδία της από το Local Storage.

```
export const AuthContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(authReducer, {
    user: null
  })
  useEffect(() => {
    const user = JSON.parse(localStorage.getItem('user'))
    if (user) {
      dispatch({ type: 'LOGIN', payload: user })
    }
  }, [])
  return (
    <AuthContext.Provider value={{ ...state, dispatch }}>
      { children }
    </AuthContext.Provider>
  )
}
```

2) Εμφάνιση του τρόπου προστασίας των ιστοσελίδων του ιστότοπου από μη επαληθευμένους χρήστες. Αυτό συμβαίνει διότι μόνο αν η μεταβλητή “user” διαθέτει τιμές στα πεδία της, “emai” και “token” θα εκτελεστεί ο δρομολογητής (Router) για την εμφάνιση της ιστοσελίδας. Αν οι τιμές αυτές είναι λανθασμένες τότε η δημιουργία αιτημάτων προς τον διακομιστή δεν θα εκπληρωθεί και το σύστημα θα μεταφέρει το χρήστη στη σελίδα σύνδεσης ή εγγραφής.

```
{user ? (
        <Routes>
          <Route path="/" element={<Trading />} />
        </Routes>
  ) : (
       <Routes>
          <Route path="/" element={<login />} />
          </Routes>
}
```

3) Δημιουργία αιτήματος προς μία διαδρομή του διακομιστή με την αξιοποίηση του πεδίου “token” της καθολικής μεταβλητής “user”. Η μεταβλητή “user” παρέχεται στο σύστημα μέσω της συνάρτησης “usdAuthContext” η οποία την αξιοποιεί την εργοστασιακή μέθοδο της React, την “useContext”, για την χρησιμοποίηση του AuthContext. Αν η τιμή του πεδίου “token” είναι λανθασμένη τότε η δημιουργία αιτημάτων προς τον διακομιστή δεν θα εκπληρωθεί και το σύστημα θα μεταφέρει το χρήστη στη σελίδα σύνδεσης ή εγγραφής μέσω της συνάρτησης “logout” του AuthContext.

useAuthContext

```
import { AuthContext } from "../../context/AuthContext"
import { useContext } from "react"
export const useAuthContext = () => {
  const context = useContext(AuthContext)
  if(!context) {
    throw Error('useAuthContext must be used inside an AuthContextProvider')
  }
  return context
}
```

Δημιουργία αιτήματος

```
import { useAuthContext } from "../../hooks/authHooks/useAuthContext";
const { user } = useAuthContext();
const fetchCoins = (pageNumber, currency, user) => {
  return axios.get(`api/coins/?currency=${currency.value}&page=${pageNumber}`, {
    headers: {
      Authorization: `Bearer ${user.token}`,
    },
  });
};
```

Χρήση της συνάρτησης “logout”

```
  if (coinsIsError) {
    return logout();
  }
```

4.5.2.3 CurrencyContext

4.5.2.3.1 Αρχιτεκτονική

Το CurrencyContext χρησιμοποιήθηκε για την παροχή μίας καθολικής μεταβλητής “currency” (νόμισμα) η οποία χρησιμοποιείται για την προβολή των δεδομένων στη συναλλαγματική αξία που ενδιαφέρει τον χρήστη. Πιο συγκεκριμένα, για την δημιουργία αιτημάτων προς τον διακομιστή είναι απαραίτητο σε αρκετές διαδρομές να συμπεριλαμβάνεται το νόμισμα με το οποίο πρέπει να εμφανιστούν τα δεδομένα, έτσι η αξιοποίηση της καθολικής μεταβλητής “currency” (νόμισμα) χρησιμοποιήθηκε για την ευκολία παροχής της στο σύστημα στη δημιουργία αιτημάτων αλλά και στην επίτευξη του πλήρους συγχρονισμού των αιτημάτων αυτών προσφέροντας μια πολύ καλή εμπειρία χρήσης της εφαρμογής.

4.5.2.3.2 Παραδείγματα κώδικα

1) Εμφάνιση του τρόπου παροχής του CurrencyContext στο σύστημα μέσω του CurrencyContextProvider. Επιπλέον παρουσιάζεται η προκαθορισμένη (default) τιμή των πεδίων της μεταβλητής “currency”. Τα πεδία αυτά αντιπροσωπεύουν το νόμισμα ευρώ.

```
export const CurrencyContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(CurrencyReducer, {
    currency: {label:"EURO", value:"eur", symbol:"€"},
  });
  return (
    <CurrencyContext.Provider value={{ ...state, dispatch }}>
      {children}
    </CurrencyContext.Provider>
  );
};
```

2) Παρουσίαση της μέθοδος μετατροπής του “currency” σε άλλες συναλλαγματικές αξίες μέσω της συνάρτησης “dispatch” που υλοποιεί τη δράση (action) “SET_CURRENCY” του CurrencyContext.

```
  const { currency, dispatch } = useCurrencyContext();
  const matches = useMediaQuery("(min-width:800px)");
  const handleChange = (event, index) => {
    dispatch({
      type: "SET_CURRENCY",
      payload: {
        label: index.props.label,
        value: event.target.value,
        symbol: index.props.symbol,
      },
    });
  };
```

3) Δημιουργία αιτήματος προς μία διαδρομή του διακομιστή με την αξιοποίηση του πεδίου “value” της καθολικής μεταβλητής “currency”. Η μεταβλητή “currency” παρέχεται στο σύστημα μέσω της συνάρτησης “useCurrencyContext” η οποία αξιοποιεί την εργοστασιακή μέθοδο της React, την “useContext”, για την χρησιμοποίηση του currencyContext.

useCurrencyContext

```
import { CurrencyContext } from '../../context/CurrencyContext'
import { useContext } from 'react'
export const useCurrencyContext = () => {
  const context = useContext(CurrencyContext)
  if (!context) {
    throw Error('useCurrencyContext must be used inside an CurrencyContextProvider')
  }
  return context
}
```

Δημιουργία αιτήματος

```
import { useCurrencyContext } from "../../hooks/currencyHooks/useCurrencyContext";
const { currency } = useCurrencyContext();
const fetchCoins = (pageNumber, currency, user) => {
  return axios.get(`api/coins/?currency=${currency.value}&page=${pageNumber}`, {
    headers: {
      Authorization: `Bearer ${user.token}`,
    },
  });
};
```

4.5.2.4 Επίλογος

Στην ενότητα αυτή, έγινε ανάλυση της αρχιτεκτονικής των δύο Context της εφαρμογής, AuthContext και CurrencyContext και παρουσιάστηκαν τμήματα κώδικα που εμφανίζουν τους διάφορους τρόπους δράσης τους.

4.5.3 React Query

4.5.3.1 Εισαγωγή

Για την ανάκτηση και διαχείριση των δεδομένων της παρούσας διαδικτυακής εφαρμογής, δηλαδή τον τρόπο εισαγωγής και διαχείρισης των τιμών των κρυπτονομισμάτων που αλλάζουν διαρκώς, ήταν αναγκαίο να επιτευχθούν λειτουργίες πλήρους συγχρονισμού και προσωρινής αποθήκευσης (cache). Για τον λόγο αυτό επιλέχθηκε η βιβλιοθήκη React Query η οποία διαθέτει εύκολη μεθοδολογία ως προς την ανάκτηση δεδομένων αλλά και πολλές εργοστασιακές λειτουργίες για την πλήρη διαχείριση τους. Παρακάτω παρουσιάζεται ο τρόπος εφαρμογής της React Query στο περιβάλλον της React και ένα σημαντικό παραδείγματα του τρόπου χρήσης της.

4.5.3.2 Query Client

Για την αξιοποίηση των λειτουργιών της React Query ως προς την ανάκτηση και διαχείριση των δεδομένων απαιτείται η δημιουργία της οντότητας “query-client” και η εφαρμογή της πάνω σε ένα σύνολο στοιχείων (components) της React. Στην παρούσα εφαρμογή, η δημιουργία της οντότητας Query Client υλοποιείται στο αρχείο “index.js” και περιλαμβάνει το στοιχείο (component) “app.js” το οποίο χειρίζεται την εκκίνηση της εφαρμογής και τη δρομολόγηση όλων των στοιχείων (components)

που δημιουργήθηκαν. Παρακάτω παρουσιάζεται το τμήμα κώδικα της εφαρμογής που επιτελεί την ενέργεια αυτή.

```
const root = ReactDOM.createRoot(document.getElementById("root"));
const client = new QueryClient();
root.render(
  <React.StrictMode>
    <AuthContextProvider>
      <CurrencyContextProvider>
          <QueryClientProvider client={client}>
            <App />
          </QueryClientProvider>
      </CurrencyContextProvider>
    </AuthContextProvider>
  </React.StrictMode>
);
```

4.5.3.3 Παράδειγμα χρήσης της React Query

Ανάκτηση των δεδομένων των κρυπτονομισμάτων

Όπως έχει προαναφερθεί, η αξιοποίηση της React Query για την ανάκτηση δεδομένων προϋποθέτει την εφαρμογή της εργοστασιακής της συνάρτησης “useQuery” η οποία δέχεται ως παραμέτρους το όνομα και τη μέθοδο που υλοποιεί την ανάκτηση των δεδομένων. Στο συγκεκριμένο παράδειγμα, το όνομα της συνάρτησης είναι το “crypto-coins” και η μέθοδος που εκτελείται είναι η “fetchCoins”.

Η μέθοδος “fetchCoins” πραγματοποιεί την δημιουργία αιτήματος προς την διαδρομή του διακομιστή που αναφέρεται στην ανάκτηση των δεδομένων των κρυπτονομισμάτων. Για την επίτευξη της απαιτείται η παροχή τριών παραμέτρων. Την παράμετρο “pageNumber”, δηλαδή τη σελίδα εμφάνισης των δεδομένων. Την “currency”, και πιο συγκεκριμένα την “currency.value” για το νόμισμα στο οποίο θέλει ο χρήστης να εμφανιστούν οι τιμές των κρυπτονομισμάτων και την “user” για την επαλήθευση του χρήστη που δημιουργεί το αίτημα μέσω του “token” που δημιουργείται από τη σύνδεση του στην εφαρμογή.

```
const fetchCoins = (pageNumber, currency, user) => {
  return axios.get(`api/coins/?currency=${currency.value}&page=${pageNumber}`, {
    headers: {
      Authorization: `Bearer ${user.token}`,
    },
  });
};
```

Για την επίτευξη της οργάνωσης και πλήρους συγχρονισμού των δεδομένων με τη διαδρομή διακομιστή που εκτελεί το αίτημα η μέθοδος “fetchCoins” κρίθηκε απαραίτητη η εφαρμογή των παρακάτω λειτουργιών της “useQuery”.

- “staleTime: 60000”: Όταν εκτελεστεί το αίτημα προς τον διακομιστή, τα δεδομένα για 1 λεπτό διαβάζονται μόνο από την κρυφή μνήμη χωρίς να επιτρέπεται συμβεί κανένα άλλο αίτημα δικτύου. Με το πέρασμα του ενός λεπτού, τότε μόνο υπάρχει η δυνατότητα δημιουργίας νέου αιτήματος.
- “refetchInterval: 60000”: Η λειτουργία αυτή αναφέρεται στην αυτόματη επανάκτηση δεδομένων δηλαδή την αυτόματη δημιουργία νέων αιτημάτων ανά 1 λεπτό. Η λειτουργία αυτή συνδέεται με την “staleTime” για την επίτευξη του πλήρη συγχρονισμού ροής των δεδομένων μεταξύ της εφαρμογής πελάτη και εφαρμογής διακομιστή.
- “select”: Η λειτουργία αυτή αποσκοπεί στην οργάνωση εμφάνισης των δεδομένων. Στο συγκεκριμένο παράδειγμα χρησιμοποιείται διότι στην περίπτωση που ο χρήστης επιλέξει την τιμή συναλλάγματος “eur” (ευρώ) απαιτείται η εναλλαγή του στην τιμή “data” για την εμφάνιση των δεδομένων των κρυπτονομισμάτων.

Επιπλέον, η συνάρτηση “useQuery” προσφέρει χρήσιμα στοιχεία για την ανάκτηση των δεδομένων όπως ο χρόνος φόρτωσης (isLoading) των δεδομένων. Παρόλα αυτά επειδή υπάρχει η μέθοδος StaleTime στην εφαρμογή πελάτη και η Cache στην εφαρμογή διακομιστή, τα δεδομένα δεν θα αλλάζουν με τον τρόπο που ορίζει η αρχιτεκτονική της εφαρμογής διότι ο χρόνος φόρτωσης θα είναι μηδενικός αφού τα δεδομένα θα υπάρχουν στην κρυφή μνήμη και έτσι η εφαρμογή θα κολλήσει. Για αυτό χρησιμοποιήθηκε το στοιχείο του χρόνου ανάκτησης των δεδομένων (isFetching), το οποίο είναι αληθής, δηλαδή φορτώνει, όσο τα νέα δεδομένα ανακτούνται από την εφαρμογή. Τέλος, εφαρμόζεται η κατάσταση στην οποία έχει προκληθεί κάποιο σφάλμα (isError) και όταν συμβεί εμφανίζει μήνυμα και αποσυνδέει τον χρήστη από την εφαρμογή. Παρακάτω παρουσιάζεται η κωδικοποίηση της συνάρτησης “useQuery”.

```
const {
  isLoading: coinsIsLoading,
  isError: coinsIsError,
  data: coins,
  isFetching: coinsIsFetching,
} = useQuery(
  ["crypto-coins", pageNumber, currency, user],
  () => fetchCoins(pageNumber, currency, user),
  {
    staleTime: 60000,
    refetchInterval: 60000,
    select: (coins) => {
      const cryptoCoins =
        currency.value === "eur"
          ? Object.values(coins.data.map((d) => d.data))
          : Object.values(coins.data.map((d) => d[currency.value]));
      return cryptoCoins;
    },
  });
```

4.6 Επίλογος

Στο κεφάλαιο αυτό αναλύθηκε η αρχιτεκτονική υλοποίησης της παρούσας διπλωματικής εργασίας. Λόγω της κατασκευής της η οποία ακολουθεί τα πρότυπα δομής της στοίβας MERN, η ανάλυση διεξήχθη σε 3 τμήματα. Αρχικά, παρουσιάστηκε η δομή με την οποία δημιουργήθηκαν τα δεδομένα στην βάση δεδομένων MongoDB. Έπειτα, επεξηγήθηκε η αρχιτεκτονική δημιουργίας του Back-End

Rest API. Η αρχιτεκτονική Back-End διαχωρίστηκε στην εισαγωγή δεδομένων μέσω σεναρίων, στις λειτουργίες επαλήθευσης χρηστών και στην παραγωγή διαδρομών για την αξιοποίηση των δεδομένων που παρέχει η βάση στην οποία είναι συνδεδεμένη. Τέλος, παρουσιάστηκε η αρχιτεκτονική με την οποία υλοποιήθηκε το Front-End. Αρχικά επεξηγήθηκε ο τρόπος παροχής των δεδομένων στην εφαρμογή με την βοήθεια του πλαισίου, React Query, και έπειτα ο τρόπος αξιοποίησης των δεδομένων αυτών μέσω της εργοστασιακού API της React, Context.

Κεφάλαιο 5ο: Παρουσίαση της διαδικτυακής εφαρμογής

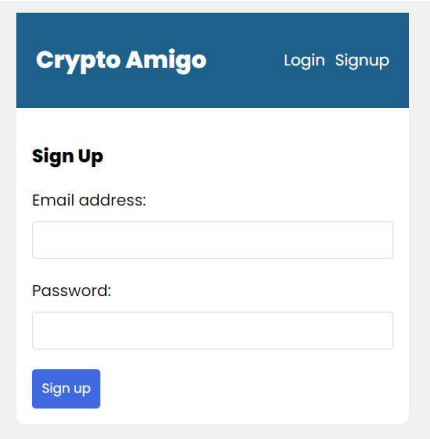
5.1 Εισαγωγή

Η παρούσα Διπλωματική Εργασία αναφέρεται στην κατασκευή μίας ολοκληρωμένης διαδικτυακής εφαρμογής η οποία προσφέρει την δυνατότητα στους χρήστες της να παρακολουθούν την εξέλιξη των τιμών των κρυπτονομισμάτων που επιθυμούν. Η εφαρμογή ακολουθεί τα πρότυπα σχεδιασμού που ανταποκρίνεται σε οθόνες οποιονδήποτε διαστάσεων (Responsive Design). Σε γενικές γραμμές, η εφαρμογή επιτρέπει στο χρήστη να έχει πρόσβαση σε ένα ευρύ φάσμα πληροφοριών όπως οι τρέχων τιμές των κρυπτονομισμάτων, οι αυξομειώσεις στην τιμή τους με βάση το χρονικό όριο που θέλει να θέσει ο χρήστης παρουσιάζοντας την αυτούσια τιμή αύξησης ή μείωσης σε κάποιο συγκεκριμένο νόμισμα που τον ενδιαφέρει. Επιπλέον, υπάρχει η δυνατότητα αποθήκευσης των κρυπτονομισμάτων που τον ενδιαφέρουν στη λίστα παρακολούθησης του. Τέλος, η εφαρμογή δίνει την δυνατότητα στο χρήστη να πάρει πληροφορίες για τα ανταλλακτήρια τα οποία επιτρέπουν την αγορά και πώληση κρυπτονομισμάτων ώστε να πετύχει τις καλύτερες προσφορές ως προς τις επενδύσεις του.

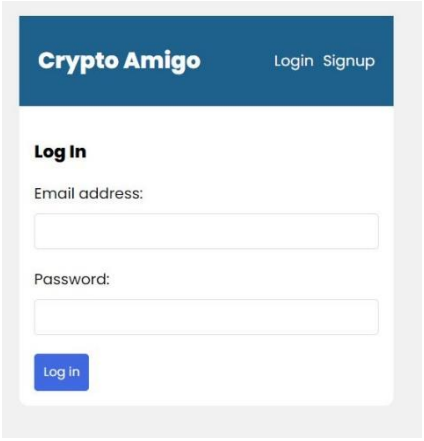
Στο κεφάλαιο αυτό θα διεξαχθεί μία παρουσίαση της διαδικτυακής εφαρμογής από την οπτική ενός συνδεδεμένου χρήστη. Η παρουσίαση διακρίνεται στην εμφάνιση των σελίδων της εφαρμογής και στην επεξήγηση των λειτουργιών τους.

5.2 Σελίδα σύνδεσης - εγγραφής χρήστη

Όπως έχει προαναφερθεί, για την εισαγωγή ενός χρήστη στην εφαρμογή και την αξιοποίηση των λειτουργιών της απαιτείται να έχει δημιουργήσει λογαριασμό. Για τον λόγο αυτό, όταν κάποιος χρήστης μεταβεί στο URL μιας οποιαδήποτε σελίδας της εφαρμογής και δεν έχει συνδεθεί τότε το σύστημα θα τον μεταφέρει στη σελίδα εγγραφής - σύνδεσης χρήστη. Ωστόσο, αν ο χρήστης έχει συνδεθεί στην εφαρμογή νωρίτερα, στο διάστημα τριών ημερών, τότε θα εκτελεστεί απευθείας η σελίδα με το συγκεκριμένο URL που επιλέγει ο χρήστης. Όπως έχει παρουσιαστεί νωρίτερα, η διατήρηση αυτή της σύνδεσης επιτυγχάνεται μέσω του JWT (Json Web Token).

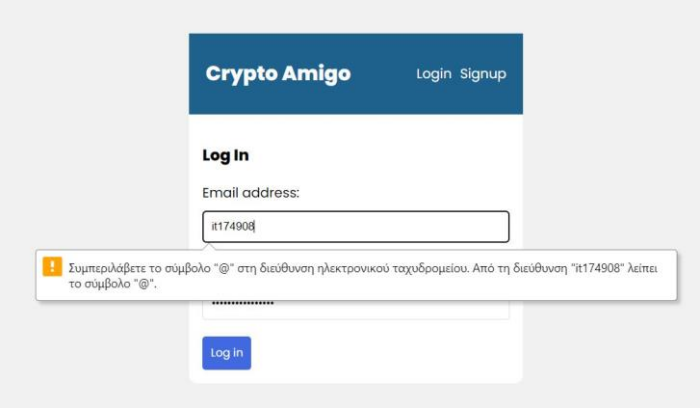


5.1: Σελίδα εγγραφής χρήστη

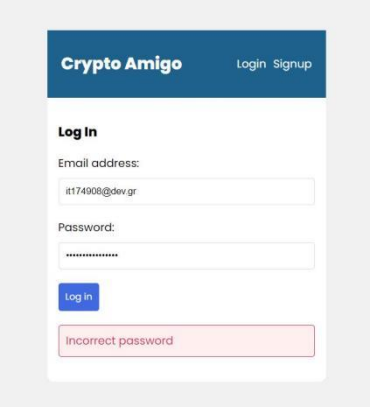


5.2: Σελίδα σύνδεσης χρήστη

Στη σελίδα εγγραφής και σύνδεσης χρήστη, ο χρήστης καλείται να πληκτρολογήσει τα στοιχεία “email” και “password” με τα οποία θέλει να δημιουργήσει λογαριασμό ή να συνδεθεί στην εφαρμογή μέσω του λογαριασμού αυτού αντίστοιχα. Αν εισάγει τα στοιχεία του σωστά και πατήσει το κουμπί “signup” ή αντίστοιχα “login”, τότε η εφαρμογή θα μεταφέρει τον χρήστη στην αρχική σελίδα. Αν προκύψει κάποιο λάθος στα στοιχεία του χρήστη, η εφαρμογή είναι προγραμματισμένη στο να ανταποκρίνεται σε οποιοδήποτε λάθος και να εμφανίζει κατάλληλο μήνυμα.



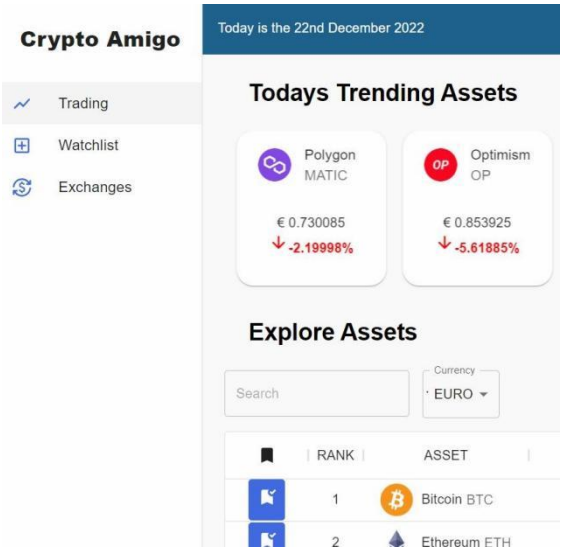
5.3: Λάθος καταγραφή της διεύθυνσης Email



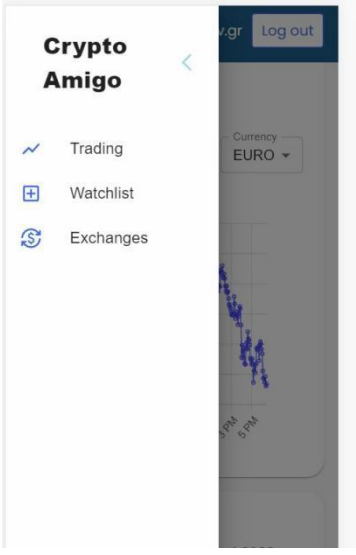
5.4: Λάθος καταγραφή του κωδικού πρόσβασης

5.3 Πλοήγηση (Navigation)

Μέτα την πραγματοποίηση εγγραφής ή σύνδεσης του χρήστη στην εφαρμογή εμφανίζεται σε κάθε σελίδα της εφαρμογής μία μπάρα πλοήγησης. Η μπάρα αυτή έχει προγραμματιστεί στο να εμφανίζεται διαφορετικά ανάλογα τις διαστάσεις της οθόνης που εκτελείται η εφαρμογή προσδίδοντας μία Responsive εμφάνιση. Τα στοιχεία που περιέχει αναφέρονται στις σελίδες τις εφαρμογής όπου η σελίδα “Trading” είναι η αρχική σελίδα της εφαρμογής η οποία περιέχει τις πληροφορίες για τα κρυπτονομίσματα. Το στοιχείο “Watchlist” αντιπροσωπεύει τη σελίδα λίστας παρακολούθησης κρυπτονομισμάτων του συνδεδεμένου χρήστη και το στοιχείο “Exchanges” τη σελίδα των ανταλλακτηρίων στα οποία ο χρήστης μπορεί να πάρει πληροφορίες για να πραγματοποιήσει την καλύτερη δυνατή επένδυση των χρημάτων του.



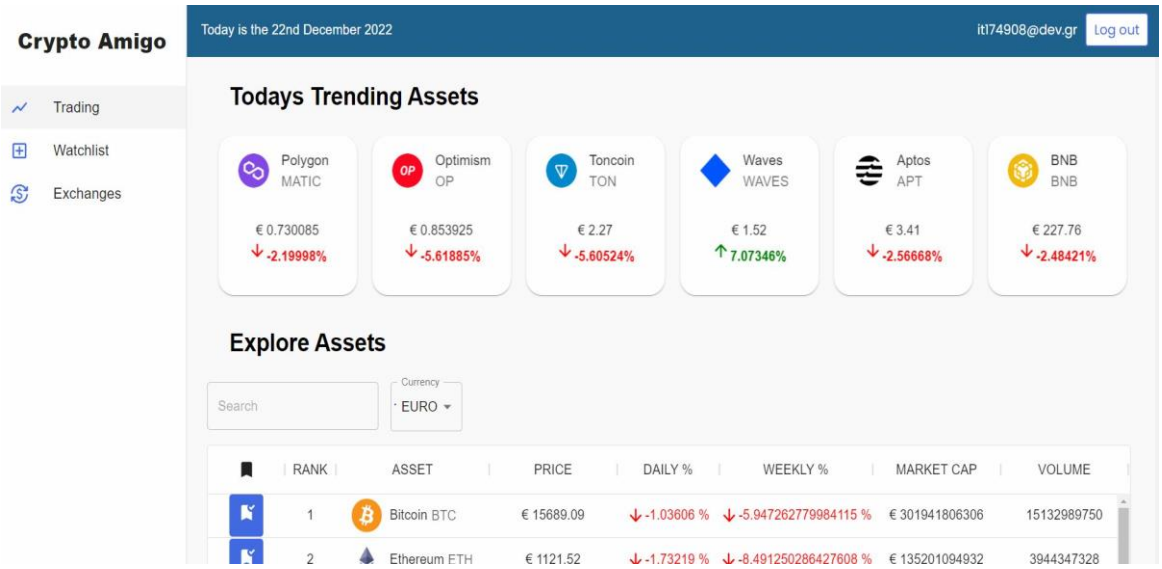
5.5: Πλοήγηση εφαρμογής για μεγάλες οθόνες



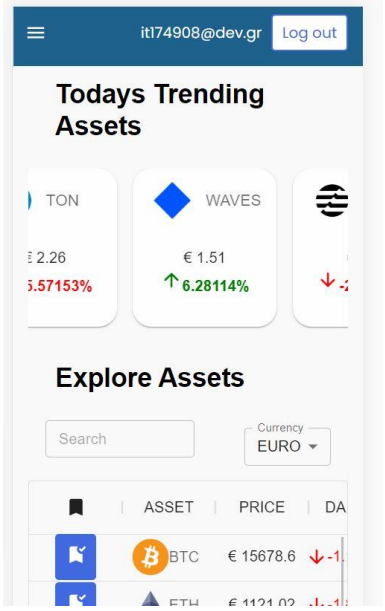
5.6: Πλοήγηση εφαρμογής για μικρές οθόνες

5.4 Αρχική Σελίδα (Trading)

Η αρχική σελίδα της εφαρμογής παρέχει τη δυνατότητα στο χρήστη να αποκτήσει πρόσβαση σε ένα σύνολο κρυπτονομισμάτων και να αποσπάσει γενικές πληροφορίες από αυτά. Η σελίδα αυτή έχει προγραμματιστεί στο να εμφανίζεται ομοιόμορφα ανάλογα τις διαστάσεις της οθόνης που εκτελείται η εφαρμογή προσδίδοντας μία Responsive εμφάνιση. Ως προς την εμφάνιση της, η σελίδα διακρίνεται σε δύο τμήματα τα οποία παρουσιάζονται αναλυτικότερα παρακάτω.

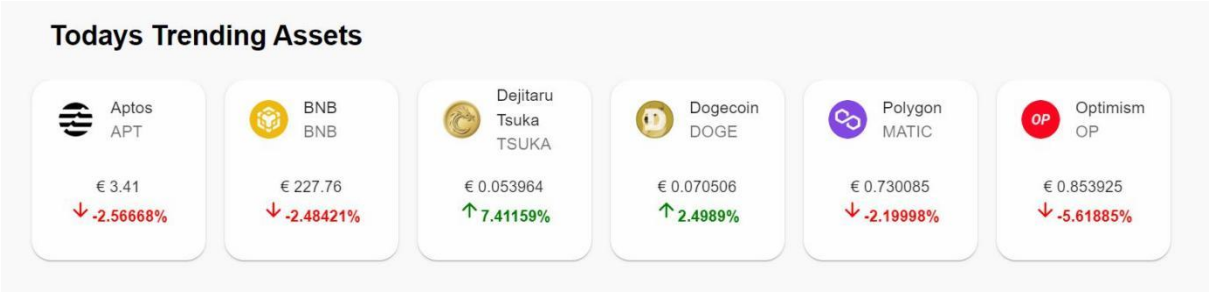


5.7: Αρχική σελίδα για μεγάλες οθόνες



5.8: Αρχική σελίδα για μικρές οθόνες

Το πρώτο τμήμα αναφέρεται στην εμφάνιση ενός καρουζέλ που περιέχει τα επτά πιο διαδεδομένα κρυπτονομίσματα σύμφωνα με την κοινότητα της εφαρμογής CoinGecko. Καθένα από τα κρυπτονομίσματα παρουσιάζεται ως καρτέλα με την τρέχουσα τιμή και με την μεταβολή της τρέχουσας τιμής τους σε μία ημέρα (24 ώρες).

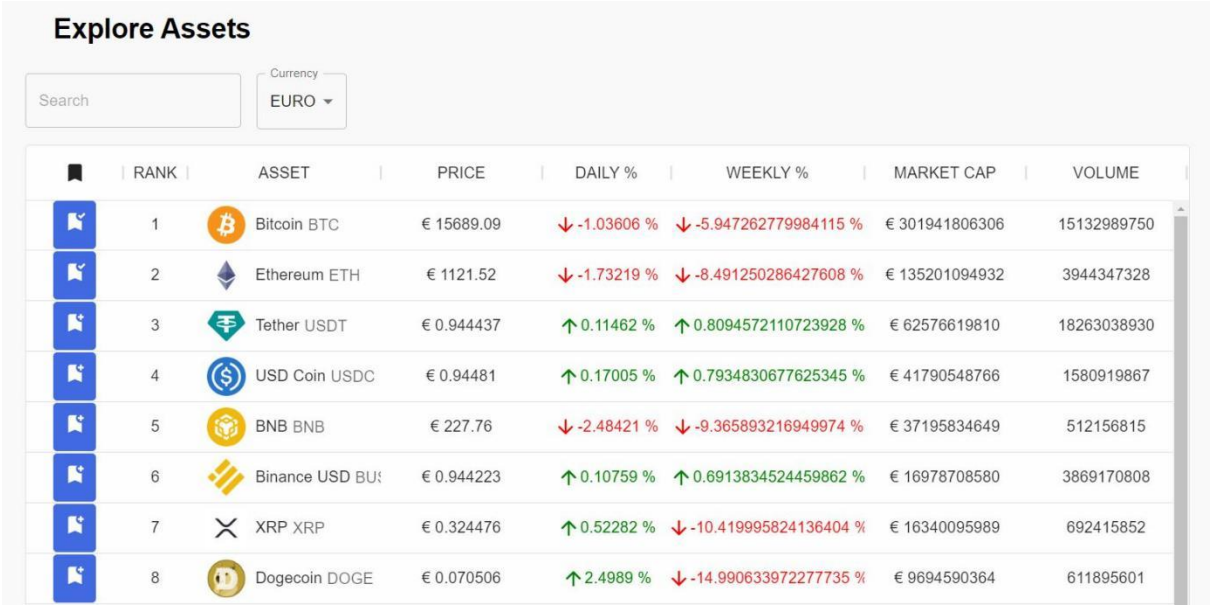


5.9: Καρουζέλ πιο διαδεδομένων κρυπτονομίσματων

Το δεύτερο τμήμα παρουσιάζει ένα πίνακα με γενικές πληροφορίες για όλα τα κρυπτονομίσματα της εφαρμογής. Για την εμφάνιση των κρυπτονομισμάτων χρησιμοποιείται σελιδοποίηση. Κάθε σελίδα παρέχει πληροφορίες για το σύνολο των 30 κρυπτονομισμάτων. Τα κρυπτονομίσματα αυτά εμφανίζονται ταξινομημένα βάσει της σειράς κατάταξης (Rank). Επιπλέον, διαθέτει τη λειτουργία ειδικής ταξινόμησης ανά των αριθμό κρυπτονομισμάτων που εμφανίζει στις κατηγορίες που παρουσιάζονται ανά στήλη του πίνακα. Τέλος, παρέχει τις λειτουργία αναζήτησης ενός κρυπτονομίσματος από όλα τα κρυπτονομίσματα που περιέχει η εφαρμογή και τη δυνατότητα εμφάνισης των κρυπτονομισμάτων βάσει της συναλλαγματικής αξίας που ενδιαφέρει το χρήστη.

Παρακάτω αναλύεται η σημασία των κατηγοριών του πίνακα:

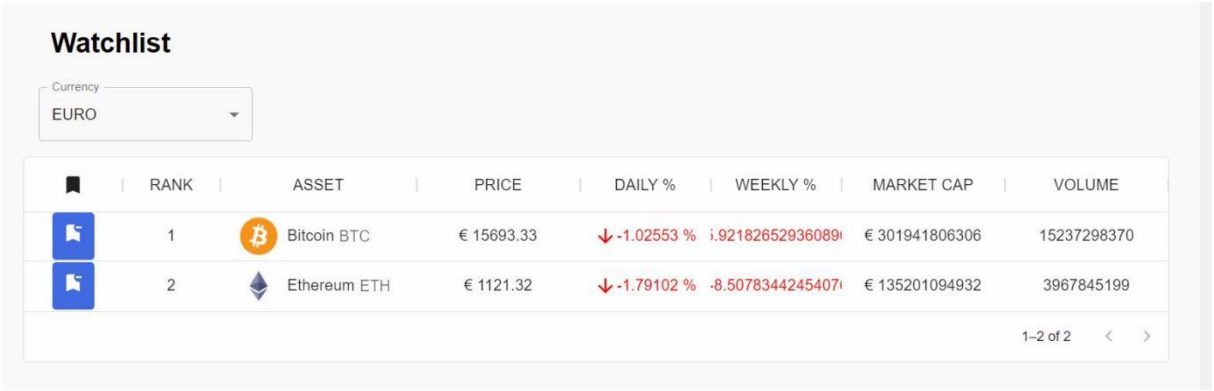
- **Εικονίδιο αποθήκευσης**: Το εικονίδιο αυτό αντιπροσωπεύει την αποθήκευση (ή κατάργηση) του συγκεκριμένου κρυπτονομίσματος στη λίστα παρακολούθησης.
- **Rank**: Αντιπροσωπεύει την κατάταξη των κρυπτονομισμάτων σύμφωνα με την κεφαλαιοποίηση που διαθέτουν. Όσο μεγαλύτερη η κεφαλαιοποίηση, τόσο μεγαλύτερη και η κατάταξη του κρυπτονομίσματος. Ο όρος την κεφαλαιοποίησης παρουσιάζεται παρακάτω.
- **Asset**: Εμφανίζει το εικονίδιο, το όνομα και το σύμβολο ενός κρυπτονομίσματος.
- **Price**: Εμφανίζει την τρέχουσα τιμή ενός κρυπτονομίσματος.
- **Daily %**: Εμφανίζει τη μεταβολή της τιμής ενός κρυπτονομίσματος σε 1 ημέρα (24 ώρες).
- **Weekly %**: Εμφανίζει τη μεταβολή της τιμής ενός κρυπτονομίσματος σε 1 εβδομάδα.
- **Market Cap**: Αντιπροσωπεύει τη νομισματική αξία της τεχνολογίας. Ορίζεται από το πολλαπλασιασμό της τρέχουσας τιμής με το συνολικό αριθμό κυκλοφορίας ενός κρυπτονομίσματος.
- **Volume**: Ορίζεται το σύνολο των συναλλαγών που πραγματοποιούνται από ένα κρυπτονόμισμα.



5.10: Πίνακας κρυπτονομισμάτων Αρχικής σελίδας

5.5 Σελίδα Λίστας Παρακολούθησης (Watchlist)

Η σελίδα αυτή παρέχει τη δυνατότητα στο χρήστη να αποκτήσει πρόσβαση στα κρυπτονομίσματα που έχει αποθηκεύσει στη λίστα παρακολούθησης του με την εμφάνιση ενός παρόμοιου πίνακα με αυτόν που επεξηγήθηκε παραπάνω ως τμήμα της Αρχικής Σελίδας της εφαρμογής. Οι μόνες διαφορές είναι ότι δεν υπάρχει η λειτουργία αναζήτησης και ότι η κατηγορία που περιέχει το εικονίδιο αποθήκευσης χρησιμοποιείται μόνο για την κατάργηση ενός κρυπτονομίσματος από τη λίστα παρακολούθησης.



5.11: Πίνακας κρυπτονομισμάτων της λίστας παρακολούθησης

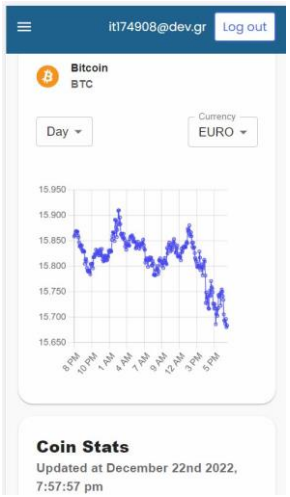
5.6 Σελίδα ενός κρυπτονομίσματος (Coin)

Η παρούσα σελίδα παρουσιάζει σημαντικές πληροφορίες για ένα συγκεκριμένο κρυπτονόμισμα. Η σελίδα αυτή έχει προγραμματιστεί στο να εμφανίζεται ομοιόμορφα ανάλογα τις διαστάσεις της οθόνης που εκτελείται η εφαρμογή προσδίδοντας μία Responsive εμφάνιση. Ως προς την εμφάνιση της, η σελίδα διακρίνεται σε τρία τμήματα τα οποία παρουσιάζονται αναλυτικότερα παρακάτω.

Το πρώτο τμήμα αναφέρεται στην εμφάνιση ενός διαγράμματος που παρουσιάζει τις τιμές ενός κρυπτονομίσματος κατά την διάρκεια μιας συγκεκριμένης περιόδου. Η παρουσία διαγράμματος είναι

απαραίτητη για τους επενδυτές ώστε να αξιοποιήσουν διάφορες τεχνικές ανάλυσης για να εντοπίσουν τις τάσεις της αγοράς και την αύξηση ή μείωση της τιμής ενός κρυπτονομίσματος. Στην εφαρμογή, παρέχεται η δυνατότητα αποτύπωσης των τιμών σε μια μέρα, εβδομάδα, μήνα, χρόνο αλλά και ολόκληρης της διάρκεια ύπαρξης του κρυπτονομίσματος. Επιπλέον, επιτρέπεται στο χρήστη να πληκτρολογήσει το σύνολο των ημερών πριν από την τελευταία τιμή που έχει καταγραφεί, δηλαδή την εμφάνιση του διαγράμματος πριν από συγκεκριμένο αριθμό ημερών. Τέλος, παρέχεται η δυνατότητα εμφάνισης του διαγράμματος στη νομισματική αξία που ενδιαφέρει το χρήστη.

5.12: Διάγραμμα τιμής ενός κρυπτονομίσματος για μεγάλες οθόνες

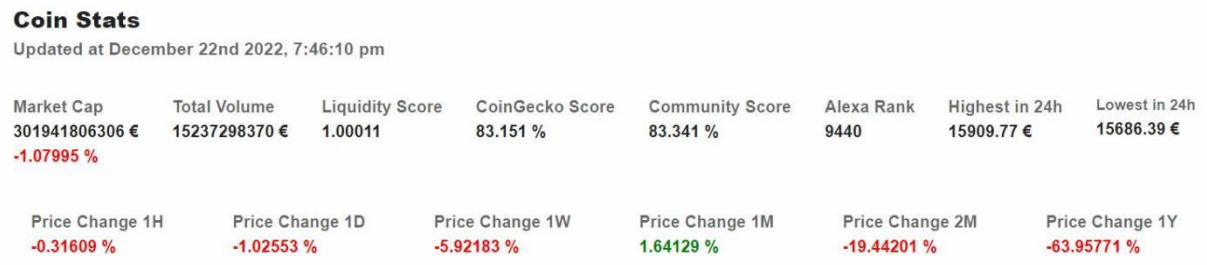


5.13: Διάγραμμα τιμής ενός κρυπτονομίσματος για μικρές οθόνες

Το δεύτερο τμήμα παρουσιάζει στατιστικά στοιχεία για ένα κρυπτονόμισμα. Τα στατιστικά στοιχεία μπορούν να παρουσιαστούν σύμφωνα με την επιλεγμένη τιμή της συναλλαγματικής αξίας που ενδιαφέρει το χρήστη. Παρακάτω αναλύεται η σημασία των στατιστικών στοιχείων που διαθέτει η εφαρμογή:

- **Market Cap:** Αντιπροσωπεύει τη νομισματική αξία της τεχνολογίας. Ορίζεται από το πολλαπλασιασμό της τρέχουσας τιμής με το συνολικό αριθμό κυκλοφορίας ενός κρυπτονομίσματος.

- **Total Volume:** Ορίζεται το σύνολο των συναλλαγών που πραγματοποιούνται από ένα κρυπτονόμισμα.
- **Liquidity Score:** Πόσο εύκολα μπορεί να πραγματοποιηθεί αγοραπωλησία για ένα κρυπτονόμισμα χωρίς να αλλάξει η συνολική τιμή της αγοράς.
- **CoinGecko Score:** Υπολογίζεται σύμφωνα με τη συχνότητα συναλλαγών ενός κρυπτονομίσματος στην κοινότητα του CoinGecko.
- **Community Score:** Υπολογίζεται σύμφωνα με τη συχνότητα συναλλαγών ενός κρυπτονομίσματος στην κοινότητα του.
- **Alexa Rank:** Είναι μια τιμή μεταξύ 1 και 100.000 σε διάστημα τριών μηνών. Όσο μικρότερη είναι η τιμή, τόσο μεγαλύτερη δημοτικότητα διαθέτει ένα κρυπτονόμισμα στο διάστημα αυτό.
- **Highest in 24h:** Η υψηλότερη τιμή ενός κρυπτονομίσματος στο διάστημα 24 ωρών.
- **Lowest in 24h:** Η χαμηλότερη τιμή ενός κρυπτονομίσματος στο διάστημα 24 ωρών.
- **Price Change 1H:** Η ποσοστιαία μεταβολή της τιμής ενός κρυπτονομίσματος στο διάστημα 1 ώρας.
- **Price Change 1D:** Η ποσοστιαία μεταβολή της τιμής ενός κρυπτονομίσματος στο διάστημα 1 μέρας.
- **Price Change 1W:** Η ποσοστιαία μεταβολή της τιμής ενός κρυπτονομίσματος στο διάστημα 1 βδομάδας.
- **Price Change 1M:** Η ποσοστιαία μεταβολή της τιμής ενός κρυπτονομίσματος στο διάστημα 1 μήνα.
- **Price Change 2M:** Η ποσοστιαία μεταβολή της τιμής ενός κρυπτονομίσματος στο διάστημα 2 μηνών.
- **Price Change 1Y:** Η ποσοστιαία μεταβολή της τιμής ενός κρυπτονομίσματος στο διάστημα 1 χρόνου.



5.14: Στατιστικά στοιχεία ενός κρυπτονομίσματος

Το τρίτο τμήμα θεωρείται το πιο καινοτόμο τμήμα της εφαρμογής αφού αναφέρεται στην παροχή των πληροφοριών που μπορούν να βοηθήσουν το χρήστη να επιλέξει το κατάλληλο ανταλλακτήριο στο οποίο θα επενδύσει πάνω για ένα συγκεκριμένο κρυπτονόμισμα. Η λέξη “Ticker” αναφέρεται στο συμβολισμό ενός κρυπτονομίσματος. Στην εφαρμογή συνοδεύεται με το νόμισμα το οποίο ενδιαφέρει το χρήστη. Οι πληροφορίες που παρέχονται αντιπροσωπεύουν τα σημαντικότερα ανταλλακτήρια στα

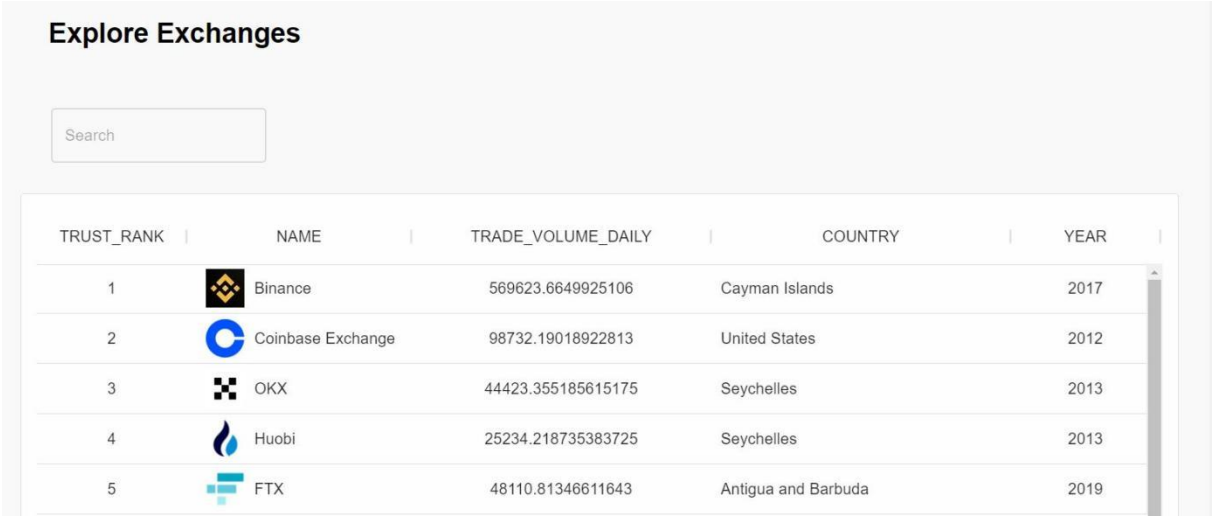
οποία μπορεί να επενδύσει ο χρήστης. Πιο συγκεκριμένα διακρίνονται στο “volume” και το “bid-ask difference” του ανταλλακτηρίου.

- **Volume (of the exchange)**: Ορίζεται το σύνολο των συναλλαγών που πραγματοποιούνται από ένα κρυπτονόμισμα σε ένα συγκεκριμένο ανταλλακτήριο.
- **Bid-Ask Difference**: Ορίζεται ως η υψηλότερη τιμή που παρέχει ένα ανταλλακτήριο μείον τη χαμηλότερη τιμή που ζητούν οι επενδυτές. Η τιμή αυτή παρουσιάζεται ποσοστιαία και όσο μικρότερη η τιμή, τόσο πιο κερδισμένος είναι ο επενδυτής στην αγορά ενός κρυπτονομίσματος.

5.15: Πληροφορίες ανταλλακτηρίων για την αγορά ενός κρυπτονομίσματος

5.7 Σελίδα ανταλλακτηρίων (Exchanges)

Η σελίδα των ανταλλακτηρίων της εφαρμογής παρέχει τη δυνατότητα στο χρήστη να αποκτήσει πρόσβαση σε ένα σύνολο ανταλλακτηρίων κρυπτονομισμάτων και να αποσπάσει γενικές πληροφορίες από αυτά. Παρουσιάζει ένα πίνακα με γενικές πληροφορίες για όλα τα ανταλλακτήρια της εφαρμογής. Για την εμφάνιση των ανταλλακτηρίων χρησιμοποιείται σελιδοποίηση. Κάθε σελίδα παρέχει πληροφορίες για το σύνολο των 20 ανταλλακτηρίων. Τα ανταλλακτήρια αυτά εμφανίζονται ταξινομημένα βάσει της σειράς κατάταξης εμπιστοσύνης (Trust Rank). Τέλος, η εφαρμογή παρέχει τις λειτουργία αναζήτησης ενός ανταλλακτηρίου από όλα τα ανταλλακτήρια που περιέχονται στην εφαρμογή και τη δυνατότητα ταξινόμησης βάσει των κατηγοριών που παρουσιάζονται ως στήλες στο πίνακα. Στο πίνακα αυτό οι κατηγορίες αντιστοιχούν στο όνομα, τόπο και χρονολογία δημιουργίας του ανταλλακτηρίου. Επιπλέον, εμφανίζονται οι κατηγορίες, κατάταξη εμπιστοσύνης που δηλώνει των αριθμό μελών που χρησιμοποιούν το συγκεκριμένο ανταλλακτήριο και το Volume Trade Daily που παρουσιάζει το σύνολο των συναλλαγών (μετοχές και συμβόλαια που ανταλλάσσονται μεταξύ αγοραστών και πωλητών) που έχουν επιτευχθεί σε μία μέρα.

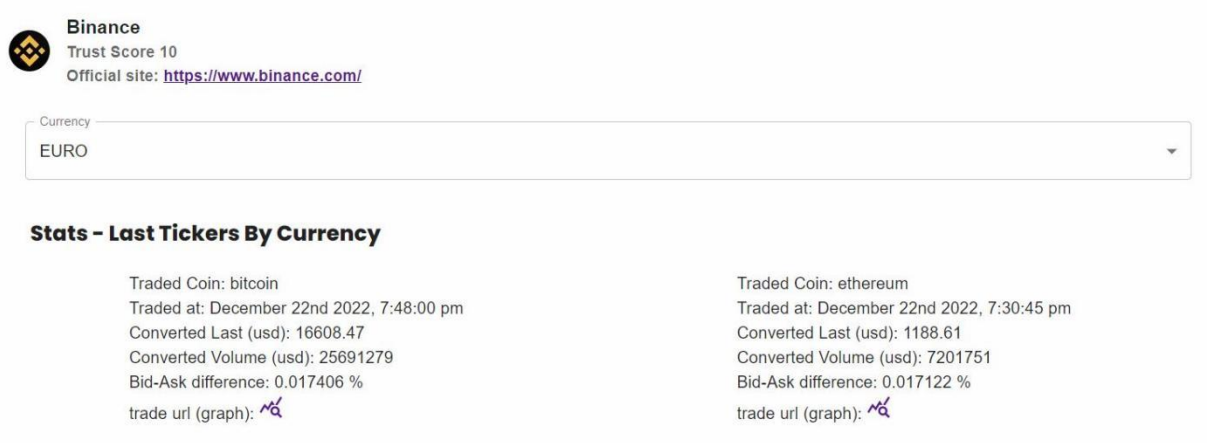


5.16: Πίνακας ανταλλακτηρίων κρυπτονομισμάτων

5.8 Σελίδα ενός ανταλλακτηρίου (Exchange)

Η παρούσα σελίδα παρουσιάζει τα στατιστικά στοιχεία των τελευταίων συναλλαγών που έχουν επιτευχθεί σε ένα συγκεκριμένο ανταλλακτήριο. Εμφανίζει το όνομα, τη βαθμολογία Η κεφαλίδα “Last Tickers By Currency” υποδηλώνει την παραπάνω έκφραση προσθέτοντας την αξιοποίηση της μεταβλητής της συναλλαγματικής αξίας ,την οποία ο κάθε χρήστης μπορεί να αλλάζει διαρκώς, για την εμφάνιση των αποτελεσμάτων βάσει του συγκεκριμένου νομίσματος που ενδιαφέρει το χρήστη. Η λειτουργία αυτή είναι πολύ σημαντική αφού προσφέρει σημαντικές πληροφορίες στο χρήστη για τα κρυπτονομίσματα τα οποία αποκτήθηκαν ή πωλήθηκαν ώστε να αναλύσει τις τιμές και να δημιουργήσει και ο ίδιος μια παρόμοια συναλλαγή αν πιστεύει πως θα του επιφέρει κέρδος. Οι κατηγορίες των πληροφοριών που προσφέρονται παρουσιάζονται παρακάτω:

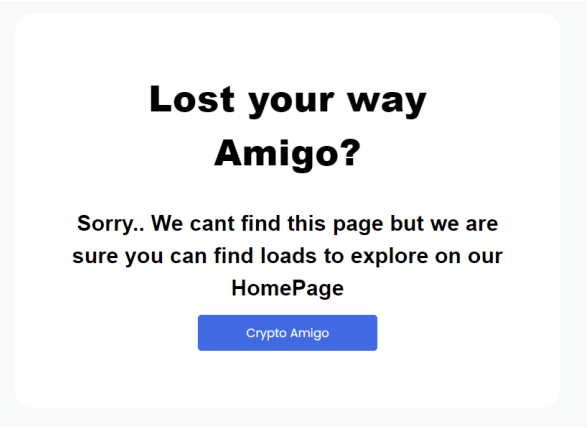
- **Traded Coin**: Το νόμισμα με το οποίο δημιουργήθηκε η συναλλαγή (αγορά ή πώληση).
- **Traded At**: Η ημερομηνία δημιουργίας της συναλλαγής.
- **Converted Last**: Το σύνολο των συναλλαγών που συμβαίνουν σε μία μέρα με ένα κρυπτονόμισμα. Όσο μεγαλύτερο το σύνολο τόσο και πιο διαδεδομένο είναι το κρυπτονόμισμα.
- **Converted Volume**: Το σύνολο των συναλλαγών που έχουν συμβεί με ένα κρυπτονόμισμα.
- **Bid-Ask Difference**: Ορίζεται ως η υψηλότερη τιμή που παρέχει ένα ανταλλακτήριο μείον τη χαμηλότερη τιμή που ζητούν οι επενδυτές. Η τιμή αυτή παρουσιάζεται ποσοστιαία και όσο μικρότερη η τιμή, τόσο πιο κερδισμένος είναι ο επενδυτής στην αγορά ενός κρυπτονομίσματος.
- **Trade URL**: Το διάγραμμα εφαρμογής της συναλλαγής που παρουσιάζεται. Το διάγραμμα αυτό αποτελεί υπερσύνδεσμο στη σελίδα του ανταλλακτηρίου στο οποίο έγινε η συναλλαγή και χρησιμοποιείται από τους επενδυτές με την εφαρμογή κάποιας τεχνικής ανάλυσης.



5.17: Τελευταίες συναλλαγές ενός ανταλλακτηρίου

5.9 Σελίδα σφαλμάτων (Error Page)

Η σελίδα αυτή χρησιμοποιείται για την ενημέρωση των επισκεπτών του ιστότοπου σε περίπτωση σφάλματος. Πιο συγκεκριμένα, η σελίδα αυτή εμφανίζεται όταν ο χρήστης προσπαθεί να συνδεθεί σε ένα τελικό σημείο (endpoint) του ιστότοπου το οποίο δεν υποστηρίζεται. Εμφανίζει ένα μήνυμα σφάλματος και προτείνει στον επισκέπτη να επισκεφτεί την αρχική σελίδα της εφαρμογής. Για την διευκόλυνση της άμεσης αυτής πρόσβασης στην αρχική σελίδα της εφαρμογής παρέχεται ένα κουμπί που πραγματοποιεί αυτή την μεταφορά.



5.18: Σελίδα σφαλμάτων

5.10 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκαν οι σελίδες της διαδικτυακής εφαρμογής από την οπτική ενός συνδεδεμένου χρήστη και αναλύθηκαν οι λειτουργίες και οι πληροφορίες που προσφέρουν.

Κεφάλαιο 6ο: Σύνοψη και Μελλοντικές Επεκτάσεις

6.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιαστεί μια σύντομη αλλά ολοκληρωμένη περιγραφή του στόχου και του λόγου ανάπτυξης παρούσας Διπλωματικής Εργασίας και θα αναφερθούν κάποιες μελλοντικές προτάσεις ως προς την επέκταση και βελτίωση της.

6.2 Σύνοψη

Ο στόχος της παρούσας Διπλωματικής Εργασίας ήταν η δημιουργία μίας ολοκληρωμένης διαδικτυακής εφαρμογής, η οποία θα προσφέρει στους χρήστες της την δυνατότητα να παρακολουθούν την εξέλιξη των τιμών των κρυπτονομισμάτων που επιθυμούν. Για την υλοποίηση της κρίθηκε απαραίτητη η ανάπτυξη μιας πολύ σωστά σχεδιασμένης αρχιτεκτονικής, αφού για την ολοκλήρωση της παρουσιάστηκαν πολύ σημαντικά θέματα μεταξύ των οποίων το δυσκολότερο ήταν ο τρόπος αξιοποίησης των δεδομένων του CoinGecko API ώστε να μην ξεπεραστεί το όριο των 50 φορτώσεων δεδομένων ανά λεπτό.

Ο λόγος που επέλεξα την δημιουργία της εφαρμογής αυτής ήταν να εξασκήσω τις γνώσεις μου για την ανάπτυξη μιας διαδικτυακής εφαρμογής από την αρχή αλλά και να εμβαθύνω στον τρόπο αλληλεπίδρασης με διαφορετικά αλλά και απαιτητικά application programming interfaces (APIs) σχεδιάζοντας μια ορθή αρχιτεκτονική.

Οι στόχοι που έθεσα εξαρχής, θεωρώ ότι έχουν εκπληρωθεί και μου έχουν προσφέρει σημαντικά εφόδια και γνώσεις για το μέλλον μου στην αναζήτηση εργασίας. Η εφαρμογή είναι διαθέσιμη στο Github και παρέχει οδηγίες για οποιονδήποτε θελήσει να την εκτελέσει και να τη χρησιμοποιήσει.

6.3 Μελλοντικές Επεκτάσεις

Στην υποενότητα αυτή παρουσιάζονται προτάσεις που μπορούν να αναπτυχθούν σε μεταγενέστερο στάδιο για την επέκταση και βελτίωση της εφαρμογής.

Αρχικά, θα μπορούσε να αξιοποιηθεί ένα νέο API για την παροχή των νέων (news) για την κοινότητα των κρυπτονομισμάτων. Ειδικότερα, ο χρήστης ο οποίος επισκέπτεται τη σελίδα ενός συγκεκριμένου κρυπτονομίσματος, θα ήταν πολύ χρήσιμο να μπορεί να παρακολουθήσει απευθείας τα νέα συμβάντα που έχουν δημιουργηθεί για το συγκεκριμένο κρυπτονόμισμα ώστε να κατευθύνει με ασφαλέστερο τρόπο τις επενδύσεις του.

Ως δεύτερη ιδέα για την επέκταση της παρούσας εφαρμογής θα μπορούσε να είναι η δήλωση μίας εικονικής επένδυσης πάνω σε ένα κρυπτονόμισμα (στο διάγραμμα ή σε κάποιο πεδίο δήλωσης ημέρας και ώρας) και η εμφάνιση των αποτελεσμάτων που θα παραχθούν από την επένδυση αυτή. Έτσι, οι χρήστες θα μπορούν να δημιουργούν εικονικές επενδύσεις και να παρακολουθούν με ευκολία τα αποτελέσματα της. Με τον τρόπο αυτό οι χρήστες εκπαιδεύονται ώστε να είναι πιο έτοιμοι για τη δημιουργία μίας νέας επένδυσης.

Τέλος, θα μπορούσε να αναπτυχθεί μία λειτουργία ψηφοφορίας όπου οι χρήστες της εφαρμογής επιλέγουν ένα κρυπτονόμισμα και μία συγκεκριμένη χρονική περίοδο που πιστεύουν ότι υπάρχει σημαντική ευκαιρία επίτευξης κέρδους με την αγορά ή πώληση του κρυπτονομίσματος αυτού. Ο κάθε χρήστης θα μπορεί να δημιουργεί μία τέτοια πληροφορία και οι υπόλοιποι θα μπορούν να την ψηφίσουν. Με τον τρόπο αυτό, επιτυγχάνεται η συνεργασία των χρηστών για την επίτευξη κέρδους.

6.4 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκε η περιγραφή του στόχου και του λόγου ανάπτυξης παρούσας Διπλωματικής Εργασίας και αναφέρθηκαν μελλοντικές προτάσεις ως προς την επέκταση και βελτίωση της.

ΒΙΒΛΙΟΓΡΑΦΊΑ

[1] Satoshi Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” Oct. 31, 2008. Accessed: Jan. 02, 2023. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] Usman W. Chohan, “A History of Bitcoin,” Feb. 2022, Accessed: Jan. 02, 2023. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3047875

[3] N. Daskalakis and P. Georgitseas, *An Introduction to Cryptocurrencies*. Abingdon, Oxon; New York, NY: Routledge, 2020. | Series: Contemporary issues in finance: Routledge, 2020. doi: 10.4324/9780429352584.

[4] M. Grabowski, *Cryptocurrencies*. Abingdon, Oxon ; New York, NY : Routledge, 2019. | Series Routledge focus on economics and finance: Routledge, 2019. doi: 10.4324/9780429201479.

[5] A. Wright and P. de Filippi, “Decentralized Blockchain Technology and the Rise of Lex Cryptographia,” *SSRN Electronic Journal*, 2015, doi: 10.2139/ssrn.2580664.

[6] P. Rosati and T. Čuk, “Blockchain Beyond Cryptocurrencies,” in *Disrupting Finance*, Cham: Springer International Publishing, 2019, pp. 149–170. doi: 10.1007/978-3-030-02330-0_10.

[7] C. Catalini and J. Gans, “Some Simple Economics of the Blockchain,” Cambridge, MA, Dec. 2016. doi: 10.3386/w22952.

[8] J. Golosova and A. Romanovs, “The Advantages and Disadvantages of the Blockchain Technology,” in *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, Nov. 2018, pp. 1–6. doi: 10.1109/AIEEE.2018.8592253.

[9] F. Fang *et al.*, “Cryptocurrency trading: a comprehensive survey,” *Financial Innovation*, vol. 8, no. 1, p. 13, Dec. 2022, doi: 10.1186/s40854-021-00321-6.

[10] G. Blokdyk, *Web Development Team A Complete Guide - 2020 Edition*. 2021.

[11] Eddy Wilson and Iriarte Koroliova, *MERN Quick Start Guide*. Packt Publishing , 2018.

[12] Vasan Subramanian, *Pro MERN Stack second edition*. Apress, 2019.

[13] “MERN Stack Explained.” https://www.mongodb.com/mern-stack (accessed Jan. 02, 2023).

[14] “Client-Server Overview.” https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview (accessed Jan. 02, 2023).

[15] “How Do HTML, CSS and JavaScript Work Together?” https://www.itonlinelearning.com/blog/how-do-html-css-and-javascript-work-together/ (accessed Jan. 02, 2023).

[16] B. Sanders, *Smashing HTML5*. Wiley, 2010.

[17] D. Cederholm and J. Zeldman, *CSS3 for Web Designers*. 2014.

[18] Douglas Crockford, *JavaScript: The Good Parts*. OReilly, 2008.

[19] “What is JSON and what is it used for?” https://stackoverflow.com/questions/383692/what-is-json-and-what-is-it-used-for (accessed Jan. 02, 2023).

[20] “NPM Node Package Manager.” https://docs.npmjs.com/about-npm (accessed Jan. 02, 2023).

[21] “React.” https://reactjs.org/? (accessed Jan. 02, 2023).

[22] R. Wieruch, *The Road to Learn React: Your Journey to Master Plain Yet Pragmatic React. Js*. CreateSpace Independent Publishing Platform, 2018.

[23] “The Best Guide to Know What Is React.” https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs (accessed Jan. 02, 2023).

[24] “Components and Props.” https://reactjs.org/docs/components-and-props.html (accessed Jan. 02, 2023).

[25] “Using the State Hook.” https://reactjs.org/docs/hooks-state.html (accessed Jan. 02, 2023).

[26] “Building Your Own Hooks.” https://reactjs.org/docs/hooks-custom.html (accessed Jan. 02, 2023).

[27] “Window.localStorage.” https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (accessed Jan. 02, 2023).

[28] “Context”, Accessed: Jan. 02, 2023. [Online]. Available: https://reactjs.org/docs/context.html#gatsby-focus-wrapper

[29] “Tanstack React Query.” https://tanstack.com/query/v4/docs/react/overview (accessed Jan. 03, 2023).

[30] “MUI: The React component library you always wanted”, Accessed: Jan. 02, 2023. [Online]. Available: https://mui.com/

[31] “Introduction to the server side.” https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction (accessed Jan. 02, 2023).

[32] “What is an API?” https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces (accessed Jan. 02, 2023).

[33] “What are Web Services?” https://www.tutorialspoint.com/webservices/what_are_web_services.htm (accessed Jan. 02, 2023).

[34] “Representational State Transfer (REST).” https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (accessed Jan. 02, 2023).

[35] R. Fielding, “Representational state transfer,” *Archit*, pp. 76–85, 2000.

[36] Bojinov, Valentin, David Herron, and Diogo Resende, *Node.js Complete Reference Guide*. Packt Publishing, 2018. Accessed: Jan. 02, 2023. [Online]. Available: https://www.perlego.com/book/868369/nodejs-complete-reference-guide-discover-a-more-sustainable-way-of-writing-software-with-high-levels-of-reusability-and-collaboration-using-nodejs-pdf

[37] R. Archer, *Express.js: Web App Development with Node.js Framework*. CreateSpace Independent Publishing Platform, 2016.

[38] “Introduction to Mongoose for MongoDB.” https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/ (accessed Jan. 02, 2023).

[39] “Hashing in Action: Understanding bcrypt.” https://auth0.com/blog/hashing-in-action-understanding-bcrypt/ (accessed Jan. 02, 2023).

[40] “Introduction to JSON Web Tokens.” https://jwt.io/introduction (accessed Jan. 02, 2023).

[41] “apicache.” https://www.npmjs.com/package/apicache (accessed Jan. 02, 2023).

[42] C.J. Date, *An Introduction to Database Systems*. Pearson Education, 2006.

[43] “MongoDB Schema Design Best Practices.” https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/ (accessed Jan. 02, 2023).

[44] “What is Postman API Test.” https://www.encora.com/insights/what-is-postman-api-test (accessed Jan. 02, 2023).

[45] “Swagger and Swagger UI: What is it and Why is it a must for your APIs?” chakray.com/swagger-and-swagger-ui-what-is-it-and-why-is-it-a-must-for-your-apis/ (accessed Jan. 02, 2023).

[46] Danielle Ellis, “What is Swagger? A Beginner’s Guide,” Jun. 26, 2022. https://blog.hubspot.com/website/what-is-swagger (accessed Jan. 02, 2023).

[47] Robert C. Martin and Kevlin Henney, *Clean Architecture: A Craftsman’s Guide to Software Structure and Design*, 1st Edition. Pearson, 2017.

[48] Woojong Suh, *Web Engineering: Principles and Techniques*. IGI Global, 2004.

[49] “CoinGecko API.” https://www.coingecko.com/en/api/documentation (accessed Jan. 03, 2023).

[50] “Fixer API.” https://apilayer.com/marketplace/fixer-api (accessed Jan. 03, 2023).

[51] “MongoDB Atlas.” https://www.mongodb.com/docs/atlas/ (accessed Jan. 03, 2023).

Παράρτημα Α΄: Χρήση της εφαρμογής

Το παράρτημα αυτό αναφέρεται στον τρόπο εισαγωγής της παρούσας Διαδικτυακής Εφαρμογής μέσω της διαδικτυακής υπηρεσίας Render.com και χρήσης της στο Παγκόσμιο Ιστό.

Render.com

Για την εισαγωγή της Διαδικτυακής Εφαρμογής στο Παγκόσμιο Ιστό αξιοποιήθηκε η Διαδικτυακή Υπηρεσία, Render.com. Η υπηρεσία αυτή αναλαμβάνει την συνεργασία με διάφορους παρόχους cloud και προσφέρει την δυνατότητα δημιουργίας, ανάπτυξης και εκτέλεσης εφαρμογών σε cloud δωρεάν.

Για την ανάπτυξη της εφαρμογής στο Διαδίκτυο απαιτήθηκε η μεταφόρτωση των λογισμικών διακομιστή και πελάτη στις υπηρεσίες του Render. Για την επίτευξη της μεταφόρτωσης, το Render.com συνεργάζεται με τις υπηρεσίες της πλατφόρμας φιλοξενίας κώδικα, GitHub. Για τον λόγο αυτό τα λογισμικά μεταφορτώθηκαν αρχικά στη πλατφόρμα GitHub και έπειτα αναπτύχθηκαν ξεχωριστά μέσω των υπηρεσιών που προσφέρει το Render.com. Το λογισμικό του διακομιστή αναπτύχθηκε ως Υπηρεσία Ιστού (Web Service) και παρέχει τη διεπαφή Swagger για την δημιουργία ερωταποκρίσεων απευθείας με το διακομιστή. Το λογισμικό πελάτη αναπτύχθηκε ως στατικός ιστότοπος (Static Site) και παρέχει τη διεπαφή χρήστη της εφαρμογής.

Παρακάτω παρουσιάζονται τα URLs των λογισμικών που δημιουργήθηκαν για την χρήση της εφαρμογής μέσω του Render.com και την εμφάνιση του κώδικα μέσω του GitHub.

Render.com

- Διεπαφή Διακομιστή (Swagger): https://crypto-amigo-api.onrender.com/api-docs/
- Διεπαφή χρήστη: https://crypto-amigo.onrender.com

GitHub

- Λογισμικό Διακομιστή: https://github.com/kwstaspexli/crypto-amigo-api
- Λογισμικό Πελάτη: https://github.com/kwstaspexli/cryptoamigo